INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
DESIGNING HIGH PERFORMANCE PARALLEL SYSTEMS: A PROCESSOR-CLUSTER BASED APPROACH

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Debashis Basak, B.Tech., M.S.

The Ohio State University

1996

Dissertation Committee:

Dr. Dhabaleswar K. Panda
Dr. P. Sadayappan
Dr. D. N. Jayasimha

Approved by

Dr. Dhabaleswar K. Panda

Adviser
Department of Computer & Information Science
To my loving parents
ACKNOWLEDGMENTS

Many thanks go to my advisor Dhabaleswar Kumar Panda for his continued guidance, support, and encouragement. Panda’s vision and enthusiasm have served as an inspiration throughout my stay at Ohio State. He has been a great source of insight and an excellent sounding board for ideas. I am grateful to him for the tremendous time, energy, and wisdom that he invested in steering my research. He has been an excellent role model through his dedication to quality research.

I also thank my thesis committee members P. Sadayappan and D. N. Jayasimha for reading through this lengthy dissertation. Their detailed comments helped improve the content and presentation of this thesis.

I am also grateful to Abhijit K. Choudhury and Ellen Hahne from Bell Laboratories, Lucent Technologies for their continued encouragement and support in my efforts to become a good researcher. My thanks to William Hsu, M. T. Raghunath, and Craig Stunkel for discussions on processor-cluster based systems which have enriched this work.

I gratefully acknowledge the financial support provided by NSF Grant MIP-9309627 and by Ohio State University Seed Grant 221444 and a Presidential Fellowship.
I thank the other members of the PAC group, especially Rajeev Sivaram, Donglai Dai, and Ram Kesavan for the hours of discussion on topics ranging from movies to multicasting which provided food for thought. I thank Venkat, Karan, Carl, Ed, Radha, Menon, Debashree, Sandeep, Manas, Sonali, Sri, Meena, and Suri for making these past years greatly enjoyable. Finally, I thank my family, my parents and and sister, for their immeasurable love, encouragement, and support of my education. I feel very happy to share with my parents, the joy of writing this thesis.
VITA

May 24, 1969 ............................................................. Born - Calcutta, India

1991 ................................................................. B.Tech., Computer Science
Indian Institute of Technology
New Delhi, India

1991-1992 .......................................................... Graduate Fellow
Computer Science Department
The Ohio State University
Columbus, Ohio

1992 ................................................................. M.S., Computer Science
The Ohio State University
Columbus, Ohio

1992-1995 .......................................................... Graduate Research Associate
Computer Science Department
The Ohio State University
Columbus, Ohio

1995-1996 .......................................................... Presidential Fellow
Computer Science Department
The Ohio State University
Columbus, Ohio

Publications


Fields of Study

Major Field: Computer and Information Science
# Table of Contents

DEDICATION ................................................................. ii

ACKNOWLEDGMENTS ....................................................... iii

VITA .................................................................................... v

LIST OF TABLES .............................................................. xii

LIST OF FIGURES ............................................................ xiv

CHAPTER PAGE

I Introduction ........................................................................ 1

1.1 Processor-cluster based systems ........................................ 2

1.2 Packaging constraints ................................................... 4

1.3 Inter-processor communication ........................................ 4

1.4 Network-cluster interface .............................................. 5

1.5 Collective communication ............................................. 7

1.6 Organization of the thesis ............................................. 7

II Processor-cluster based parallel systems ............................ 11

2.1 Advantages from processor-cluster based design ............... 11

2.2 $k$-ary $n$-cube cluster-organization ............................... 14

2.2.1 Architectural Alternatives and Choices ....................... 16

2.3 Intra-cluster organizations ............................................ 17

2.3.1 Direct-network based clusters ................................. 18

2.3.2 MIN based clusters ............................................. 18

2.3.3 Bus-based clusters ............................................. 19

2.3.4 Star-based clusters ............................................. 19
2.4 Summary ................................................................. 20

III  Design space of processor-cluster based systems ......................... 21

  3.1 Enhanced design space with processor-clustering .................. 24
  3.2 Characterizing communication performance and bottlenecks ... 28
    3.2.1 Messaging Overheads .................................. 29
    3.2.2 Network bisection .................................. 32
    3.2.3 Determining communication performance - interplay of above
         constraints .............................................. 35
  3.3 Characterizing parallel systems based on communication performance 37
  3.4 Designing a system with a given number of processors ........ 41
    3.4.1 Deriving the balanced configuration for a given channel width 42
    3.4.2 Required scaling in channel width with larger cluster sizes . 43
    3.4.3 Inter-cluster interconnection cost .......................... 45
  3.5 Scaling systems to support larger number of processors ....... 48
    3.5.1 Deriving the largest system size supportable on a given inter-
         cluster network ........................................ 50
    3.5.2 Required increase in channel width for further scaling .... 52
  3.6 Experimental validation - pattern and application based approach . 54
    3.6.1 Experimental platform .................................. 54
    3.6.2 Traffic patterns/applications ............................ 55
    3.6.3 Designing a system with a given number of processors .... 56
    3.6.4 Scaling systems to support larger number of processors .... 59
  3.7 Summary ............................................................. 62

IV  Routing and communication interface design .................................... 64

  4.1 Routing alternatives .............................................. 69
    4.1.1 Absorb & retransmit (AB) Strategy ............................. 70
    4.1.2 No absorb at interface (NAB) ................................ 71
  4.2 Designing efficient consumption channel interface ............ 77
    4.2.1 Typical interface with single consumption channel ........ 78
    4.2.2 Consumption collision and domino-blocking effect .......... 80
    4.2.3 Design with multiple consumption channels ................ 81
    4.2.4 Supporting virtual consumption channels .................. 84
    4.2.5 Number of physical consumption channels supportable by
         router pinout ........................................ 85
  4.3 Analytical model for estimating consumption collision delay? ... 86
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1 Estimating probability of consumption collision</td>
<td>88</td>
</tr>
<tr>
<td>4.3.2 Estimating collision delay</td>
<td>89</td>
</tr>
<tr>
<td>4.3.3 Parameters affecting collision delay and benefits from more</td>
<td>90</td>
</tr>
<tr>
<td>consumption channels</td>
<td></td>
</tr>
<tr>
<td>4.4 Applying the model to determine required number of physical</td>
<td>92</td>
</tr>
<tr>
<td>consumption channels</td>
<td></td>
</tr>
<tr>
<td>4.4.1 Impact of routing - balanced versus unbalanced rates</td>
<td>92</td>
</tr>
<tr>
<td>4.4.2 Impact of message length</td>
<td>94</td>
</tr>
<tr>
<td>4.4.3 Impact of network dimensionality</td>
<td>95</td>
</tr>
<tr>
<td>4.5 Constraints affecting maximum message arrival rate and number of</td>
<td>96</td>
</tr>
<tr>
<td>physical consumption channels</td>
<td></td>
</tr>
<tr>
<td>4.5.1 Network bisection</td>
<td>96</td>
</tr>
<tr>
<td>4.5.2 Non-uniform traffic</td>
<td>98</td>
</tr>
<tr>
<td>4.5.3 Adaptivity and virtual multiplexing on network channels</td>
<td>99</td>
</tr>
<tr>
<td>4.5.4 Messaging overhead</td>
<td>99</td>
</tr>
<tr>
<td>4.5.5 Interface to processor bandwidth</td>
<td>100</td>
</tr>
<tr>
<td>4.5.6 Summary</td>
<td>102</td>
</tr>
<tr>
<td>4.6 Simulation experiments and results</td>
<td>103</td>
</tr>
<tr>
<td>4.6.1 Simulation Set-up</td>
<td>103</td>
</tr>
<tr>
<td>4.6.2 Impact of routing adaptivity</td>
<td>104</td>
</tr>
<tr>
<td>4.6.3 Impact of message length</td>
<td>106</td>
</tr>
<tr>
<td>4.6.4 Impact of system dimensionality</td>
<td>107</td>
</tr>
<tr>
<td>4.6.5 Impact of system size with uniform traffic</td>
<td>108</td>
</tr>
<tr>
<td>4.6.6 Impact of more virtual channels</td>
<td>109</td>
</tr>
<tr>
<td>4.6.7 Impact of messaging overhead</td>
<td>110</td>
</tr>
<tr>
<td>4.6.8 Impact of I-to-P bandwidth</td>
<td>111</td>
</tr>
<tr>
<td>4.7 Summary</td>
<td>115</td>
</tr>
<tr>
<td>V Integrated framework for design under packaging and technological</td>
<td>116</td>
</tr>
<tr>
<td>advancements</td>
<td></td>
</tr>
<tr>
<td>5.1 Demanded Network Performance</td>
<td>120</td>
</tr>
<tr>
<td>5.1.1 Effect of Processor and Interconnect Speed on Demanded</td>
<td>120</td>
</tr>
<tr>
<td>System Throughput</td>
<td></td>
</tr>
<tr>
<td>5.1.2 Demand on Average Network Latency</td>
<td>124</td>
</tr>
<tr>
<td>5.2 Packaging model for clustered systems</td>
<td>125</td>
</tr>
<tr>
<td>5.3 Parameterizing packaging technologies</td>
<td>128</td>
</tr>
<tr>
<td>5.3.1 Processor Board Technology</td>
<td>128</td>
</tr>
<tr>
<td>5.3.2 Board Pinout Technologies</td>
<td>130</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Channel Width Technology</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Router Pinout Technology</td>
</tr>
<tr>
<td>5.4</td>
<td>Impact of varying architectural parameters</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Offered Inter-Cluster Channel Width</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Supporting a Fixed Inter-Cluster Channel Width</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Designing with Under Utilization of Resources</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Offered Inter-Cluster Bisection</td>
</tr>
<tr>
<td>5.5</td>
<td>Latency-throughput performance model for k-ary n-cube cluster-c systems</td>
</tr>
<tr>
<td>5.5.1</td>
<td>The Model</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Comparing Performance of the Model with Simulations</td>
</tr>
<tr>
<td>5.6</td>
<td>Design Framework Methodology</td>
</tr>
<tr>
<td>5.7</td>
<td>Applying the framework</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Deriving Design-Feasible Configurations</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Deriving Good Configurations</td>
</tr>
<tr>
<td>5.7.3</td>
<td>Deriving the Best Configuration</td>
</tr>
<tr>
<td>5.8</td>
<td>Impact of varying packaging an demand parameters on the design process</td>
</tr>
<tr>
<td>5.8.1</td>
<td>Processor and Interconnect Technology</td>
</tr>
<tr>
<td>5.8.2</td>
<td>Increasing Supported Channel Width</td>
</tr>
<tr>
<td>5.8.3</td>
<td>Increasing Board Pinout Density</td>
</tr>
<tr>
<td>5.8.4</td>
<td>Increasing the Maximum Router Pinout</td>
</tr>
<tr>
<td>5.9</td>
<td>Summary</td>
</tr>
<tr>
<td>VI</td>
<td>Interplay between cluster organizations and collective communication algorithms</td>
</tr>
<tr>
<td>6.1</td>
<td>Processor-cluster organizations and communication costs</td>
</tr>
<tr>
<td>6.2</td>
<td>Evaluation of different processor-cluster organizations</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Deriving the time for broadcast</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Comparison of broadcast times</td>
</tr>
<tr>
<td>6.3</td>
<td>Broadcast on clustered meshes</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Single port model with multiple injection channels</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Illustrating U_mesh in terms of inter- and intra-cluster steps</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Usage of injection channels by U_mesh</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Illustrating clus_mesh</td>
</tr>
<tr>
<td>6.3.5</td>
<td>The clus_mesh algorithm</td>
</tr>
<tr>
<td>6.4</td>
<td>Comparison of Clus_mesh and U_mesh performance</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Impact of increasing number of clusters</td>
</tr>
</tbody>
</table>
6.4.2 Impact of increasing the cluster size .............................................. 185
6.5 Summary ................................................................................................. 186

VII Conclusions and suggested future research ........................................... 187

7.1 Summary of Research Contributions ...................................................... 187
7.2 Suggestions for future research ............................................................... 190
7.2.1 Short-Term problems ........................................................................ 190
7.2.2 Long-Term problems ........................................................................ 191

BIBLIOGRAPHY ........................................................................................... 193
# List of Tables

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Message Startup Overhead ($t_s$) on Some Recent Systems.</td>
</tr>
<tr>
<td>2</td>
<td>Achievable Asymptotic Messaging Bandwidth Per Processor and Transmission Overhead Per Byte, $t_p$.</td>
</tr>
<tr>
<td>3</td>
<td>Summary of Important Symbols and Notations Used in this Chapter.</td>
</tr>
<tr>
<td>4</td>
<td>Sample values of $\lambda$ for various representative combinations of processor and Interconnection Link speeds. A computation to communication ratio of $r = 0.5$ FLOPs/bit[64] is assumed.</td>
</tr>
<tr>
<td>5</td>
<td>Summary of Interplay Between Board Size ($b$), Pinout Density ($p_p$ or $p_s$), Cluster Size ($c$), and Inter-Cluster Dimensionality ($n$) to Maintain a Fixed Channel Width ($W'$).</td>
</tr>
<tr>
<td>6</td>
<td>Valid ranges of cluster size ($c$) for packaging constraints of $b_{max} = 8$, $p_p = 128$, $R = 250$, and three different supportable channel widths of $W' = 12, 24, \text{ and } 40$. Number of clusters/board ($b'$) is varied from 1 - 8.</td>
</tr>
<tr>
<td>7</td>
<td>Design Feasible Configurations to Build Systems with $N \approx 1024$ and 4096 Processors under the Packaging Parameters $p_p = 128$, $W' = 24$, $R = 250$, and $b_{max} = 8$. Good Configurations are Shown in Boldface and these Offer $m_{max} \geq \lambda/L = 0.015$ with $T_{max} = 200$ cycles.</td>
</tr>
<tr>
<td>8</td>
<td>Impact of varying demanded throughput ($\lambda$) on the design of a system with $N \approx 1024$ processors. Parameters $p_p = 128$, $W' = 24$, $R = 250$, and $b_{max} = 8$ are assumed.</td>
</tr>
</tbody>
</table>
9 Impact of increased channel width technology ($W' = 40$) on the design of a system with $N \approx 1024$ Processors. Parameters $p_p = 128$, $R = 250$, and $b_{max} = 8$ are assumed. ...................................................... 162

10 Impact of increasing pinout ($p_p$) on the design of a system with $N \approx 1024$ processors. Parameters $W' = 24$, $R = 250$, and $b_{max} = 8$ are assumed. ...................................................... 163

11 Impact of increased router pinout ($R = 500$) on the design of a system with $N \approx 1024$ Processors. Parameters $p_p = 128$, $W' = 40$, and $b_{max} = 8$ are assumed. ...................................................... 164

12 Important Symbols Used in Paper. ...................................................... 169

13 The time required in network cycles for the inter- and intra-cluster phases of broadcasting a data of size $m$ words in a clustered mesh using the U.mesh algorithm. The number of clusters in the system is $N/c$. The time for the intra-cluster phase of the broadcast varies with the cluster organization. The relationships $i \leq c$ and $b \leq c$ are assumed. ...................................................... 172

14 Cost of a broadcast in a linear array of fewer than 9 clusters. The parameters $S$ and $F$ denote inter- and intra-cluster communication costs, respectively, in network cycles, while the parameter $B$ denotes cost of broadcast inside a cluster. ...................................................... 182
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Typical example of a processor-cluster based system.</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Examples of parallel systems using computing nodes based on a) single processors and b) processor-clusters of size 4. The system in (b) interconnects the same number of processors on a smaller network.</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Examples of scaling a base parallel system using computing nodes based on a) single processors and b) processor-clusters of size 4. The system in (b) offers a more compact interconnection network.</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Two-level clustering with k-ary n-cube cluster-c organization.</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Four possible cluster interconnections under k-ary n-cube cluster-c organization.</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Depicting the enhanced design space for parallel system configurations using larger processor-clusters. For designing systems with more processors, traditional approach only considered varying the interconnect size leading to higher $R$. Using processor-clusters adds a second dimension of variability, the cluster size (c). The third dimension, $W$, represents the added flexibility of using higher bandwidth network channels.</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>The process of a message transfer from a source processor to a destination processor showing the various sub-components.</td>
<td>28</td>
</tr>
</tbody>
</table>
8 Latency vs. throughput plots with different message overhead costs ($t_s$). Results are shown for two different systems a) (4x4,1,2) and b) (8x8,1,2). The maximum throughput, corresponding to $t_s \to 0.0 \mu$sec, depicts the value of $\lambda_{nw} = 2.5$ and $\lambda_{nw} = 1.0$ messages/\musec, respectively, in the two systems. Interplay between $\lambda_{nw}$ and $\lambda_{su}$ leads to the observed value of $\lambda$.

9 a) A typical $R-c$ plane depicting three regions into which parallel system configurations fall under: $B$-boundary (balanced), $MC$-region (messaging overhead constrained), and $NC$-region (network-bisection constrained). b) Impact of increasing $W$ from $W1$ to a higher value $W2$, the $B$-boundary shifts outwards, resulting in a larger $MC$-region.

10 Three configurations on an iso-processor line for $N = 64$ on an $R-c$ plane with fixed channel width ($W = 1$ byte).

11 Latency vs. throughput plots for a) three configurations with $N = 64$ processors and channel width $W = 1$ byte: (8x8,1,1), (8x4,2,1), and (4x4,4,1) and b) (4x4,4,2) configuration with wider channel width ($W = 2$ bytes).

12 Increased channel width in the $Z$ configuration (from $W = 1$ to $W = 2$) leading to configuration $Z$ becoming balanced and offering $\lambda = \lambda_{nw}$.

13 a) Different options to scale a configuration $Y$ with 64 processors to a larger size of 128 processors. b) Scaling with the same topology but larger clusters ($Y$ to $Y3$ option).

14 Determining the largest system size that can be supported without degradation in performance on a given base network by using larger clusters. Results are shown for $t_s = 2.0$ and 5.0 \mu sec/message.

15 Studying the scalability of a balanced configuration (8x8,2,2) using larger cluster and proportionately larger channel width. A $t_s = 2.0 \mu$sec was assumed.

16 Studying the scalability of the approach of using larger clusters and proportionately larger channel widths. To ensure configurations not being $MC$ (i.e. being $B$ or $NC$), a very small $t_s = 0.25 \mu$sec was assumed.
Impact of using larger clusters in designing systems: deriving the balanced configuration. The normalized completion times are depicted for LU, BPC, and FFT on three configurations on ClusterSim and two configurations on the Cray T3D. On ClusterSim parameters of \( t_a = 2.0 \mu \text{sec/message} \) and \( W = 2 \) bytes, matching those on the T3D, are assumed. Number of processors is kept fixed at \( N = 64 \).

Matching performance with wider channels beyond the balanced configuration. The normalized completion times of FFT is shown on three configurations. Parameters \( N = 64 \) and \( t_s = 2.0 \mu \text{sec} \) are assumed.

Impact of using larger clusters in scaling a base system having under-utilized network bisection resources: determining largest system (cluster) size supportable on same network. Normalized completion times are shown for BPC and FFT on a \((8x8,1,2)\) base system and its scaled counterparts having larger clusters. The parameters of \( t_s = 5.0 \mu \text{sec/message} \) and \( W = 2 \) bytes are assumed.

Impact of using larger clusters in scaling a base system having no under-utilized network bisection resources: determining required increase in channel width. Normalized completion times are shown for FFT on a \((8x8,1,2)\) base system and its scaled counterparts having larger clusters and proportionately wider channels. The parameter \( t_s = 0.25 \mu \text{sec/message} \) is assumed.

An example of a deadlock in a \((2,1,5,1)\) system using dimension order routing with 2 virtual channels a) system connectivity with its virtual channels and b) channels occupied by messages leading to a deadlock.

Partition of the added virtual channel network under the modified-composite-routing scheme, to route messages entering the cluster interface to reach a destination inside the cluster. The routing is minimal. The partitioned virtual channel network is acyclic.

A two-dimensional mesh of size \(3x3\) (3-ary 2-cube). A typical router-processor interface is illustrated for a \(k\)-ary \(n\)-cube system with one consumption channel per node.
An example of consumption collision in a 5x5 mesh with one consumption channel under dimension-order (x-y) routing. Two messages M1 and M2 are destined towards the node $D_1$. A third message M3 destined to $D_2$ is blocked on M2. 

A router-processor interface supporting multiple consumption channels per node. 

Supporting multiple virtual consumption channels on a physical consumption channel. On the router side, the virtual channel implementation on a router network channel is extended to a consumption channel. On the interface side, each virtual consumption channel has message buffers associated with it. 

A simplified model for estimating probability of consumption collision and consumption delay with a) single consumption channel and b) multiple consumption channels. 

Demonstrating impact of more physical consumption channels at a node in a (16x16) 2D network system. Routing assumed is: a) dimension-order leading to unbalanced traffic along incoming channels and b) adaptive leading to more balanced traffic along incoming channels. 

Demonstrating impact of more physical consumption channels at a node in a (16x16) 2D network system with longer messages. Message lengths assumed are $L = 64$ and 256 bytes, respectively. 

Demonstrating maximum impact of more physical consumption channels at a node in two systems with different dimensionality: 2D and 3D, respectively. 

Impact of routing adaptivity and number of consumption channels on performance in a 4x4x4 system. Both uniform and non-uniform traffic were considered. 

Impact of message length and number of consumption channels on uniform traffic performance in a 4x4x4 system. Two different message length of $L = 64$ and 256 bytes, respectively, were considered.
Impact of dimensionality and number of consumption channels on system performance. Two different systems 8x8 and 4x4x4 were considered. 107

Impact of more consumption channels on uniform traffic performance on: two 2D system sizes of 8x8 and 16x16 and two 3D system sizes of 4x4x4 and 6x6x6. 109

Impact of more virtual lanes and consumption channels on performance in a 4x4x4 system. Three different values of v = 2, 4, and 6 were considered. 110

Impact of messaging overheads on maximum message rate and effective number of consumption channels. Uniform traffic performance is shown for a 4x4x4 configuration with three different messaging overheads: t_{sw} = 50, 100, and 200 cycles. 111

Impact of increasing I-to-P bandwidth (C_m) and consumption channels (C) on the completion time of an all-to-one pattern in a 16x16 and 6x6x6 system. Three different values of C_m = 1, 2 and 8 are considered. 113

Impact of increasing I-to-P bandwidth (C_m) and consumption channels (C) on system performance with uniform traffic. 114

Impact of increasing I-to-P bandwidth (C_m) and consumption channels (C) on system performance with non-uniform traffic. 114

A supply-demand optimization framework for designing and developing scalable architectures under varying processor, interconnection, and packaging technologies. 119

Typical multi-level hierarchy used in packaging a large clustered multiprocessor system. 127

Growth of pincount with board area under peripheral and surface technologies shown for two different relationships between p_p and p_a. The parameters p_p and p_a reflect the pincount that can be supported from a board of unit capacity under periphery and surface pinout technologies, respectively. 132
A system with 48 clusters being placed on 12 boards each with a capacity for holding 4 processor clusters. The overall inter-cluster topology is 4x4x3. The sub-topology placed on each board is 2x2x1.

Offered inter-cluster bisection per processor (B/N) as a function of cluster size and inter-cluster dimensionality while maintaining a channel width $W' = 24$ for a 1024 processor system. Note that for a given pinout technology, the board sizes are not fixed and are changed with cluster size to maintain $W' = 24$.

Offered inter-cluster bisection per processor (B/N) in a system of 1024 processors as a function of board size and different inter-cluster dimensionalities under two different pinout technologies: a) periphery pinout with $p_p = 128$ and b) surface pinout with $p_s = 128$.

Comparing the analytical latency-throughput model with simulations for: a) 4-ary 3-cube cluster-4 system with message length being varied from $F = 2$ to 12 flits and b) 4-ary 3-cube inter-cluster network with cluster size being varied from $c = 1$ to 8 while message length is kept fixed at $F = 8$ flits. Lines shown without points correspond to model predictions.

Comparing the performance, obtained through simulation, of the design-feasible configurations shown in Table 7 to build a system with $N \approx 1024$ processors.

Comparing the performance, obtained through simulation, of the design-feasible configurations shown in Table 9 to build a system with $N \approx 1024$ processors.

Four alternate processor-cluster organizations.

Comparing the performance of U_Mesh broadcast algorithm on four different clustered organizations as the cluster size is increased from $c = 1 \rightarrow 16$. The total number of processors in the system was maintained at $N = 256$ and $N = 1024$ in the left and right columns of plots, respectively. Typical values of $t_s/t_i = 10$, $t_s/t_p = 128$, $t_p/t_p = 2$, $b = 4$, and $m = 128$ flits were assumed.
Example demonstrating the communication steps to perform a broadcast using the U_mesh algorithm in a 9 cluster linear array (1D mesh). The source in figures (a) and (b) is chosen as processor 16 and 0, respectively. Each cluster is assumed to have two injection channels. All communication steps are contention-free. 

Example demonstrating the communication steps to perform a broadcast using the U_mesh algorithm in a 3x3, 2D mesh of clusters. The source in figures (a) and (b) is chosen as processor 16 and 0, respectively. Each cluster is assumed to have two injection channels. All communication steps are contention-free and the broadcast times are similar to those obtained on a 9 cluster linear array.

Example demonstrating the communication steps to perform a broadcast using the novel clus_mesh algorithm in a 9 cluster linear array (1D mesh). Each cluster is assumed to have two injection channels. All communication steps are contention-free.

Example demonstrating the communication steps to perform a broadcast using the recursive clus_mesh algorithm in a 9x9 2D mesh of clusters. Each cluster is assumed to have two injection channels. All communication steps are contention-free.

Comparing the performance of clus_mesh and U_mesh broadcast algorithms with the number of clusters in the system being increased from $N_c = 2 - 1024$. The cluster size is kept fixed at $c = 2$. The three plots depict results for three different values of inter-cluster communication step ($S$) to intra-cluster communication step cost ($F$) of $S/F = 2$, 5 and 10, respectively.

Comparing the performance of Clus_mesh and U_Mesh broadcast algorithms with increasing cluster size. The number of clusters is kept fixed at $N_c = 81$. An inter-cluster communication step of $S = 1000$ cycles and intra-cluster communication step of $F = 100$ cycles are assumed.
Comparing the performance of Clus.Mesh and U.Mesh broadcast algorithms. The impact of increasing cluster size is shown in two different systems with \( N = 256 \) and 1024 processors. An inter-cluster communication step of \( S = 1000 \) cycles and intra-cluster communication step of \( F = 100 \) cycles are assumed.
CHAPTER I

Introduction

In recent years rapid technological advances and increasing demands for computing power have ushered in *massively parallel* machines. Such machines with thousands of powerful processors have potential to solve real compute-challenging applications in a small amount of time. However, the large size of such machines makes their design extremely challenging and the cost prohibitively expensive. Thus, efficient and cost-effective design of such large scale machines is an area of active research in parallel computer architecture.

The broad objectives desired in large scale multiprocessor systems are: 1) *balanced design* such that no particular sub-system constrains the performance leading to under-utilization of other components, 2) *low-cost* to make such systems affordable, and 3) *future scalability* in terms of performance as the number of processors in the system is increased to generate more computing power to handle larger problems. These goals need to be kept in mind while making design decisions for building future systems.
1.1 Processor-cluster based systems

Advancements in VLSI and packaging technologies are now allowing the integration of a small number of processing elements, interconnected through a network, onto a single multi-chip or a board [41, 62] module. Such processor-cluster modules are being increasingly used instead of single processors to build a variety of computing vehicles such as servers, workstations, etc. This is leading to a greater demand for such modules, consequently larger production and lower per-unit cost. Larger systems are being built by interconnecting multiple processor-clusters [73]. Such an approach offers scalability while exploiting the cost-effectiveness and convenience of processor-cluster modules. Similar motivations are also increasingly guiding the development of traditional parallel systems using processor-clusters [8, 23].

Using a cluster as building block instead of a single processor offers a scalable, modular, and cost-effective approach [11, 62] to build massively parallel systems. The processor-clusters are interconnected through a scalable inter-cluster network, e.g. meshes and tori. Thus, processor-cluster based systems are two-level architectures as shown in Fig. 1. The cluster configuration can vary from a simple star connection as in the Cray T3D to a bus interconnection as in the Stanford DASH. Currently the number of processors in a cluster is usually small, ranging from 2 to 4. However, with advancements in VLSI, larger clusters are expected to become common.

In this thesis we introduce a new taxonomy to represent the class of processor-cluster based parallel systems. This is defined as $k$-ary $n$-cube cluster-$c$ systems [10]. Such systems consist of two levels. The lower level consists of $k^n$ clusters of processors
interconnected by a higher level direct $k$-ary $n$-cube inter-cluster network. Each cluster consists of $c$ processors interconnected through an intra-cluster network. The significance of such systems is twofold: processor clusters allow easy exploitation of communication locality and the $k$-ary $n$-cube inter-cluster, which is a generalized grid topology, provides easy scalability for designing a large-scale system.

![Figure 1: Typical example of a processor-cluster based system.](image)

Current processor-clustered organizations connect each cluster to the inter-cluster network via a single interface, referred to as the network interface. This allows a system with a given total number of processors to be interconnected by using fewer network interfaces and network routers, leading to a more cost-effective design [16]. The interface offers a number of channels to and from the network router. These channels (referred to as injection/consumption channels [14]) are used to inject/consume messages to/from other clusters.

Although such a clustered solution promises an elegant and modular approach, there are many open issues that need to be solved to make it feasible and practical. To build a system with a given number of processors, optimal design choices need to be made from the vast number of architectural options in terms of system topology, dimensionality, cluster size, cluster interface design, routing, and support for efficient
communication and synchronization. Currently, no systematic framework exists to design such clustered parallel systems and use them effectively.

1.2 Packaging constraints

It is intuitively clear that systems built with thousands of processors will be huge in size and span a large number of processor boards and board cabinets. Practical packaging constraints like limited board area, pinout densities, and wire delays make the design task more complicated. Some designs which may look impressive on paper, may not translate to real machines, because they violate some packaging constraint. For a more realistic design, it becomes a challenging problem to first characterize the packaging constraints in terms of board sizes, pinout densities, channel widths, interconnect speeds etc., and show how these affect system configurations, i.e., size of various levels, bisection size and channel width that can be offered at each level.

1.3 Inter-processor communication

Messages sent from one processor in one cluster to another processor in another cluster may need to travel over a sequence of channels in the inter-cluster and intra-cluster networks. To ensure correct and guaranteed delivery of such a message, the networks need to support an efficient *routing* scheme. Towards this goal, in this thesis we consider wormhole-routing to be used in direct networks. The wormhole-routing switching technique, often referred to as wormhole routing, is becoming the trend in building parallel systems due to its inherent advantages like low-latency communication and reduced communication hardware overhead [52]. Intel Paragon [42], Cray
T3D [23], Ncube, J-Machine [53], and Stanford DASH/FLASH [35] are representative systems falling into this category.

In the wormhole-switching technique, the flow of a header flit triggers the movement of the remaining flits in a message. If the header flit gets blocked due to unavailability of a network channel, the remaining flits are blocked in the network. Such blocking, besides leading to under-utilization of channels, may block other messages in the network. Wormhole routing on single-level networks is a well-studied problem. However, on multi-level networks like those in clustered architectures, supporting deadlock-free routing delivering low-latency is still an open problem.

1.4 Network-cluster interface

Wormhole-routing was initially proposed with deterministic routing[26, 22]. However, this basic scheme is very restrictive with high probability of message blocking resulting and low network utilization. A number of adaptive routing schemes have been proposed recently by many researchers [18, 20, 48, 49, 59, 67, 68] (on meshes/hypercubes/tori) to reduce such message blocking. Similarly, the use of multiple virtual networks (or lanes) with Virtual Channel Flow Control (VCFC) has been proposed [25] in literature to be used with either deterministic or adaptive routing. With multiple lanes, an active message can bypass a blocked message, leading to lower blocking and better network utilization.

However, the above schemes did not consider the impact of available consumption
capacity on performance. Once a worm reaches a destination node, it needs to be consumed (taken out from network and placed into system message buffers) [52]. Thus, after the header flit of a worm reaches its destination the movement of its subsequent flits in the network depends on the rate at which the message gets consumed. This movement depends on the consumption capacity available at the destination node-network interface, which can be characterized by a) number of consumption channels and b) available local memory bandwidth for message consumption. If the worm gets blocked or slowed down due to limited consumption capacity, then the subsequent flits of the worm remain in the network for additional time, leading to wastage of network bandwidth. Thus, a good network design must evaluate its routing scheme together with the consumption capacity available at a node.

Many parallel applications exhibit non-uniform traffic with hot-spots [29, 47, 76], both in distributed-memory and distributed-shared-memory paradigms. Such traffic leads to more messages being destined towards some nodes in the system. Even with uniform traffic, at higher rates of communication, it is possible for some nodes to receive multiple messages during a small time interval. For both types of traffic, if multiple messages reaching a destination node can not be consumed at full speed due to limited consumption capacity, they get blocked in the network surrounding the destination node. Such blocking can lead to tree-saturation [58] in the network, which may affect the propagation of other hot and cold messages in the system. This has potential to degrade performance in terms of lower throughput and higher latency. Thus, it is challenging to first determine the performance degradation resulting from
limited consumption capacity and to then derive the required consumption capacity per node to take advantage of these schemes.

1.5 Collective communication

Different clusters solving the same problem need to synchronize with each other to ensure the correctness of the result generated. For example, the computation on one cluster may be dependent on the successful termination of a prior computation on another cluster. Supporting fast synchronization primitives and collective communication is a recognized necessity in current and future parallel systems. Collective communication refers to regular communication patterns among processors, for example, a processor broadcasting a value to all other nodes, all nodes exchanging information with each other etc. Since clustered systems provide better locality of communication, it raises challenges on whether the set of collective communication can be implemented on clustered systems with reduced cost.

1.6 Organization of the thesis

Overall, this research is targeted at establishing the processor-cluster based approach as a feasible, dominant, and attractive option to design future large scale parallel systems. Currently some commercial systems like the Cray T3D, Intel Paragon, Convex SPP are designing clustered parallel systems in an ad hoc manner. Though such systems are being built, there exists no systematic framework to guide such designs for building future large scale systems. This research attempts to provide such a framework [12, 8]. Thus, the guidelines developed in this research have immediate
applicability to parallel computer industries to build cost-effective massively parallel systems.

The dissertation is organized into six chapters. Chapter II introduces the processor-cluster based approach and its impact on designing large multiprocessor systems. We present four possible cluster organizations and discuss their respective merits and demerits in terms of scalability and ease of design.

The communication performance of a parallel system is determined by the performance of its sub-components. In Chapter III we analyze two potential bottlenecks affecting communication performance in a parallel system. We present a characterization of parallel systems based on communication performance. Design guidelines to organize and scale systems in a balanced and cost-effective manner are derived. A validation of these guidelines using simulation modeling is also presented.

Chapter IV emphasizes the constraints imposed by packaging technologies on the design of a processor-cluster based system. Similarly, the impact of processor and interconnect technologies on demanded performance is also considered. A new supply-demand framework is presented for multiprocessor design by considering packaging, processor, and interconnect technologies in an integrated manner. For a given set of technological parameters the framework derives the best configuration while considering practical design aspects like maximum board area, maximum available pinout, fixed channel width, and scalability. In order to build a scalable parallel system with a given number of processors, the framework explores the design space of flat $k$-ary $n$-cube topologies and their clustered variations ($k$-ary $n$-cube cluster-$c$) to derive
design-feasible configurations with best system performance.

Chapter V presents some of the issues in inter-processor communication in processor-cluster based systems. A deadlock-free wormhole routing scheme is presented for direct-network based clusters. For more commonly used bus/star-based clusters we propose a network-cluster interface architecture. This architecture proposes using potentially more than one physical consumption/injection channel per cluster. To match the virtual channels in the network typically associated with wormhole routing schemes, we also propose multiple virtual consumption/injection channels on each physical consumption channel. We derive guidelines as to the effective number of such channels required, based on the interplay of various factors including system dimensionality, routing adaptivity, and traffic pattern. Results validating these guidelines using simulation experiments are also provided.

Chapter VI emphasizes that to make processor-clustered systems more attractive, there is a strong need to understand the details about the topology inside the cluster, its memory organization, and the impact of this organization on system performance. We analyze the communication costs for accessing inter-cluster and intra-cluster memories under different cluster organizations. The merits of these organizations are evaluated based on performance of collective communication algorithms (like broadcasting). Results are derived indicating that tightly coupled cluster organizations with shared access to memory offer faster intra-cluster communication. We also demonstrate that such faster intra-cluster access in clustered systems can be exploited to design better collective communication algorithms. Based on this
a new broadcasting algorithm on clustered meshes - *clus_mesh* is proposed which outperforms existing algorithms.

Chapter VII concludes this dissertation and suggests future research directions. The first part emphasizes on summary of results and contributions. The second part indicates future short-term and long-term research directions.
CHAPTER II

Processor-cluster based parallel systems

In this chapter we introduce the processor-cluster based approach and its impact on designing large multiprocessor systems. We present four possible cluster organizations and discuss their respective merits and demerits in terms of scalability and ease of design.

2.1 Advantages from processor-cluster based design

A parallel system consists of a large number of computing nodes interconnected through a network. Typical topologies for the interconnection are meshes/tori, hypercubes, multi-stage interconnection networks, and fat-trees. For inter-processor communication, each processor is connected to a router. Messages are routed along these routers on the interconnection network. An example system with 16 nodes interconnected by a 4x4 mesh network is shown in Fig. 2(a). It can be observed that in such an organization the required number of network routers, denoted by $R$, is equal to the number of processors in the system i.e., $R = 16$. In the same system the number of channels in the network can be derived as 48. The number of parallel wires in a channel determines its width, denoted as $W$. The amount of data that can
be transferred across it directly depends on $W$. Let $N_W$ denote the total number of wires required to connect the channels in a given network. For the system shown in Fig. 2(a), $N_W = 48W$.

![Figure 2: Examples of parallel systems using computing nodes based on a) single processors and b) processor-clusters of size 4. The system in (b) interconnects the same number of processors on a smaller network.](image)

With advancements in VLSI and packaging technologies, computing nodes having more than one processor on a single multi-chip module or processor-board are becoming increasingly viable. Many current parallel systems like the CRAY T3D [23], Intel Paragon, Stanford DASH [35], and Sequent NUMA-Q [73] are using processor-cluster based design to build cost-effective systems. Figure 2(b) shows an example system with processor-clusters each having 4 processors. This configuration uses 4 processor-clusters connected by a 2x2 mesh network, thus supporting a total of 16 processors.

In systems built using processor-clusters, let $c$ denote the number of processors available in a cluster, also referred to as the size of a cluster. The maximum value of $c$ depends on the level of integration available inside clusters. The network interconnecting processor-clusters is referred to as inter-cluster network. For uniformity
of representation, a single processor based computing node is treated as a processor-cluster of size one. For designing a system with \( N \) processors with processor-clusters of size \( c \), \( N/c \) clusters are necessary.

Processors inside a cluster communicate with each other using intra-cluster network (\textit{intra-cluster communication}) provided as part of the processor-cluster module. This network connecting the processors inside a cluster can be a bus, crossbar, star, etc. as discussed in [55]. Although the choice of the cluster topology is an important design issue, in this study we emphasize more on how and when processor-clusters can be used in designing \textit{balanced} and larger cost-effective systems. While developing design guidelines, we assume that a processor-cluster technology offering up to \( c = 8 \) as being available.

Inter-processor communication across clusters is achieved along the inter-cluster network. Each processor-cluster is connected to a router as shown in Fig. 2(b). A processor injects a message into its corresponding router which is routed along intermediate routers to the destination router where it is consumed by the destination processor. The number of inter-cluster routers required is the same as the number of clusters \( (R = N/c) \). For example, in the system shown in Fig. 2(a), with \( N = 16 \) and \( c = 1 \), we have \( R = 16 \). Similarly, in the system shown in Fig. 2(b), with \( N = 16 \) and \( c = 4 \), we require fewer routers \( (R = 4) \). Thus, such a processor-cluster based system with larger clusters requires lesser inter-cluster routers. Similarly, it can be observed that a system with \( c = 4 \) requires lesser \( N_W = 8W \), compared to \( N_W = 48W \) in a system with \( c = 1 \). This leads us to the intuition that processor-cluster based
approach to designing parallel systems has potential in offering lower interconnection costs. For a fair cost comparison, one should also keep in mind the increased routing hardware inside clusters for intra-cluster communication. However, with higher level of integration inside clusters, we expect the intra-cluster routing capabilities being offered at comparatively much lower cost [41, 62], leading to an overall lowering in interconnection cost with clustering.

To match the increasing demand on computing bandwidth, parallel systems need to be scaled to larger configurations supporting more processing nodes. The default approach to such scaling has been to increase the network size with more links and routers to interconnect more processors. Figure 3(a) illustrates such an approach to scale a 4x4 mesh interconnected 16 processor base system to a larger 8x8 mesh interconnected 64 processor system. This default approach maintains the cluster size at one (c = 1). The channel width (W) is assumed to be maintained in this example. It can be observed that the scaled system requires R = 64 and NW = 224W. An alternative approach to scaling, using larger processor-clusters is illustrated in Fig. 3(b). This design maintains the interconnection network to R = 16 and NW = 48W while using larger clusters of size four to achieve similar scaling. This once again illustrates the potential of processor-cluster based design in offering lower interconnection costs.

2.2 k-ary n-cube cluster-c organization

In this thesis we introduce a new k-ary n-cube cluster-c organization [10, 11, 55] to capture the upcoming trend in building cluster-based parallel systems and to analyze
the various design issues involved. In this organization, the lower level consists of \( k^n \) processor clusters. These clusters are interconnected by a higher level direct \( k \)-ary \( n \)-cube network, also referred to as inter-cluster network or internet in this thesis. Systems with buses as the inter-cluster network are not very scalable. A MIN-based inter-cluster network scales well but the locality of communication between adjacent clusters is not exploited. On the other hand, \( k \)-ary \( n \)-cube direct network offers both these features, making it a good choice for the inter-cluster network. Each cluster consists of \( c \) processors leading to a total of \( N = (k^n \cdot c) \) processors in the system. This interconnection achieves two main design objectives: a) direct network-based internet providing easy scalability and b) processor clusters providing the convenience of packaging modularity and potential for better exploitation of communication locality. Figure 4 shows the overall configuration of such a system.
2.2.1 Architectural Alternatives and Choices

To build an $N$-processor system, there are a vast number of possible alternatives with clustered configurations. The degrees of freedom are: number of processors (size) in each cluster and topologies of the two levels (inter-cluster and intra-cluster). Let us consider designing a system with $N = 1024$ processors.

**Size of levels:** This system can be designed with 64 clusters of 16 processors each, 16 clusters of 64 processors each, and so on. Note that for a given system size, fixing the size of one level automatically determines the size of the other level.

**Topology choices:** Having fixed the size of each level, there is still freedom to vary the topology in each level. For example, in a system with 64 clusters with 16 processors each, the topologies can be: 4-ary 3-cube internet with bus-based clusters, 8-ary 2-cube internet with MIN-based clusters, and so on. Our objective is to select the configuration which, a) satisfies packaging constraints, b) meets desired performance of latency and throughput, and c) is easily scalable to larger sizes.

It is easy to observe that the flat $k$-ary $n$-cube systems can be derived as a special
case of the $k$-ary $n$-cube cluster-$c$ family by choosing cluster size $c = 1$. Since inter-cluster interconnections are more expensive than intra-cluster interconnections [62], we primarily emphasize on internet and cluster size.

### 2.3 Intra-cluster organizations

The interconnection within a cluster (also referred to as intra-cluster network or intranet) can be chosen as bus/MIN/star network/direct network as shown in Fig. 5. Each cluster is connected to the rest of the system through a cluster interface. The main task of the cluster interface is to handle the volume of communication to/from the cluster. Other functionalities may be added to the cluster interface to efficiently implement various communication, synchronization, and cache-coherence operations to enhance overall system performance [35, 55].

![Figure 5: Four possible cluster interconnections under $k$-ary $n$-cube cluster-$c$ organization.](image)

The memory in such systems is distributed physically across the clusters. Organization of memory within a cluster is currently left as an open choice depending on the size of cluster and its configuration. In Chapter VI we derive guidelines that tightly coupled cluster organizations with shared access to memory offer faster intra-cluster
communication leading to better performance. The exact nature of this distribution is not critical to the design and analysis presented in the next few chapters. Therefore, without loss of generality we assume the cluster memory to be distributed uniformly amongst the processors in the cluster.

Let us analyze some of the architectural issues for the four candidate cluster interconnections:

2.3.1 Direct-network based clusters

In such configurations, processors in the cluster are connected by a direct point-to-point network. Such networks provide scalability and exploitation of communication locality. However, it is generally more expensive in terms of routing hardware. Broadcasting and multicasting patterns need to be executed on these systems as a sequence of point-to-point messages. The popular $k$-ary $n$-cube network can be adopted within the cluster. To distinguish the cluster $k$-ary $n$-cube topology from the inter-cluster $k$-ary $n$-cube topology we refer to $k$-ary $n$-cube inside the cluster as $k1$-ary $n1$-cube. We represent the overall $k$-ary $n$-cube cluster-c architecture for such direct-network based clusters as $k$-ary $n$-cube “$k1$-ary $n1$-cube” and refer to it in short as $(k,n,k1,n1)$.

2.3.2 MIN based clusters

In a MIN-based organization, cluster processors are interconnected through a wrap-around MIN. There are additional input and output lines from the MIN to the CI router switch. MINs are scalable and allow the flexibility of having larger cluster sizes. In conventional MINs all intra-cluster messages have to travel through all
stages leading to poor exploitation of intra-cluster communication locality. However, this problem can be easily solved by using bidirectional links with loopbacks as used in the IBM SP1/SP2 system. Efficient implementation of broadcast/multicast patterns in a deadlock-free manner with wormhole-routing is somewhat difficult in such organizations.

2.3.3 Bus-based clusters

In a bus-based scheme, $c$ cluster processors are connected to an intra-cluster bus-system (single/multiple buses). The router has a set of input and output links connected to this bus-system. For clusters with small number of processors this leads to an easy shared memory implementation within the cluster. However, this organization is not scalable to large clusters because of limitation in available bus-bandwidth. Broadcasting communication is supported naturally in this organization using a snoopy bus protocol.

2.3.4 Star-based clusters

In a star-based scheme, $c$ processors are connected directly to the CI. Therefore all inter- and intra-cluster traffic need to go through the CI. Such an organization has been used in the Cray-T3D cluster. The advantage of such a centralized cluster organization is that the CI can mimic a cluster bus and also implement intra-cluster broadcast/multicast communication easily. However, the size of the cluster is limited by the available connectivity and bandwidth to/from the CI.
2.4 Summary

In this chapter we have introduced the processor-cluster based approach and its impact on designing large multiprocessor systems. We have presented four possible cluster organizations and discussed their respective merits and demerits in terms of scalability and ease of design. In the next chapter we derive design guidelines to organize and scale processor-cluster based systems in a balanced and cost-effective manner.
CHAPTER III

Design space of processor-cluster based systems

The performance of a multiprocessor system is often limited by the bisection bandwidth of its interconnection network [43, 75]. In the recent past, systems have been built using advanced interconnection networks with 16 and 32 bit channels and channel cycle times of 6.6-10.0 ns. The offered bandwidth for inter-processor communication in such systems is very high. This coupled with the recent trends in routing mechanisms like fast packet switching, virtual-cut through, and wormhole routing has led to systems offering very small point-to-point network delay in the range of less than a microsecond.

The increasing availability of high bandwidth interconnection networks raises the following question: Can the offered higher bandwidth be effectively exploited to always deliver higher system performance? In point-to-point communication, a message is associated with processing overheads from packetization, checksum computing, packet-ordering, and memory copying. Such overheads, usually in the range of few to tens of microseconds, can severely limit the rate at which messages can be sent or received from a processor [44]. This can lead to much of the communication bandwidth offered by the expensive fast interconnect remaining unutilized. Several
research studies are ongoing [45, 32] for designing messaging protocols and hardware to reduce such overheads. However, even with efficient protocols these overheads cannot be eliminated. Even though the overheads get lowered, these will continue to be reasonably high compared to network speeds. For example, on NCUBE-2 messaging start-up overhead is around 154 microseconds and network channel transmission time per-byte is only 0.4 microseconds [30]. This leads to a ratio of message overhead to per-byte transmission delay as 385. In contemporary systems like the Cray T3D, message overhead is 2.0 microseconds and per-byte network transmission time is 0.0033 microseconds, leading to a ratio of 600. Thus, messaging overheads continue to be a dominant factor for poor utilization of network resources in current generation parallel systems.

Advancements in VLSI are making it cost-effective to use processor-clusters as building blocks in the development of traditional parallel systems [8, 23]. Previous research on using processor-clusters for building large parallel systems have mainly focused on proposing and proving different interconnection topologies [28, 55, 60] and studying the design problem under very realistic packaging constraints [11, 62]. Such processor clustering also demonstrates potential to alleviate the network under-utilization problem by allowing more number of processors to use a given set of network resources. However, no formal study is available in the literature which shows the benefits of processor clustering towards better utilization of network resources.

In this chapter we take on such a challenge and demonstrate how to exploit
the emerging cluster and interconnect technologies in designing balanced and cost-effective systems. Our solutions span along two directions. Systems with high messaging overheads can have poor network utilization. In this case we show that processor clustering can be used to share network resources. This leads to effective exploitation of the communication bandwidth of the interconnection network resulting in larger as well as balanced and more cost-effective systems. Along the second direction, we show that the advantages of clustering can also be extended to design large systems when messaging overheads are low. In this case, building larger systems is proposed by using larger clusters while maintaining the interconnection network topology. The performance is maintained by using proportionately wider network channels. We demonstrate that as long as cluster size and channel width can grow under technological constraints, such an approach offers solutions to design cost-effective systems while maintaining performance.

This chapter is organized as follows. In Sec. 3.1 we present the impact of processor-clustering in designing large multiprocessor systems. In Sec. 3.2 we discuss two potential bottlenecks affecting the communication performance in a parallel system. In Sec. 3.3 we present a characterization of parallel systems based on communication performance. Design guidelines to organize and scale systems in a balanced and cost-effective manner are presented and verified in Secs. 3.4 and 3.5. Application benchmark based evaluation of results are presented in Sec. 3.6.
3.1 Enhanced design space with processor-clustering

The illustrations in the last chapter emphasize the potential of processor-clustering in designing parallel systems. Besides developments in processor-clustering, recent advancements in interconnection technology are also allowing wider and faster network channels. Such channels are leading to higher network bandwidth being offered. However, high bandwidth networks are expensive and this cost increases with offered bandwidth. Further, as we demonstrate in the next section, having higher network bandwidth does not necessarily translate to higher performance. Thus, for a system architect designing a parallel system, it is imperative to determine network configurations offering the best achievable performance at minimum cost. In this section we present the spectrum of options to build parallel systems under the flexibility of selecting a given inter-cluster size, cluster size, and channel width from a wide range of available choices.

In the traditional approach of using single processors for parallel system design, larger configurations necessarily lead to larger interconnects requiring more routers and interconnection wires. With advancements in processor-cluster technology such approach is undergoing a change. Larger processor-clusters are allowing a wider choice of design configurations. Such extensions to design options can be summarized as a design space graph (DSG) as shown in Fig. 6. The vertical axis of the DSG indicates the number of inter-cluster routers ($R$) in a given system. The horizontal axis in Fig. 6 represents the upcoming flexibility of using larger processor-clusters ($c$)
in design. The $R$ routers are interconnected by a given inter-cluster network topology. For sake of illustration, and to match with the previous examples, we consider a 2D mesh inter-cluster topology. However, the approach is applicable to any other topology. On the third axis we denote the impact of advancing channel bandwidth technology. Increase in channel bandwidth can be due to increased channel speed and/or larger channel width technology. To represent the impact of both we require two separate technological dimensions on the DSG. However, in most analysis we are interested in the overall impact on channel bandwidth leading to higher interconnection performance. Thus, we study increase in channel bandwidth due to increase in only channel width ($W$). We represent $W$ in bytes to reflect a trend of designing byte-multiple channel width to maintain an integral relationship with processor and memory modules [8].

The plane formed by the $R$ and $c$ axes is referred to as the $R-c$ plane. Each point on the $R-c$ plane, denoted by tuple $(R, c)$, represents a parallel system configuration with $(N = Rc)$ processors. For any given $(R, c)$ tuple we can have configurations with different channel bandwidth. These are depicted in the DSG as an arrow out of the $R-c$ plane parallel to the $W$ axis and represented as triple $(R, c, W)$. For example, in Fig. 6, the configuration with $R = 4$, $c = 1$ and $W = 1$ byte, representing a 2x2 flat configuration, is shown as tuple $(4,1,1)$. Similar configurations with higher channel bandwidth are represented as: $(4,1,2)$, $(4,1,4)$, and $(4,1,8)$.

Let us illustrate how the example systems considered earlier in Figs. 2 and 3 fit into the DSG framework. Let the channel width ($W$) considered in these examples
be 1 byte. Thus, these configurations lie on the $R$-$c$ plane for $W = 1$. The 4x4 flat configuration in Fig. 2(a) with $c = 1$ and $R = 16$, gets represented as tuple $(16,1,1)$. The corresponding 2D inter-cluster network (4x4) is shown underlined. The default approach of scaling four times with a larger 8x8 interconnection with 64 routers, while maintaining cluster size to one ($c = 1$), is depicted in Fig. 6 as tuple $(64,1,1)$. A vertical arrow from $(16,1,1)$ to $(64,1,1)$ represents such an approach. Similar scaling can be alternatively achieved by using larger processor-clusters, e.g. ($c = 4$). This approach, applied to tuple $(16,1,1)$ leads to tuple $(16,4,1)$ having 16 clusters (64 processors) connected by a 4x4 inter-cluster network. In this second approach the number of routers is maintained at $R = 16$.

![Diagram showing enhanced design space for parallel system configurations using larger processor-clusters.](image)

**Figure 6:** Depicting the enhanced design space for parallel system configurations using larger processor-clusters. For designing systems with more processors, traditional approach only considered varying the interconnect size leading to higher $R$. Using processor-clusters adds a second dimension of variability, the cluster size ($c$). The third dimension, $W$, represents the added flexibility of using higher bandwidth network channels.

In general case $R$, $c$, and $W$ can be varied leading to new configurations. For
example, the (16,1,1) configuration leads to a (4,4,1) configuration by reducing $R$ four times while compensating for this reduction by choosing $c = 4$. This example demonstrates the approach of clustering to design systems having same number of processors with a smaller inter-cluster network and fewer routers. Let us introduce a set of definitions with respect to the DSG to be used in the remaining part of the chapter:

**Definition 1** The set of configurations $\left(\frac{N}{c}, c, W\right)$ having same total number of processors, $N$, for different values of $c$ and $W$ lie in the $R - c - W$ space on a plane, defined as the iso-processor plane for $N$. For a given $W = W'$, the subset of these configurations $\left(\frac{N}{c}, c, W'\right)$ lie on the $R - c$ plane for $W = W'$ as a line defined as iso-processor line for $N$.

Let us consider a given $(R, c, W)$ configuration reflecting contemporary trends in designing parallel systems. Let us assume the inter-cluster topology as an $n$-dimensional mesh. Thus, the $R$ routers in the $(R, c, W)$ configuration are interconnected by a $k_1 \times k_2 \times \cdots \times k_n$ mesh, where $\prod_{i=1}^{n} k_i = R$. In this case a more detailed notation (instead of $(R, c, W')$) is useful to represent a configuration:

**Definition 2** A processor-cluster based system with cluster size $c$ and an $n$-dimensional inter-cluster network of size $R = k_1 \times \cdots \times k_n$ with channels of width $W$ bytes is represented as $(k_1 \times \cdots \times k_n, c, W)$.

There are many different ways to factorize $R$ into $n$ such factors. Factorization with nearly equal values for all $k'$ is preferred as it leads to smaller network organization
and benefits from packaging compactness [8]. With \((k_1 = \cdots = k_n = k)\) the notation \((k_1 \times \cdots \times k_n, c, W)\) can be further shortened to \((k^n, c, W)\). For example, the system configuration in Fig. 2(a) can be represented as \((4 \times 4, 1, 2)\) or \((4^2, 1, 2)\).

3.2 Characterizing communication performance and bottlenecks

The communication performance of a parallel system is determined by the performance of its sub-components. Figure 7 shows these subcomponents as different stages in the process of a message transfer from a source processor to a destination processor. Typically the message gets initiated at the sender through a call to a communication library function. The message is then prepared and injected into the network through the network interface. The network interface provides to and fro connectivity between a processor (or cluster) and its network router. The message traverses across the network to the destination processor, where it is consumed through the network interface. The destination processor acquires the message through a call to another communication function. The overall performance of the system is clearly determined by the sub-component with lowest bandwidth.

Figure 7: The process of a message transfer from a source processor to a destination processor showing the various sub-components.
The communication performance of a parallel system is often characterized by its average message latency versus sustained message throughput under uniform traffic. Such latency-throughput characteristic is used to derive the \textit{maximum sustained throughput} (point close to saturation) of a system. We represent the maximum sustained throughput as $\lambda$ messages/$\mu$sec. Uniform traffic is considered here as it is more representative of general traffic in a parallel system [62].

In the following subsections we use $\lambda$ as a metric to analyze the communication performance of a parallel system. We first consider the overheads of messaging. In [16] it was demonstrated that under uniform traffic, the network bandwidth is a stronger restriction on performance than network interface. We assume the cluster module technology offering reasonable bandwidth at the interface for it not to be a bottleneck. Thus, we focus only on the network bandwidth and messaging overheads. The derived observations are validated in later sections using simulation and experimental studies with uniform traffic as well as benchmark applications.

3.2.1 Messaging Overheads

The overheads of messaging can be divided into two parts. The first part involves a software latency which includes the overhead of procedure calls and possibly, memory copying from user space to system space. We represent such overhead as $t_s$ $\mu$sec/message. Based on the study presented by Dongarra and Dunigan in [30], representative values of $t_s$ on current machines are shown in Table 1. On most systems $t_s$ can be observed to vary from a few hundreds of cycles up to thousands of cycles. Even recent machines like the IBM SP2 and Convex SPP have significantly high $t_s$. 
The value of $t_s$ on a given system can vary with the communication library. Libraries offering higher degree of functionality can have high $t_s$. For example, on the Cray T3D communication with PVM requires $t_s = 21\mu\text{sec/message}$. Similar communication with Shared Memory Library (sm) requires a much lower $t_s = 2\mu\text{sec/message}$. Although the message startup overhead depends on the communication library, we observe that even for faster libraries the overhead is significantly high in contemporary systems. This trend is expected to continue in the near future.

Table 1: Message Startup Overhead ($t_s$) on Some Recent Systems.

<table>
<thead>
<tr>
<th>Machine</th>
<th>OS</th>
<th>$t_s$ (\mu\text{sec/message})</th>
<th>Clock Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCUBE 2</td>
<td>Vertex 2.0</td>
<td>154</td>
<td>3080</td>
</tr>
<tr>
<td>Intel Paragon</td>
<td>OSF 1.0.4</td>
<td>29</td>
<td>1450</td>
</tr>
<tr>
<td>IBM SP1</td>
<td>MPL</td>
<td>270</td>
<td>16875</td>
</tr>
<tr>
<td>IBM SP2</td>
<td>MPI</td>
<td>35</td>
<td>2310</td>
</tr>
<tr>
<td>Cray T3D (sm)</td>
<td>MAX 1.2.0.2</td>
<td>2</td>
<td>450</td>
</tr>
<tr>
<td>Cray T3D (PVM)</td>
<td></td>
<td>21</td>
<td>3150</td>
</tr>
<tr>
<td>Convex SPP1000 (sm)</td>
<td>SPP-UX 3.0.4.1</td>
<td>11</td>
<td>1080</td>
</tr>
<tr>
<td>Convex SPP1000 (PVM)</td>
<td></td>
<td>76</td>
<td>7600</td>
</tr>
</tbody>
</table>

The second part of messaging overhead is incurred in dividing the message into standard sized packets, accessing memory, forming packet headers, checksum computing, and possibly waiting for acknowledgments to ensure reliable delivery. Such overhead is typically proportional to the length of the message. Let $t_p \mu\text{sec/byte}$ denote overhead incurred per byte of a message. The parameter $t_p$ determines the asymptotic message injection bandwidth offered by a communication library on a
given system. Table 2 provides representative values of $t_p$ obtained on different machines and messaging platforms [30]. It can be observed that on most systems the asymptotic messaging bandwidth achieved is significantly lower than the theoretical channel maximum, indicating relatively high messaging overheads on such systems. As with $t_s$, the parameter $t_p$ also depends on the underlying communication mechanism/protocol.

Table 2: Achievable Asymptotic Messaging Bandwidth Per Processor and Transmission Overhead Per Byte, $t_p$.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Asymptotic Bandwidth (MB/sec)</th>
<th>$t_p$ (μsec/byte)</th>
<th>Theoretical Link Bandwidth (MB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCUBE 2</td>
<td>1.7</td>
<td>0.6</td>
<td>2.5</td>
</tr>
<tr>
<td>Intel Paragon</td>
<td>154</td>
<td>0.0065</td>
<td>175</td>
</tr>
<tr>
<td>IBM SP1</td>
<td>7</td>
<td>0.143</td>
<td>40</td>
</tr>
<tr>
<td>IBM SP2</td>
<td>35</td>
<td>0.028</td>
<td>40</td>
</tr>
<tr>
<td>Cray T3D (sm)</td>
<td>128</td>
<td>0.008</td>
<td>300</td>
</tr>
<tr>
<td>Cray T3D (PVM)</td>
<td>27</td>
<td>0.037</td>
<td>300</td>
</tr>
<tr>
<td>Convex SPP1000 (sm)</td>
<td>59</td>
<td>0.017</td>
<td>250</td>
</tr>
<tr>
<td>Convex SPP1000 (PVM)</td>
<td>11</td>
<td>0.09</td>
<td>250</td>
</tr>
</tbody>
</table>

Based on the above discussion, the overheads for a message of average length $L$, denoted by $T$, is the sum of the overheads incurred in the two parts:

$$T = t_s + t_p L \, (\mu\text{sec}).$$

(3.1)

This limits the maximum message handling rate from a given processor, denoted by $\lambda_{sw}$ messages/μsec. Assuming a processor cannot initiate the next message before the
previous message is fully transmitted into the network, \( \lambda_{sw} \) is obtained as \( 1/T \). Let us consider a more optimistic non-blocking communication model in which a processor continues program execution after the first part of messaging overhead \( t_s \) is incurred. This allows the initiation of another message by the processor while the previous one is still in its second part. It assumes the existence of hardware capability to inject a message independent of processor support; capability that is commonly available in current parallel systems. Thus, the maximum rate of message handling in this model is only limited by the larger overhead and is obtained as:

\[
\lambda_{sw} = \frac{1}{\max(t_s, t_p L)} \text{ messages/\( \mu \)sec.} \tag{3.2}
\]

The above expression leads us to the following observation:

**Observation 1** The value of the parameter \( \lambda_{sw} \) is independent of the system configuration. Thus, different system configurations with similar messaging overheads offer same \( \lambda_{sw} \).

### 3.2.2 Network bisection

Besides messaging overheads, the other main determinant of communication performance is the network bisection. The bisection size of a network is the minimum number of network channels to be cut so as to divide the network into roughly two equal halves [24]. Under uniform traffic, on the average half of the generated messages in the system need to cross the inter-cluster network bisection. Let the maximum rate of injection of messages per processor imposed by the bisection size, assuming no messaging overhead \( t_s = t_p = 0.0 \), be denoted by \( \lambda_{nw} \) messages/\( \mu \)sec.
In [3] Agrawal presented an analytical model of contention in a wormhole-routed k-ary n-cube system. In [8] we extended this model to clustered systems with \((k^n, c, W)\) topology. The expression for such average message latency \(T\) versus throughput per processor \(\lambda\) as derived in detail in Chapter 5.5 is:

\[
T = \left[1 + \frac{\lambda t_c c F^2}{(1 - \lambda t_c c F d)} \frac{(d - 1)}{d} \left(1 + \frac{1}{n}\right)\right] nd + F. \tag{3.3}
\]

where, \(T\) denotes the average message latency expressed in network cycles. The parameter \(\lambda\) is the message injection rate by a processor in terms of messages/μsec and \(t_c\) denotes the network cycle time in microseconds. The parameter \(d\) represents the average number of hops taken by a message in a dimension. For bidirectional channels this value is \((k - 1)/2\). The message size expressed in flits is \(F = L/W\), where \(L\) is the message length in bits and \(W\) the inter-cluster channel width. From the above equation, it can be observed that \(nd + F\) cycles is the minimum latency suffered by a message. This happens at a very small value of message injection rate.

Equation 5.12 can be rewritten to express the throughput per processor as a function of \(T\) and other parameters as:

\[
\lambda = \frac{(T - F - nd)d}{cFdt_c((n + 1)(d - 1)F + (T - F - nd)d)} \text{ messages/μsec} \tag{3.4}
\]

We identified a reasonable maximal operational point on this latency versus throughput plot, by assuming the maximum average latency to be bounded by a multiple of the minimal average latency \(T\). Let \(f\) denote such a multiple. Typical values of \(f\) are chosen between 2 and 4. Without loss of generality, in the following discussion we choose \(f\) to be 4. Let \(\lambda_{nw}\) denote the throughput at the point maximum average
latex is attained. This point is identified using Eqn. 3.4 as:

\[
\lambda_{nw} = \frac{3(F + nd)d}{cF dt_c((n + 1)(d - 1)F + 3(F + nd)d)} \text{ messages/\mu sec} 
= \frac{1}{cF dt_c(\frac{(n+1)(d-1)}{3(F+nd)d} + 1)} \text{ messages/\mu sec} 
\]

(3.5) 
(3.6) 
(3.7)

For reasonably large values of \( F \) and \( d \), and using \((F = L/W)\) and \( d = (k - 1)/2 \) the above expression can be further simplified to:

\[
\lambda_{nw} = \frac{2W}{cL(k - 1)t_c(1 + (n + 1)/3)} \text{ messages/\mu sec}. 
\]

(3.8)

Equation 3.8 leads us to the following observations:

**Observation 2** Given a \((k^n,c,W)\) system, the impact of varying selected system parameters on the maximum throughput supportable per processor by network bisection \((\lambda_{nw})\) can be summarized as follows:

<table>
<thead>
<tr>
<th>System Parameters</th>
<th>Variation</th>
<th>Effect on ( \lambda_{nw} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>↑</td>
<td>↓</td>
</tr>
<tr>
<td>( c )</td>
<td>↑</td>
<td>↓</td>
</tr>
<tr>
<td>( W )</td>
<td>↑</td>
<td>↑</td>
</tr>
</tbody>
</table>

An entry in the table: ↑ or ↓ corresponds to the respective system parameter being increased or decreased, respectively, and its resultant impact on \( \lambda_{nw} \). Similarly, relationships obtained by reversing the sense of all arrows in the above table also hold true. It can be noted that the value of \( \lambda_{nw} \) may be maintained by increasing \( c \) while decreasing \( k \), or vice-versa. Similarly, \( \lambda_{nw} \) can also be maintained by increasing \( c \) and/or \( k \) while compensating the resultant loss in \( \lambda_{nw} \) by increasing \( W \).
3.2.3 Determining communication performance - interplay of above constraints

Let us derive the overall impact of messaging overhead ($\lambda_{sw}$) and network bisection ($\lambda_{nw}$) on the communication performance ($\lambda$) of a parallel system. The maximum achievable throughput per processor in a system is determined by the smaller of the two parameters, leading to:

$$\lambda = \min(\lambda_{nw}, \lambda_{sw}).$$  \hspace{1cm} (3.9)

As discussed earlier, in this study we do not propose new messaging protocols and hardware to reduce $\lambda_{sw}$. Given a messaging overhead ($\lambda_{sw}$) we demonstrate on how to exploit emerging cluster and interconnect technologies for achieving the best performance of $\lambda = \lambda_{sw}$ with more cost-effective designs. We used simulation modeling to verify the interplay of $\lambda_{nw}$ and $\lambda_{sw}$ on system performance, as predicted by Eqn. 3.9. We describe the experimental set-up briefly before proceeding to discuss the results. Design guidelines presented in later sections are also similarly validated using experimentation on a variety of system configurations using different traffic patterns and applications.

Experimentation and simulation setup

Real systems have fixed cluster size, inter-cluster channel width, and other architectural parameters. Thus, to study the variation of such parameters we conducted experiments on the Ohio State clustered architecture simulation testbed - ClusterSim. ClusterSim[7] is an event-driven simulation testbed developed using CSIM [66], offering flexibility to analyze the impact of varying system parameters like cluster size,
topology, channel width, network channel cycle time, and messaging overheads on system performance. ClusterSim accurately models flit-level transfer corresponding to wormhole/cut-through routing as in real systems like the Paragon and Cray T3D. The following contemporary timing parameters were assumed for the network channels and routers: network channel cycle time of 5ns, routing time at a node of 15ns, and switching time across a router crossbar of 5ns. Network latency vs. throughput result plots were derived with uniform traffic for different system configurations. In this section we only present simulation results to validate Eqn. 3.9. Further results from the simulation testbed and experimentation on the Cray-T3D to validate overall design guidelines are presented in Sec. 3.6.

Validation of communication performance predictions

Figure 8 shows network latency versus throughput plots for a (4x4,1,2) system obtained with different values of $t_s = 0.25, 1.0, 5.0, 10.0 \mu\text{sec}/\text{message}$ and $t_p = 0.01\mu\text{sec}/\text{byte}$. The message length was fixed at $L = 256$ bits. For such value of $L$ it can be shown using Eqn. 3.2 that $\lambda_{sw} \approx 1/t_s$. Thus the above values of $t_s$ correspond to $\lambda_{sw} = 4.0, 1.0, 0.2, \text{and } 0.1 \text{ messages/}\mu\text{sec}$, respectively. The value of $\lambda_{nw}$ in this system, derived as the maximum achievable network throughput corresponding to $t_s \to 0.0\mu\text{sec}$, was 2.5 messages/\mu\text{sec}. From Fig. 8(a) it can be observed that for a low value of $t_s = 0.25$, corresponding to $\lambda_{sw} = 4.0$, the maximum throughput in the system is determined as $\lambda = \lambda_{nw} = 2.5 \text{ messages/}\mu\text{sec}$. However, for a higher value of $t_s = 1.0$, corresponding to $\lambda_{sw} = 1.0 \text{ messages/}\mu\text{sec}$, the maximum performance in the system is throttled at $\lambda = \lambda_{sw} = 1.0$, thus not achieving the full network potential of
\( \lambda_{nw} = 2.5 \text{ messages/\mu sec} \). Similar trends were observed with other higher values of \( t_s = 5.0 \) and 10.0. With a (8x8,1,2) system the maximal achievable network throughput \( \lambda_{nw} \) is only 1.0 messages/\mu sec, corresponding to \( t_s \to 0.0 \). As shown in Fig. 8(b), in such a system having \( t_s = 1.0 \) or lower, corresponding to \( \lambda_{sw} = 1.0 \text{ messages/\mu sec} \) or higher, allows the maximal possible system performance \( (\lambda = \lambda_{nw} = 1.0) \) to be achieved. These observations validate Eqn. 3.9.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure8}
\caption{Latency vs. throughput plots with different message overhead costs \( t_s \). Results are shown for two different system a) (4x4,1,2) and b) (8x8,1,2). The maximum throughput, corresponding to \( t_s \to 0.0 \text{ \mu sec} \), depicts the value of \( \lambda_{nw} = 2.5 \) and \( \lambda_{nw} = 1.0 \text{ messages/\mu sec} \), respectively, in the two systems. Interplay between \( \lambda_{nw} \) and \( \lambda_{sw} \) leads to observed value of \( \lambda \).}
\end{figure}

3.3 Characterizing parallel systems based on communication performance

In systems having \( \lambda_{sw} > \lambda_{nw} \), the network bisection is used effectively \( (\lambda = \lambda_{nw}) \) while the message processing capacity at processors remains under-utilized \( (\lambda < \lambda_{sw}) \). Thus, such system configurations are network bisection constrained. Similarly, in
systems having high messaging overheads with $\lambda_{sw} < \lambda_{nw}$, the throughput is throttled by messaging overheads ($\lambda = \lambda_{sw}$) causing the network resources to remain underutilized ($\lambda < \lambda_{nw}$). Such system configurations are therefore messaging overhead constrained. Finally, systems with ($\lambda_{sw} \approx \lambda_{nw}$) have balanced message processing capacity and network bisection. In such a system configuration both resources are effectively utilized. This leads us to the following characterization of parallel systems:

**Definition 3** A parallel system can be classified as:

- **network bisection constrained (NC),** if ($\lambda_{nw} < \lambda_{sw}$)
- **messaging overhead constrained (MC),** if ($\lambda_{sw} < \lambda_{nw}$)
- **balanced (B),** if ($\lambda_{sw} \approx \lambda_{nw}$).

In the DSG in Fig. 6 we analyzed the design space of configurations in building parallel systems. Let the parameter $\lambda_{nw}$ for a $(k^n, c, W)$ configuration be derived based on Eqn. 3.8. For a given set of messaging overheads, $t_s$ and $t_p$, the value of $\lambda_{sw}$ is similarly derived based on Eqn. 3.2. As derived in Observation 1, the value of $\lambda_{sw}$ remains same across all configurations. By comparing the values of $\lambda_{nw}$ and $\lambda_{sw}$, a configuration can be classified as NC, MC, or B. The following theorem analyzes a property on such classification:

**Theorem 1** Given a balanced (B) configuration $(R, c, W)$, another configuration having the same network $(R)$ and channel width $(W)$ may be characterized as follows:
1. configuration with smaller cluster size, $c' < c$, is message overhead constrained (MC)

2. configuration with larger cluster size, $c' > c$, is network bisection constrained (NC).

Proof: Given a balanced $(R, c, W)$ configuration, it clearly satisfies ($\lambda_{sw} \approx \lambda_{nw}$).

From Eqn. 3.8, we observe that while maintaining the inter-cluster network (keeping $k$ and $W$ fixed), a smaller cluster size ($c' < c$) leads to a higher value of $\lambda_{nw}$ and vice-versa. The value of $\lambda_{sw}$ remaining fixed implies that a configuration with smaller cluster size has $\lambda_{sw} < \lambda_{nw}$, making it message overhead constrained (MC). Similarly, a configuration with larger cluster size ($c' > c$) can be shown to have $\lambda_{sw} > \lambda_{nw}$, making it network bisection constrained (NC).

The above theorem leads us to divide a typical $R$-$c$ plane into the following three regions: B-boundary, MC-region, and NC-region. The B-boundary is derived by joining the set of balanced configurations by a curve as shown in Fig. 9(a). By Theorem 1, any configuration to the left of the B-boundary is MC. Thus, the configurations enclosed by the B-boundary and the $R$ and $c$ axes are MC. This region is denoted as the MC-region. Similarly, any configuration to the right of the B-boundary is NC. Thus, the open part of the $R$-$c$ plane is referred to as the NC-region. The following result reflects the communication performance in the three regions, respectively:

Result 1 All configurations within the MC-region and on the B-boundary have the same maximal throughput performance of $\lambda = \lambda_{sw}$. Within the NC-region maximal
throughput performance is determined by $\lambda_{nw}$ and, based on Eqn. 3.8, this throughput falls with increasing cluster and inter-cluster size.

The impact of increasing inter-cluster channel width ($W$) from $W_1$ to a higher value $W_2$ is shown in Fig. 9(b). Let us consider a balanced configuration $(R, c, W_1)$ on the B-boundary for $W = W_1$. From Eqn. 3.8, a configuration $(R, c, W_2)$ with $W_2 > W_1$ has higher $\lambda_{nw}$. Thus, $(R, c, W_2)$ has $\lambda_{sw} < \lambda_{nw}$ leading us to conclude it to be MC configuration. From Theorem 1 a balanced configuration with $W = W_2$ has cluster size $> c$. Consequently, with increase in $W$ the B-boundary shifts towards the right (outwards), resulting in the MC-region to also become larger.

![Figure 9](image)

Figure 9: a) A typical $R-c$ plane depicting three regions into which parallel system configurations fall under: **B-boundary** (balanced), **MC-region** (messaging overhead constrained), and **NC-region** (network-bisection constrained). b) Impact of increasing $W$ from $W_1$ to a higher value $W_2$, the B-boundary shifts outwards, resulting in a larger MC-region.

A system with a given number of processors can be designed using different configurations. Each configuration having different cost and communication performance characteristics. Depending on the messaging and network parameters a configuration could belong to the MC-region, the B-boundary, or the NC-region. Such variation
in characteristics raises an interesting question: for designing a system with a given number of processors which configuration delivers best performance in a cost-effective manner? This question is addressed in the next section where we analyze the impact on performance and cost of a system, with a total of \( N \) processors, as larger cluster sizes are used.

### 3.4 Designing a system with a given number of processors

For designing a system with \( N \) processors with clusters of size, \( c \), the number of required routers is given by \( R = N/c \). Higher integration is leading to higher value of \( c \) and as a result smaller \( R \). However, there are limits on what technology [8] can offer at a given time. For example, the maximum allowable cluster size can be limited. The impact of such constraints on the design is studied in Chap. V. In this section we present a more general analysis without restraining the study.

Figure 10 illustrates three different configurations \( X (64,1,1) \), \( Y (32,2,1) \), and \( Z (16,4,1) \).
Z (16,4,1) on an iso-processor line on the $R - c$ plane for $W = 1$ byte. All three configurations can be used to build a system with $N = 64$ processors. In Fig. 10 configuration $X$ lies in the MC-region, $Y$ is balanced on the $B$-boundary, and $Z$ lies in the NC-region. From Result 1, the performances ($\lambda$) at configurations $X$, $Y$, and $Z$ are $\lambda_{sw}$, $\lambda_{sw}$, and $\lambda_{nw}$, respectively. For configurations on an iso-processor line for a given $N$ and $W$, we are interested in deriving the following:

1. The balanced configuration at which network and messaging performance ($\lambda_{sw}$) match each other?

2. For configurations using cluster size larger than at the balanced configuration, the required scaling in channel width to maintain performance at $\lambda_{sw}$?

3. Estimating inter-cluster interconnection costs for configurations as larger cluster sizes are used.

We analyze each of these in the following subsections.

3.4.1 Deriving the balanced configuration for a given channel width

Let $c_{bal}^{N,W}$ denote the cluster size at the balanced configuration. The routers ($R = N/c_{bal}^{N,W}$) in the system are assumed to be organized in a $n$-D mesh topology with radix $k = (N/c_{bal}^{N,W})^{1/n}$. For such configuration, equating $\lambda_{nw} \approx \lambda_{sw}$ obtained from Eqns. 3.2 and 3.8 leads us to:

$$1/ \max(t_s, t_p L) = \frac{2W}{c_{bal}^{N,W} L((N/c_{bal}^{N,W})^{1/n} - 1)t_c(1 + ((n + 1)/3)}.$$

(3.10)
For \( n = 2 \) this simplifies into a quadratic equation in \( \sqrt{c_{bal}^{N,W}} \) as:
\[
\sqrt{N} \sqrt{c_{bal}^{N,W} - c_{bal}^{N,W}} = \frac{W \max(t_s, t_p L)}{L t_c} \tag{3.11}
\]

Solutions to Eqn. 3.11 which round off to positive integers are proper solutions to balanced configurations. An obtained solution should also lie within the limits of offered cluster technology. Depending on the values of timing parameters, average message length in the system, channel width, and system size, the derived value of \( c_{bal}^{N,W} \) can vary over a wide range. Consider a typical example with \( t_s = 3.0 \mu \text{sec} \), \( t_p = 0.01 \mu \text{sec}/\text{byte} \), \( L = 32 \text{ bytes} \), \( W = 1 \text{ byte} \), \( t_c = 0.01 \mu \text{sec} \), and \( N = 64 \). The two derived values of \( c_{bal}^{N,W} \), based on Eqn. 3.11, are 2 and 43. Clearly, the solution 43 is too large to be reasonable, leaving only \( c_{bal}^{N,W} = 2 \). We validated this example experimentally. Three configurations: (8x8,1,1), (8x4,2,1), and (4x4,4,1) corresponding to \( X \), \( Y \), and \( Z \) in Fig. 10 (a), respectively, were considered. Other parameters were chosen same as mentioned above. Note that our simulator is general enough to model networks with unequal radixes like 8x4. Figure 11 shows the network latency versus throughput plots for these three systems. Given \( t_s = 3.0 \), \( \lambda_{sw} \) is \( \approx 0.3 \text{ messages/\mu sec} \). We observe such \( \lambda_{sw} \) performance to be delivered by the (8x8,1,1) and (8x4,2,1) configurations but not by the (4x4,4,1) configuration. This indicates (8x4,2,1) to be the balanced configuration with cluster size of two leading us to conclude \( c_{bal}^{64,1} = 2 \).

### 3.4.2 Required scaling in channel width with larger cluster sizes

In configurations using \( c > c_{bal}^{N,W} \) and channel width \( W \), it can be observed that network bisection performance becomes a bottleneck leading to overall performance
Figure 11: Latency vs. throughput plots for a) three configurations with $N = 64$ processors and channel width $W = 1$ byte: (8x8,1,1), (8x4,2,1), and (4x4,4,1) and b) (4x4,4,2) configuration with wider channel width ($W = 2$ bytes).

falling below $\lambda_{sw}$. In the above example the configuration (4x4,4,1) with $c = 4$ ($c_{bal} = 2$) delivered less throughput than desired ($\lambda_{sw} = 0.3$). Let us derive the minimum increased channel width $W'$ to maintain performance at $\lambda = \lambda_{sw}$ in such configurations while using clusters of size $c$. The network performance, can be derived, using Eqn. 3.8 and $k = (N/c)^{1/n}$, as $\lambda_{nw} = \frac{2W'}{cL((N/c)^{1/n} - 1)c(1+(n+1)/3)}$. Equating this to $\lambda_{sw}$ (or the network performance at the balanced configuration in Eqn. 3.10), and simplifying we obtain: $\frac{W}{c_{bal}((N/c_{bal})^{1/n} - 1)} = \frac{W'}{c((N/c)^{1/n} - 1)}$. This leads to:

$$W' = \left[ \frac{Wc((N/c)^{1/n} - 1)}{c_{bal}N,N,W}((N/c_{bal}W)^{1/n} - 1) \right].$$

(3.12)

where the ceiling function ensures byte-multiple channel widths as discussed in Sec. 3.1.

Using wider channels can be visualized as transitioning from one $Rc$ plane to another $Rc$ plane with higher channel width as depicted in gray on the iso-processor line in Fig. 12. Configuration Z, with $W = 1$ byte, was depicted in Fig. 10 as being in the NC-region offering $\lambda = \lambda_{nw} < \lambda_{sw}$. It is shown in Fig. 12, that this configuration with $W = 2$ bytes, moves within the B-boundary delivering $\lambda = \lambda_{sw}$. This was verified by
considering a corresponding (4x4,4,2) configuration in our experiments. The network latency versus throughput plot for this system is depicted in Fig. 11 (b). It can be observed that it delivers maximal throughput matching $\lambda_{sw} = 0.3$ messages/µsec.

![Network latency versus throughput plot](image)

Figure 12: Increased channel width in the Z configuration (from $W = 1$ to $W = 2$) leading to configuration Z becoming balanced and offering $\lambda = \lambda_{sw}$.

### 3.4.3 Inter-cluster interconnection cost

The cost of the inter-cluster interconnection network comprises of router and link costs which depend on the channel width of the system. We are interested in comparing interconnection costs across different configurations. Such cost comparison is considered easier across system configurations having same channel width. With same channel width the per-router and link costs are expected to be same, thus, a system requiring fewer routers and interconnection links is expected to have lower expensive interconnects. Cost comparisons across configurations with different channel widths is more difficult. Configurations with wider channel width need routers
and link connectors with matching wider data paths inside. Such routers and connectors are expected to have higher costs. Let us first compute the total number of links, denoted earlier as $N_W$, in the inter-cluster interconnection. This is an indication of connector/connectivity cost across routers [8]. Similarly, we estimate the change in router costs. Overall, the following analysis supports the intuition that interconnection costs reduce with larger cluster sizes (and possibly wider channels) while maintaining per-processor communication performance.

Let us derive the expression for $N_W$ in a $(R, c, W)$ design configuration, with $N = Rc$ and $R = k^n$, where $k = (N/c)^{1/n}$, and the $k^n$ routers interconnected as an $n$-dimensional mesh. Each row along any dimension has $k$ routers with $(k - 1)$ interconnection channels. For bidirectional channels this number is $2(k - 1)$. Given $n$ dimensions each having $k^{n-1}$ rows, the total rows in the network is $nk^{n-1}$ leading to a total of $2nk^{n-1}(k - 1)$ network channels. Each channel being $W$ wires wide, the total number of interconnection wires in the network can be derived as:

$$N_W = 2nk^{n-1}(k - 1)W = 2n(N/c)^{n-1/n}((N/c)^{1/n} - 1)W.$$ (3.13)

Let us consider the impact on total inter-cluster wiring costs ($N_W$) as larger cluster sizes (and possibly wider channels as indicated in Eqn. 3.12) are used to interconnect a system with $N$ processors while maintaining performance. Using Eqns 3.12 and 3.14 we have:

$$N_W = \begin{cases} 
2n(N/c)^{n-1/n}((N/c)^{1/n} - 1)W & \text{if } c \leq c_{b_{\text{bal}}}^{N,W} \\
2n(N/c)^{n-1/n}((N/c)^{1/n} - 1) \frac{Wc((N/c)^{1/n} - 1)}{b_{\text{bal}}^{N,W} ((N/c)^{1/n} - 1)^{1/n}} & \text{if } c > c_{b_{\text{bal}}}^{N,W}
\end{cases} \quad (3.14)$$

It can be shown that $N_W$ is a monotonically decreasing function of $c$ proving the intuition that link connector/connectivity costs reduce with larger clusters.
Similarly, a router supporting wider channels requires wider data paths inside and higher pinout outside. Predicting the impact of such increased pinout and wider paths on router cost is not always easy. Do we model the impact of increased channel width on router cost as sub-linear, linear, or super-linear? Market forces often have a critical role to play in determining such costs. Initial costs may be very high (super-linear) but with increasing demand leading to cheaper production [38], typical costs tend to become sub-linear. A linear model is therefore reasonable and considered in further analysis. Let \( H^{n,W} \) denote the cost of a router with \( n \) ports (supporting an \( n \)-dimensional mesh) each \( W \) bits wide. For a linear cost model we derive the cost of a router offering \( W' \) bit wide ports as \( H^{n,W}(W'/W) \). Let \( H_{tot} \) denote the total router costs in a system. For a system with \( R \) routers each having \( n \) ports of \( W \) bits wide leads to \( H_{tot} = RH^{n,W} \). As larger cluster sizes (and possibly wider channels as derived in Eqn. 3.12) are used to interconnect a system with \( N \) processors while maintaining performance, the total router cost becomes:

\[
H_{tot} = \begin{cases} 
(N/c)H^{n,W} & \text{if } c \leq c_{bal}^{N,W} \\
(N/c)H^{n,W} \frac{Wc((N/c)^{1/n}-1)}{c_{bal}^{N,W} ((N/c_{bal}^{N,W})^{1/n}-1)} & \text{if } c > c_{bal}^{N,W}
\end{cases}
\]  

(3.15)

It can be shown that the above expression for \( H_{tot} \) is a monotonically decreasing function of \( c \). Equations 3.14 and 3.15 lead us to the following result:

**Result 2** For designing a system with a given number of processors, larger processor-clusters offer effectiveness in terms of link connectivity and router costs, while offering equal performance.
3.5 Scaling systems to support larger number of processors

In the last section we investigated the design space for building a system with a given number of processors. In this section we analyze scaling system size using larger cluster sizes. The scaling of a system can be represented on the $R - c - W$ space as moving from a configuration on one iso-processor plane to a larger configuration on a higher iso-processor plane. Let us consider an example of scaling a configuration $Y$ with 64 processors to a larger configuration with 128 processors as shown in Fig. 13(a). In this illustration four possible scaled configurations are shown: $Y_1$, $Y_2$, $Y_3$, and $Y_4$. The respective cluster sizes used increase in these four configurations. From discussion in the last section, maintaining performance at $\lambda = \lambda_{sw}$ may require increasing channel width with increasing cluster size. However, from Result 2, link connectivity and router costs are expected to decrease with larger clusters. Thus, it is reasonable to expect that given an existing system $Y$ using clusters of size $c'$, a scaled system with similar or better available cluster technology need to use $c \geq c'$.

Figure 13: a) Different options to scale a configuration $Y$ with 64 processors to a larger size of 128 processors. b) Scaling with same topology but larger clusters ($Y$ to $Y_3$ option).

In Fig. 13, the scaling from $Y$ to $Y_1$ depicts an approach using smaller cluster
size with larger network. Based on the above discussion this approach is not attractive from link and router cost considerations. It can be profitable if smaller cluster technology offers a significant cost advantage over larger clusters so as to offset the savings in link and router costs. The scaling from $Y$ to $Y_2$ is the default approach by maintaining cluster size while interconnecting more clusters. Such approach is useful if clusters with larger number of processors are not available. However, given that advancements in cluster technology are offering larger cluster modules, scaling approaches as depicted in $Y$ to $Y_3$ and $Y_4$ using larger cluster technology are becoming more attractive. In this section we study one such approach (from $Y$ to $Y_3$) in more detail. A transition from $Y_3$ to $Y_4$ can be achieved by the principles discussed in Sec. 3.4.

The approach is depicted as a horizontal dotted arrow along the $c$ axis in Fig. 13(a). Scaling is achieved by using larger clusters while maintaining network topology. Let $N_{base}$ denote the size of a message-constrained (MC) base configuration with $R$ routers and channel width $W$. Figure 13(b) shows the scaling of such a configuration with larger clusters. With larger systems the network is more utilized leading us to balanced and then into the NC-region. Let $N^{R,W}_{bal}$ denote the size of the balanced configuration. Thus, scaling up to a size of $N' < N^{R,W}_{bal}$ can be achieved without increasing $W$. Scaling beyond $N^{R,W}_{bal}$ to a larger size $N''$ requires larger $W$ to maintain performance. This leads us to analyze the following subproblems:

1. What is the largest system size achievable ($N^{R,W}_{bal}$), on a given network topology with $R$ routers and channel width $W$, using large clusters while maintaining
per-processor communication performance at $\lambda = \lambda_{sw}$?

2. With further scaling using larger clusters, while maintaining network topology fixed with $R$ routers and per-processor communication performance at $\lambda = \lambda_{sw}$, what is the minimal required increase in channel width?

3.5.1 Deriving the largest system size supportable on a given inter-cluster network

For a given $R$ and $W$ let us derive the value of $N_{bal}^{R,W}$. Let $c_{bal}^{R,W}$ denote the cluster size at such balanced configuration. Thus, $c_{bal}^{R,W} = N_{bal}^{R,W} / R$. Let us consider the network performance as derived in Eqn. 3.8. Keeping network topology fixed during scaling implies keeping $k$ fixed, leading to $\lambda_{nw} \propto 1/c$. For such scaling let us consider a base configuration $(R, c, W)$ with $\lambda_{nw} > \lambda_{sw}$. Let $f$ denote the factor $\lambda_{nw}/\lambda_{sw}$. This indicates a base configuration with under-utilized network bandwidth (and therefore in MC-region). As cluster size is increased from $c$ the per-processor network performance ($\lambda_{nw}$) falls proportional to the increase in cluster size becoming closer to $\lambda_{sw}$, with the overall network bandwidth being utilized more effectively. At cluster size ($c_{bal}^{R,W} = cf$) the parameters $\lambda_{nw}$ and $\lambda_{sw}$ are balanced. This leads us to:

**Result 3** Given a base system $(R, c, W)$ with $\lambda_{nw} > \lambda_{sw}$ and $f = \lambda_{nw}/\lambda_{sw}$, designing a $f'$ ($f' \leq f$) times larger system delivering similar performance requires using clusters of size $cf'$ while keeping the inter-cluster network size $(k_1 \times \cdots \times k_n)$ and channel width ($W$) fixed.
For a given base network, the value of $c^{R,W}_{bal}$ can be derived using Eqns. 3.2 and 3.8, equating these expressions and solving for $c$. Thus, $c^{R,W}_{bal}$ is the solution of $c$ in

\[
\frac{1}{\max(t_s, t_p L)} = \frac{2W}{cL(k-1)tc(1+(n+1)/3)} \quad \text{or} \quad c^{R,W}_{bal} = \frac{2W \max(t_s, t_p L)}{L(k-1)tc(1+(n+1)/3)}.
\]

We verified Result 3 through simulation modeling of a fixed network interconnecting processor-clusters of increasing size. Figure 14 shows the latency versus throughput characteristics of (8x8,c,2) configurations with $c = 1, 2, 4, \text{and } 8$. The plots are shown for two different message overheads $t_s = 2.0$ and $5.0\mu\text{sec/message}$ (corresponding to $\lambda_{sw} = 0.5$ and $0.2$ messages/$\mu\text{sec}$). Using simulation experiments with $t_s \to 0.0$, the value of $\lambda_{nw}$ in a (8x8,1,2) system was derived as 1.0 message/$\mu\text{sec}$. Under $t_s = 2.0$, $\lambda = 0.48$ messages/$\mu\text{sec}$ (close to 0.5) was observed on a (8x8,1,2) system. The maximal cluster size supported under $t_s = 2.0$, without degradation in $\lambda \approx 0.5$, was confirmed to be $c = \lambda_{nw}/\lambda_{sw} \approx 2$. With larger cluster sizes ($c > 2$), the achieved value of $\lambda$ dropped significantly. Similarly, maximum cluster size under $t_s = 5.0$ ($\lambda_{sw} = 0.2$), was observed as 4 (from the expression $\lambda_{nw}/\lambda_{sw}$ it should be $1.0/0.2 = 5$, but in our experiments we varied cluster size as powers of 2). These observations validate Result 3.

Scaling using larger clusters while maintaining the same network works until $\lambda_{nw}/\lambda_{sw} > 1$. When a scaled system configuration becomes balanced ($f = 1$), the above approach is no longer applicable. Further scaling leads to degradation in system performance.
Figure 14: Determining the largest system size that can be supported without degradation in performance on a given base network by using larger clusters. Results are shown for $t_s = 2.0$ and $5.0 \mu s/\text{message}$.

### 3.5.2 Required increase in channel width for further scaling

Let us now consider the problem of scaling a $(R, c, W)$ base system, which is not message-constrained ($\lambda_{aw} \geq \lambda_{nw}$), using larger clusters. In such a system the network is not under-utilized and the achieved throughput per processor is derived as $\lambda = \lambda_{nw}$.

In scaled configurations it is desired to maintain the same performance. In scaling a $(R, c, W)$ system by a factor $g$ while keeping inter-cluster topology fixed, larger clusters with $cg$ processors are used. From Eqn. 3.8 the inter-cluster channel width needs to be proportionally increased to $Wg$ to maintain $\lambda_{nw}$. This leads us to:

**Result 4** A base system, not message constrained, with $(R, c, W)$ configuration can be scaled $g$ times, while maintaining the average throughput in the system, by using $(R, cg, Wg)$ configuration.

We verified that system performance remains unchanged when a base system is scaled to larger sizes using the approach proposed in Result 4. We first considered
scaling the balanced configuration, (8x8,2,2) (derived in the above subsection with $t_s = 2.0\mu\text{sec}$), two times to a (8x8,4,4) configuration. Figure 15 compares the latency versus throughput results for the (8x8,2,2) and (8x8,4,4) configurations. It can be observed that the scaled configuration offers very similar throughput, thus, validating Result 4.

![Figure 15: Studying the scalability of a balanced configuration (8x8,2,2) using larger cluster and proportionately larger channel width. A $t_s = 2.0\mu\text{sec}$ was assumed.](image)

To further validate Result 4 we also considered scaling a network-constrained base system with (4x4,1,2) configuration up to large cluster sizes ($c = 16$). To ensure the base system being indeed network-constrained we chose a very small message overhead of $t_s = 0.25\mu\text{sec}$. Figure 16 shows the latency versus throughput results for the (4x4,1,2) configuration and the corresponding larger configurations (4x4,$c',2c'$), $1 \leq c' \leq 16$. It is observed that the performance is maintained in larger systems resulting in ideal scaling. Similar results were obtained with other configurations like (8x8,$c',2c'$), $1 \leq c' \leq 8$. 
Figure 16: Studying the scalability of the approach of using larger clusters and proportionately larger channel widths. To ensure configurations not being MC (i.e. being B or NC), a very small \( t_s = 0.25 \mu \text{sec} \) was assumed.

### 3.6 Experimental validation - pattern and application based approach

In the previous sections we validated the theory using uniform traffic. In this section we present similar results derived using a set of specific traffic patterns/applications. As earlier, the objective is to demonstrate that the architectural configurations derived with the proposed guidelines using larger clusters maintain system performance while being more cost-effective. Our methodology involves executing traffic patterns/applications on a set of configurations and comparing the resultant execution times.

#### 3.6.1 Experimental platform

Experiments were conducted on the ClusterSim simulator as described in Sec. 3.2.3. We also conducted experiments on a 64 node Cray T3D for configurations supportable
on it. The T3D machine is a MIMD multiprocessor machine using a direct communication network with 3D torus topology [23]. Two alpha processors are attached to each node (cluster) of the torus and share a common hardware interface to the network. Each processor has a private memory. For communication across processors we used the remote memory access routines: `shmem_put()` and `shmem_get()`, offering the maximum communication bandwidth [5, 45].

3.6.2 Traffic patterns/applications

We selected three traffic patterns/applications [46]: Bit Permute Complement exchanges (BPC), Fast Fourier Transform (FFT), and LU matrix decomposition. These are briefly discussed below:

1. **BPC (Bit permute complement):** A communication round in BPC involves a message being sent out by each node to a destination, decided by a bit-permute-complement operation on its radix 2 node number representation. In a given round all nodes apply the same bit-permute-complement bit operation and so that each node receives exactly one message in any round. BPC is an important benchmark as it exercises the communications of a computer heavily on a realistic problem. Each BPC requires $O(N)$ number of messages, where $N$ is the number of processors in the system.

2. **Fast Fourier Transform (FFT):** The computation of the Fast Fourier transforms (FFTs) is the cornerstone of many scientific applications. It is a rigorous test of long-distance communication performance. We use the Cooley Tuckey
algorithm to compute the FFT with interleaved data allocation. An FFT requires $O(N \log N)$ number of messages. While executing FFT across systems with varying number of processors we scale problem size linearly with $N$.

3. Dense LU factorization with partial pivoting: This computation appears in almost any numeric application. Searching for a pivot is basically a reduction operation within one column of the processor mesh. Exchange of pivot rows is a point-to-point communication. Update phase requires data to be broadcast along rows and columns of the processor mesh. The number of messages required in LU is $O(N^2)$. However, computation being $O(N^3)$ dominates communication. The problem size is scaled linearly with $N$ when running on a system with more processors.

3.6.3 Designing a system with a given number of processors

Deriving the balanced configuration for a given channel width

We evaluated the completion times of LU, BPC, and FFT on three different configurations having $N = 64$ processors: (8x4,1,2), (8x4,2,2) and (4x4,4,2). A messaging overhead of $t_s = 2.0\mu\text{sec/message}$ closely matching that on the Cray T3D was used. The left column of Fig. 17 shows three sets of bar charts depicting completion times of LU, BPC, and FFT, respectively, derived on the ClusterSim simulator for the three configurations. For each application the completion times presented are normalized with respect to the smallest of the timings obtained. The completion time of LU can be observed to be similar on all three systems. BPC also yielded similar results. However, with FFT, characterized by very rigorous long-distance communication,
the small network bisection bandwidth in the (4x4,4,2) configuration becomes a bottleneck. This is reflected by a 10% increase in completion time over the (8x8,1,2) configuration. These results lead us to conclude that among these configurations, performance can be maintained only up to cluster size of 2, implying (8x4,2,2) to be the balanced configuration.

Similar results were confirmed on a Cray T3D system on which only the (8x8,1,2) and (8x4,2,2) configurations are supportable. The (8x8,1,2) configuration was attained by choosing a (8x8,2,2) partition of the system and running the application on exactly one processor in each cluster. The right column of Fig. 17 shows three sets of bar charts depicting completion times of LU, BPC, and FFT, respectively. It can be observed that for all three applications the achieved performances of the (8x8,1,2) and (8x4,2,2) configurations are similar. This verifies the simulation results and leads us to conclude cluster size of 2 being balanced. These trends also validate the theory established in Sec. 3.4.1.

Required scaling in channel width with larger clusters

In Sec. 3.4.2 we demonstrated the required increase in channel width to maintain performance beyond the balanced configuration. In the previous example the (4x4,4,2) configuration did not match the FFT performance on the (8x8,1,2) and (8x4,2,2) configurations. A (4x4,4,4) configuration with increased channel width $W = 4$ (derived based on Eqn. 3.12) can be shown to offer matching performance. Figure 18 compares the completion times of FFT on the (4x4,4,4) configuration with that on the (8x8,1,2) and (4x4,4,2) configurations, derived on the ClusterSim simulator with
Figure 17: Impact of using larger clusters in designing systems: deriving the balanced configuration. The normalized completion times are depicted for LU, BPC, and FFT on three configurations on ClusterSim and two configurations on the Cray T3D. On ClusterSim parameters of $t_s = 2.0\mu$sec/message and $W = 2$ bytes, matching those on the T3D, are assumed. Number of processors is kept fixed at $N = 64$. 
$t_s = 2.0 \mu\text{sec}$. It can be observed that there is a close match in the completion times on the $(8\times8, 1, 2)$ and $(4\times4, 4, 4)$ configuration, thus confirming the theory in Sec. 3.4.2.

![Figure 18: Matching performance with wider channels beyond the balanced configuration. The normalized completion times of FFT is shown on three configurations. Parameters $N = 64$ and $t_s = 2.0 \mu\text{sec}$ are assumed.](image)

**3.6.4 Scaling systems to support larger number of processors**

**Deriving the largest system size supportable on a given inter-cluster network**

In Sec. 3.5.1 we derived that with high messaging overheads, a network can support larger systems with larger clusters (up to a certain point) while maintaining performance. This was verified with uniform traffic, on a $8\times8$ mesh inter-cluster network with channel width of 2 bytes and $t_s = 5.0 \mu\text{sec}$. The cluster size was varied from 1 to 8 processors. With such parameters the largest system size was derived as 256 obtained with cluster size of 4. Similar results were also obtained with BPC and FFT traffic as discussed below.
To determine the execution time of an application on a scaled system we considered problem size to be similarly scaled. The execution time of the scaled problem on a scaled system is a function of the communication structure/volume of the problem. For the two applications, BPC and FFT, we use communication volume as a metric to determine the expected increase in completion time of these applications on larger systems. For example, the communication in BPC is $O(N)$. Thus, in BPC increasing the number of processors and application size by a given factor leads to similar increase in the volume of communication. Such scaling is expected to lead to minimal increase in the completion time until the network bisection bandwidth resources are under-utilized. However, beyond a certain system size the network no longer has under-utilized bandwidth and the time required is expected to sharply increase. Such results were observed for BPC, as shown in Fig. 19. The completion times for the (8x8,2,2), and (8x8,4,2) configurations did not increase substantially over the (8x8,1,2) configuration. However, for the (8x8,8,2) configuration the completion time jumped to 175%. This indicates that for the given network and messaging characteristics scaling is effective up to a factor of 4. This matches the results presented in Fig. 14 with uniform traffic.

The communication traffic in FFT is more complex. It consists of $\log(N)$ rounds, with each round requiring $O(N)$ messages. Thus, increasing the system size (along with problem size) leads to completion time increasing by at least the number of extra rounds required. As system size is increased each round becomes more communication intensive. However, initially when the network is under-utilized such increase in
communication translates to minimal increase in execution time of a round. However, beyond a certain system size the network no longer has under-utilized bandwidth and the time required per round is expected to sharply increase. Such results were observed with FFT, as shown in Fig. 19. The completion times for the (8x8,2,2) and (8x8,4,2) configurations were derived to be 120% and 140%, respectively, compared to the completion time on a base (8x8,1,2) configuration. These values are close to the expected increase in completion times of 117% and 134%, respectively. However, for the (8x8,8,2) configuration the completion time increases sharply to 215% (expected 151%) indicating network resources becoming a severe bottleneck. For the given network and messaging characteristics scaling is effective up to a factor of 4, thus, matching the results derived earlier.

![Graph](image1.png)

**Figure 19: Impact of using larger clusters in scaling a base system having under-utilized network bisection resources: determining largest system (cluster) size supportable on same network. Normalized completion times are shown for BPC and FFT on a (8x8,1,2) base system and its scaled counterparts having larger clusters. The parameters of \( t_s = 5.0 \mu \text{sec/message} \) and \( W = 2 \) bytes are assumed.**
Required increase in channel width for further scaling

In Sec 3.5.2 we studied the approach of scaling configurations which are not message constrained i.e. do not have under-utilized network resources. It was demonstrated that scaling in such systems could be achieved in a cost-effective manner by using still larger clusters with wider network channels. This was achieved while maintaining per-processor performance with uniform traffic. We evaluated the same approach with FFT traffic on three system configurations: (8x8,1,2), (8x8,2,4), and (8x8,4,8). As discussed in Sec. 3.5.2 a low messaging overhead of $t_s = 0.25 \mu s e c$ was used in this experiment to ensure that the base system, (8x8,1,2), is not message-constrained (it is network-constrained). The normalized completion times are shown in Fig. 20. As discussed earlier, the communication in FFT has $\log(N)$ rounds, with each round requiring $O(N)$ messages. The increased communication per round is expected to be matched by increased network bisection bandwidth. Thus, the completion time is expected to increase proportional to the number of rounds required - $O(\log N)$. These trends can be observed in Fig. 20 validating the results obtained earlier in Sec. 3.5.2.

3.7 Summary

In this chapter we have demonstrated that messaging overheads can limit interconnection utilization. Even with new messaging protocols such overheads have been shown to remain high compared to network speeds. In order to alleviate the impact of high messaging overheads, we have analyzed an alternative of exploiting the emerging
processor-cluster technology for designing high-performance and cost-effective parallel systems. Using a new design-space-graph framework which considers processor-clustering, messaging overheads, and network performance in an integrated manner, we have established the following design guidelines. With messaging overheads constraining performance in a system, processor clustering can be used to build a) an equal-sized system with a smaller network or b) a larger system with an equal-sized network. With messaging overheads not being a constraint, a combination of processor clustering and wider channels can be used to build a range of larger-sized systems. The design guidelines have been validated through a large set of simulation experiments as well as experimentation on Cray T3D. To the best of our knowledge, this is the first formal analysis in literature that demonstrates when and how processor-clusters benefit parallel system design.
CHAPTER IV

Routing and communication interface design

Messages sent from one processor in one cluster to another processor in another cluster may need to travel over a sequence of channels in the inter-cluster and intra-cluster networks. To ensure correct and guaranteed delivery of such a message, the networks need to support an efficient routing scheme. Towards this goal, in this thesis we consider wormhole-routing to be used in direct networks. The wormhole-routing switching technique, often referred to as wormhole routing, is becoming the trend in building parallel systems due to its inherent advantages like low-latency communication and reduced communication hardware overhead [52]. Intel Paragon [42], Cray T3D [23], Ncube, J-Machine [53], and Stanford DASH/FLASH [35] are representative systems falling into this category.

In the wormhole-switching technique, the flow of a header flit triggers the movement of the remaining flits in a message. If the header flit gets blocked due to unavailability of a network channel, the remaining flits are blocked in the network. Such blocking, besides leading to under-utilization of channels, may block other messages in the network. Wormhole routing on single-level networks is a well-studied
problem. However, on multi-level networks like those in clustered architectures, supporting *deadlock-free* routing delivering low-latency is still an open problem and needs to be solved.

Although wormhole-routing was initially proposed with *deterministic* routing [26, 22], a number of *adaptive* routing schemes have been proposed recently by many researchers [18, 20, 48, 49, 59] to reduce such message blocking in the network. Similarly, the use of multiple *virtual networks* (or lanes) with *Virtual Channel Flow Control* (VCFC) has been proposed [25] in literature to be used with either deterministic or adaptive routing.

These schemes emphasize reducing message blocking or contention in the network. However, once a worm reaches a destination node, it needs to be *consumed* (taken out from network) [52]. Thus, after the header flit of a worm reaches its destination the movement of its subsequent flits in the network depends on the rate at which the message gets consumed. This movement depends on the *consumption capacity* available at the destination node-network interface, which can be characterized by number of *consumption channels* for message consumption. If the worm gets blocked or slowed down due to limited consumption channels, then the subsequent flits of the worm remain in the network for additional time leading to wastage of network bandwidth. In [21], it was first reported that wormhole systems can undergo severe performance degradation due to limited number of router-to-processor consumption channels. The results were based on experimentation on the deterministic wormhole routed Symult-2010 system and suggested using multiple consumption channels in
future wormhole systems. Thus, a good network design must consider its routing scheme together with the consumption capacity available at a node to evaluate its overall performance.

Most contemporary systems provide only one consumption channel per node. Under traffic conditions when multiple messages are destined to a node within a short duration, a single consumption channel can become a bottleneck. For example, many parallel applications exhibit non-uniform traffic with hot-spots [29, 47, 76], both in distributed-memory and distributed-shared-memory paradigms. Such traffic leads to more messages being destined towards some nodes in the system. Even with uniform traffic, at higher rate of communication, there is more probability that some nodes receive multiple messages during a small interval of time. For both types of traffic, if multiple messages reaching a destination node can not be consumed at full speed due to limited consumption capacity, they get blocked in the network surrounding the destination node. Such blocking can lead to domino-blocking (tree-saturation)[58] in the network which may affect the propagation of other messages in the system. This has potential to degrade performance in terms of lower throughput and higher latency. Adaptive routing and VCFC schemes aim to achieve higher network performance by alleviating message blocking in the network. However, with a closer look it can be observed that these schemes drive more messages to nodes, leading to increasing demand on the consumption capacity at a node. In the absence of sufficient consumption capacity at a node the advantages of these schemes become somewhat limited.
Over the years multiple virtual channels per network channel has been proposed to provide deadlock-freedom [31], routing adaptivity [4, 19], and VCFC [25, 63]. Contemporary systems are adopting such multiple virtual channels/lanes on network channels [23]. In such systems, a consumption channel is required to support virtual multiplexing. Otherwise, virtual multiplexing on the network channels, causing slowdown in propagation of messages, can lead to messages arriving at their destinations at a speed slower than the full network channel speed. This can lead to situations where a message having a consumption channel is consumed slowly (not utilizing the full consumption channel bandwidth) while another message is blocked on the consumption channel. Thus, a good system design should incorporate virtual consumption channels to take advantages of the benefits of adaptivity and VCFC schemes.

The above discussion suggests that an architect must provide sufficient consumption capacity as well as virtual consumption channels to alleviate consumption channel bottleneck problem and obtain better performance. However, there is no systematic study of this problem in the literature to provide necessary design guidelines to an architect. In this chapter we take on such challenges and propose a new network interface design with multiple consumption channels from a router to its cluster. To avoid wastage of consumption channel bandwidth due to message slowdown by virtual multiplexing in the network, we propose using matching number of virtual lanes on each consumption channel. A message arriving at a destination node chooses a free consumption channel, gets consumed, and buffered at the interface. A direct memory
access (DMA) then transfers the message to the processor-memory over the local bus.

In designing such a network interface, the main question needs to be solved is: How many consumption channels should we provide in a given system? Clearly, the blocking effect on a consumption channel (defined by us as consumption collision) reduces with increasing number of channels. As a naive solution one can provide $2^n$ physical consumption channels per node each having $v$ virtual channels\(^1\), where $n = $ network dimensionality and $v = $ number of virtual lanes per network channel. In such a system at most $2^{nv}$ messages can arrive simultaneously at a node, and thus having a total of $2^{nv}$ virtual consumption channels can handle this worst case. However, this naive solution requiring $2^n$ physical channels is an overkill and may not be realizable due to limited router pinout constraint. Thus, the challenge is to determine how many physical consumption channels are effective as well as realizable to deliver best performance.

In order to derive this effective number, in this chapter, we consider several factors like network bisection, messaging overheads, interface-to-processor bandwidth, system dimensionality, routing adaptivity, and virtual networks in an integrated manner. These factors determine the rate of messages arriving at a node. First, we develop an analytical consumption collision model to estimate consumption collision delay based on the message arrival rates. Using this model we analyze the interplay between the above parameters and the number of consumption channels. Using this analysis, we derive guidelines and trends for using effective number of consumption channels under

\(^1\)The terms virtual lane and virtual channel are used interchangeably in this thesis
different values of the above factors. Finally, through detailed flit-level simulation, we verify these guidelines for a range of system sizes, system dimensionalities, routing adaptivity, messaging overheads, interface-to-processor bandwidths, and number of virtual networks. The overall study indicates that 2-4 consumption channels per node need to be used on current and future generation wormhole $k$-ary $n$-cube systems to alleviate the consumption channel bottleneck. It is also demonstrated that these consumption channels are realizable under current and future router pinout constraints.

The chapter is organized as follows. Section 4.1 presents routing alternatives in a clustered system. Section 4.2 introduces the new network interface design with more consumption channels and virtual multiplexing on consumption channels to match network channel multiplexing. Section 4.3 characterizes various aspects of the consumption bottleneck and presents a contention model for analyzing the benefits from more consumption channels. The impacts of routing algorithm, virtual channels, topology, messaging overheads, and communication traffic on demanded number of consumption channels are analyzed in Sections 4.4 and 3.2. Section 4.6 presents simulation experiments and results to derive the effective number of consumption channels under different system parameters.

### 4.1 Routing alternatives

We use wormhole routing switching technique in the inter-cluster network because it offers low-latency point-to-point communication with reduced hardware overhead [24].
Routing within bus/star cluster is straight-forward. For example, on a bus-based cluster, the cluster-bus is used to communicate among the cluster processors. In the star-organization all intra-cluster communication is forced through the cluster interface. For intra-cluster routing within a MIN or direct-network cluster, wormhole routing again offers an attractive alternative. Although various deterministic and adaptive routing algorithms using wormhole-routing techniques have been proposed in the literature, these works have focused on one-level k-ary n-cube architectures only. However, with wormhole-routing being used on both levels of a clustered system the earlier framework can no longer guarantee a deadlock-free routing. In this section we develop a framework of composite-routing of messages in two-level direct networks \((k, n, k1, n1)\) in a deadlock-free manner [9, 10]. Let us first analyze two routing strategies possible for transferring messages from/to cluster and the inter-cluster network across cluster interface (CI).

### 4.1.1 Absorb & retransmit (AB) Strategy

This strategy uses a message forwarding mechanism at the CI. An inter-cluster message starting from a source processor travels to the source-CI using intranet routing restrictions. At the source-CI, the message is fully absorbed and then retransmitted into the internet. The message now travels from the source-CI to the destination-CI using internet routing strategy. At the destination-CI, the message is again fully absorbed before being retransmitted to the destination node using intranet routing restrictions. This strategy is typically used with bus/star cluster organization but can be used with any other intra-cluster organization. The advantage of this strategy
is that it facilitates advanced operations like message combining at the interfaces. A disadvantage with this strategy is that messages incur longer latencies, because they need to be absorbed fully at two intermediate places. For longer messages, the latency becomes more critical. This leads us to the following alternative strategy for direct-network clusters.

4.1.2 No absorb at interface (NAB)

In this strategy, once a message (worm) reaches the source-CL, it continues into the internet. When the worm reaches the destination-CL, it continues into the cluster towards the destination. The main advantage of this technique is that it can lead to lower message latency. However, the rate of message travel in this scheme is determined by the slower of the two networks (the speed is determined by the slowest stage of the pipelined worm movement). When channel cycle times are similar in both hierarchies, the network with smaller flit size determines the speed of the message. This leads to flit-mismatch problem in the system.

In [9], we have suggested the use of virtual channels[24] with demand multiplexing, along with a concatenation scheme, to alleviate this flit mismatch problem. Both AB and NAB are general routing strategies. Actual routing algorithms can be built on top of it.

However, a straightforward implementation of NAB strategy with composite-routing can be deadlock-prone. Message worms can simultaneously occupy channels in more than one network. We show an example of a deadlock and then provide a
design deadlock-free routing schemes for \((k, n, k1, n1)\) systems\([10]\) by adding just one extra virtual channel for each intra-cluster physical link.

**Deadlock-free Routing**

Consider a \((2, 1, 5, 1)\) system using dimension-order minimal routing in both networks as shown in Fig. 21. The two clusters are basically bidirectional rings. Consider a situation where all processors, except processors 4 and 9, are trying to send messages to their respective 2-hop away right cyclic neighbor in the cluster. Processors 4 and 9 on the other hand are trying to send inter-cluster messages to processors 6 and 1, respectively. There are two virtual channels per physical channel and the algorithm dictates that a message can travel on virtual channel 2 in a dimension, iff it is/has taken the wrap-around connection in that dimension. Otherwise virtual channel 1 is used. Though this dimension-order routing is proven to be deadlock-free in a flat(non-clustered) system, it is not so in a \((k, n, k1, n1)\) system. Fig. 21(b) shows a deadlock situation with the specified communication pattern.

The reason why such compositions of deadlock-free routing schemes do not work is because they assume that when a worm reaches the local destination(some CI) in a network, it gets absorbed fully. However in a clustered system using NAB scheme, the worm is not absorbed but released into the next network where it may get blocked. This violates the basic assumption. Since efficient algorithms already exist for handling routing in \(k\)-ary\( n\)-cube networks, it is desirable to come up with a scheme for composite-routing in \((k, n, k1, n1)\) systems that use these algorithms.

We provide a solution to design deadlock-free routing schemes for \((k, n, k1, n1)\)
A (2,1,5,1) system with 2 clusters of 5 processors each. Each of the 2 clusters are bidirectional rings with 2 virtual channels on each physical channel. A message uses virtual channel 1 if it has not taken wrap around, otherwise it uses virtual channel 2.

Assume that the following messages each of greater than 2 flits get initiated at same time: intra-cluster msgs: 0->2, 1->3, 2->4, 3->6, 5->7, 6->8, 7->9, 8->5.
inter-cluster msgs: 4->6, 6->1.

A label on a channel depicts the source id of the message occupying the channel.

Figure 21: An example of a deadlock in a (2,1,5,1) system using dimension order routing with 2 virtual channels a) system connectivity with its virtual channels and b) channels occupied by messages leading to a deadlock.
systems, requiring just one extra virtual channel for each intranet physical link, while retaining the simplicity and flexibility of composite-routing scheme. Consider a composite-routing scheme which uses a routing algorithm \( R_{\text{intra}} \) inside each cluster and \( R_{\text{inter}} \) in the internet. Let \( R_{\text{intra}} \) require \( v \) virtual channels per physical link. We use \( R_{\text{intra}} \) together with its \( v \) virtual channels to route only intra-cluster messages. It is to be noted that such scheme is different from the example in Fig. 21, where \( R_{\text{intra}} \) was also used for routing inter-cluster messages. For routing inter-cluster messages when they are traveling inside the cluster we use a separate virtual channel network as follows. We add an extra virtual channel on all physical links inside the cluster, thus requiring \( v + 1 \) virtual channels per physical link in total. These set of extra virtual channels form a \( k1 \)-ary \( n1 \)-cube virtual network, say \( S \). Any minimal routing strategy \( R \) can be used on \( S \) to route inter-cluster messages inside the cluster. We identify this composite scheme as \textit{modified Composite-routing}.

The inter-cluster messages using network \( S \) of a cluster, can be divided into exactly two classes, A) messages coming from other clusters, B) messages going out of the cluster to other clusters. Given this we can state the following:

**Lemma 1** The set of channels used by messages of class A and class B in the virtual channel network \( S \) are disjoint.

**Proof by contradiction:** Assume two messages, \( m_a \) from class A and \( m_b \) from class B, share a channel under the routing strategy \( R \). It can be shown easily that the path used by one of the messages is non-minimal, and a minimal path can be established without using the shared channel.
This leads us to:

**Corollary 1** *Messages belonging to the two different classes cannot get into a deadlock inside a cluster.*

![Diagram](image)

Figure 22: Partition of the added virtual channel network under the modified-composite-routing scheme, to route messages entering the cluster interface to reach a destination inside the cluster. The routing is minimal. The partitioned virtual channel network is acyclic.

From Lemma 1 we know that the set of channels of $S$ used by class $A$ messages, say $S^A$, is disjoint from the set of messages used by class $B$ messages, say $S^B$. Thus we can partition the channels in $S$ into $S^A$ and $S^B$. It is easy to see that $S^B$ contains exactly the reverse of the channels in $S^A$. Figure 22 shows $S^A$ in a 4-ary 2-cube cluster.
Let $v_{c_{d,x_1,x_2}}$ represent a set of virtual channels of $S$ in dimension $d$ which point from position $x_1$ to $x_2$ in $d$. Here $d \in \{0 \cdots n_1 - 1\}$ and $x_1, x_2 \in \{0 \cdots k_1 - 1\}$. It is easy to see that either all links in set $v_{c_{d,x_1,x_2}}$ belong to $S^A$ or no link in $v_{c_{d,x_1,x_2}}$ belong to $S^A$. Also if $v_{c_{d,x_1,x_2}}$ belongs to $S^A$ then the reverse channel set $v_{c_{d,x_2,x_1}}$ cannot belong to $S^A$. This leads us to the fact that $S^A$ is acyclic(this can be also observed in Fig. 22). Similarly $S^B$ can also be shown to acyclic. This leads us to:

**Lemma 2** An inter-cluster message after entering the cluster and while being routed to its destination inside the cluster, cannot get permanently blocked on a virtual channel of $S$.

**Theorem 2** The modified composite-routing scheme, using an added virtual channel per physical channel of the intranet, is deadlock-free for a two-level clustered system.

**Proof:** Intra-cluster messages travel inside each cluster using deadlock-free $R_{\text{intra}}$ routing. Since they use an independent virtual network, it is ensured that they cannot get into a deadlock. The deadlock-free inter-cluster routing algorithm $R_{\text{inter}}$ ensures that a message injected into the internet at a source cluster interface is bound to reach the destination cluster interface. By Lemma 2 inter-cluster messages entering a cluster cannot get permanently blocked inside the cluster. Using the fact that inter-cluster messages going out of the cluster cannot get into a deadlock, they are guaranteed to reach the cluster interface. All these facts together imply that inter-cluster messages also cannot deadlock. ■
4.2 Designing efficient consumption channel interface

Multicomputers with direct networks supporting wormhole switching have become a popular trend in building large parallel systems. Most commonly used direct network topologies are meshes/tori belonging to the class of generalized $k$-ary $n$-cube interconnections\[52\]. These are $n$-dimensional grid structures with $k_0, k_1, \cdots, k_{n-1}$ nodes along dimensions $0, 1, \cdots, n - 1$, respectively. The values of $k_0, k_1, \cdots, k_{n-1}$ may be same or different. Figure 23 shows a two dimensional 3x3 mesh (a 3-ary 2-cube) system. Each node in the system comprises of a router and a processor. The routers are interconnected as a 3x3 mesh. Each processor is connected to its router so that it can inject messages to other nodes and consume messages destined to it. In this study we assume the inter-processor messages to traverse across the interconnection network using wormhole-routing\[24\]. In such routing a message is transferred as a train of flow control digits (flits). A flit is typically the size of the network channel width. Only the header flit has routing information. Each router only dedicates a few flit buffers for a message. With the header getting blocked due to contention, the remaining flits in the network are blocked in place. Many current systems built in industry and academia, like the Cray T3D\[23\], Intel Paragon\[42\], and Stanford FLASH\[36\] employ such wormhole-switched mesh/torus interconnection networks with 2D/3D organization.
Figure 23: A two-dimensional mesh of size 3x3 (3-ary 2-cube). A typical router-processor interface is illustrated for a $k$-ary $n$-cube system with one consumption channel per node.

### 4.2.1 Typical interface with single consumption channel

Let us analyze the typical router processor interface as shown in Fig. 23. A router can receive incoming messages in either direction on any of the $n$ dimensions leading to $2n$ incoming physical channels into the router. Similarly a message may need to be routed on any of the $2n$ outgoing physical channels from the router. Depending on the routing scheme and virtual channel flow control used, a set of $v$ virtual channels may be demand-multiplexed on each physical channel. Besides these network channels, a single 	extit{consumption} channel is used for the local processor to consume messages, destined to it, from the network. Similarly, a single 	extit{injection} channel is used for the local processor to inject messages into the network. With a single consumption channel, a processor is constrained to consuming only one message at a time. For example, it can be observed in Fig. 23 that while a message (shown in gray) is being
consumed, another message (shown in black) destined to be also consumed at the same node, is forced to wait. Let us consider the process of message consumption in more detail. A message being consumed is removed from the network along a consumption channel, stored temporarily at the interface in a message buffer, and then transferred over a local bus [34] to the processor-memory. In the past programmed I/O was used [33] to transfer bytes of a message from the interface to the processor-memory. Clearly, this leads to low communication bandwidth with the processor becoming the bottleneck. It also results in lack of overlap between computation and communication. In recent systems, DMA-based message consumption/injection) is used [37] for transferring messages between the interface and processor-memory.

As shown in Fig. 23, the local processor-memory bus bandwidth is shared between DMA operations to write consumed messages, messages to be injected, memory read/write operations due to processor-cache subsystem, and other I/O. Thus, only a fraction of the bus bandwidth may be available to DMA for writing consumed messages from interface to processor-memory (I-to-P). Let $B_{mem}$ denote such available I-to-P bandwidth. The buffering of messages at the interface helps to isolate the network from the vagaries of the processor-memory bus and vice-versa. With the processor-memory bus being busy with other operations, messages can still be removed from the network and buffered at the interface. The number of message buffers at the interface depends on the message arrival rate, message length, and the available I-to-P bandwidth. To have smaller buffers, a limit is typically put on the message transfer size in the network. Larger messages are divided into such smaller
messages packets before being transferred over the network. For example, on the Cray-T3D and IBM SP2 such transfer size on the network is approximately 64 and 256 bytes [23, 34], respectively. Earlier research[1, 71] has demonstrated that providing a small amount of memory, corresponding to a few message packet buffers, is sufficient to approximate infinite buffering performance. In the eventuality that all message packet buffers are full, further consumption is stalled and incoming messages forced to block in the network. From here on, in this paper we refer to a message packet as simply a message.

4.2.2 Consumption collision and domino-blocking effect

A single consumption channel can be a bottleneck to system performance. We illustrate such bottleneck through an example and demonstrate the alleviating impact of multiple consumption channels on performance. The following definitions are useful in the following discussion:

Definition 4 A message whose header flit has reached its destination node is defined as a terminating message.

Definition 5 An event when a terminating message gets blocked in the network due to unavailability of consumption channels is defined as a consumption collision.

Definition 6 The increase in latency of a terminating message due to consumption collision is defined as consumption delay.

Figure 24 shows an example of two terminating messages M1 and M2 arriving simultaneously at destination node $D_1$ in a 2D mesh along the $+y$ and $-y$ links. Let
node $D_1$ have exactly one consumption channel. A single consumption channel forces these two messages to be consumed one at a time. Let the consumption channel service $M_1$ first with a service time denoted by $T_c$. Thus, $M_2$ encounters a consumption collision and a consumption delay of $T_c$. Wormhole routing forces the flits of $M_2$ to be held in place while holding onto network channels. As secondary impact, a third message, $M_3$, destined to a different node gets blocked on a network channel held by $M_2$. Although not shown in the figure we can consider a fourth message being blocked on $M_3$, and so on. We refer to such an event as a domino-blocking effect. Such blocking leads to higher average message latency and lower message throughput due to wastage of network resources. The probability of domino-blocking effect increases with network load. Having more consumption channels (two for this example) can reduce (or eliminate) consumption delay leading to alleviating of the domino-blocking effect. This leads us to the following observation:

**Observation 3** With more consumption channels the possibility of a consumption collision is reduced. This has two impacts: a direct impact in lowering the latency of the message being consumed and an indirect impact on the gain in overall network performance (throughput) by alleviating the wasteful effects of domino-blocking.

### 4.2.3 Design with multiple consumption channels

Based on the above observation, in this paper, we propose supporting multiple consumption channels at a node to alleviate the consumption bottleneck. Let $C$ denote the number of consumption channels (also referred to as physical consumption
Figure 24: An example of consumption collision in a 5x5 mesh with one consumption channel under dimension-order (x-y) routing. Two messages M1 and M2 are destined towards the node D1. A third message M3 destined to D2 is blocked on M2.

channels) being supported at a node. Figure 25 shows a router-processor interface supporting multiple physical consumption channels per node.

Figure 25: A router-processor interface supporting multiple consumption channels per node.

As discussed earlier, a naive solution to alleviate the consumption bottleneck is to provide 2n physical consumption channels in an n-dimensional system, to handle the worst case message arrival. However, this naive solution may be an overkill and may not be realizable due to limited router pinout constraint, as discussed in more detail in Sec. 4.2.5. In most cases a much lesser number may be sufficient.
to deliver best system performance. The required number of consumption channels depends on the complex interplay of various factors including the traffic pattern, routing adaptivity, interconnection dimensionality, network bisection size, messaging overheads, and degree of virtual channel multiplexing on network channels. Our objective is to analyze this interplay and derive the minimum number of required consumption channels for best performance. The following questions are addressed in this paper:

- **When does having more than one physical consumption channel lead to better performance?**

- **What are the factors contributing towards more demand for consumption channels?**

- **What is the effective number of consumption channels, beyond which returned increase in gained performance does not merit more channels?**

The above issues are discussed in more detail in the following sections. In this paper, we focus on the design of consumption channel interface. It is to be noted that a message in a wormhole network may also get blocked at its source node due to limited number of injection channels [52]. Under high network load, a message may hold onto an injection channel longer because of its slower propagation due to network contention. In such a situation it seems that having additional injection channels may help to inject waiting messages out of the node. However, this may be of limited use because an injected message can get blocked on a busy network channel. Thus,
providing more injection capacity may not lead to better system performance. This indicates that the injection channel bottleneck is less critical than the consumption channel bottleneck. However, to be fair to the injection side and to derive balanced design guidelines we assume the number of injection channels at a node being equal to the number of consumption channels.

4.2.4 Supporting virtual consumption channels

To avoid the wasteful effects of consumption channel bandwidth caused by message slowdown due to virtual multiplexing in the network, a good design must provide matching number of virtual consumption channels on each physical consumption channel. Let us consider an enhanced interface design to support multiple physical consumption channels, each supporting virtual channels to match degree of network channel multiplexing.

Design implementation of virtual channels has been studied in detail in [25]. Assuming routers already supporting \( v \) virtual channels on each network channel it is straight-forward to extend such design to also support \( v \) virtual consumption channels on each physical consumption channel as shown in Fig. 26. Each virtual consumption channel has message buffers associated with it. These serve the same purpose of intermediate buffering as the message buffers shown earlier in Fig. 23. With multiple virtual channels per physical channel, messages along a virtual channel come at a slower rate. Thus, the buffering required per virtual channel need not be very large. Based on preliminary experimentation, in this study we associate only two buffers per virtual consumption channel. While one buffer is full and waiting to be written
Figure 26: Supporting multiple virtual consumption channels on a physical consumption channel. On the router side, the virtual channel implementation on a router network channel is extended to a consumption channel. On the interface side, each virtual consumption channel has message buffers associated with it.

to processor-memory by DMA, the other buffer is used to gather the next message along the virtual channel. Experimentation with higher number of message buffers yielded similar performance.

4.2.5 Number of physical consumption channels supportable by router pinout

Current technology supports up to 250 to 400 pins [8] per router. The pinout restricts the number of channels and channel width that can be supported from a router[3]. Although a very high number of consumption channels is clearly not possible, let us derive an estimate on reasonable number of consumption channels supportable by contemporary router pinout technology. Let $R$ denote the required router pinout, in a $k$-ary $n$-cube system with $W$ bytes wide data channels. In a system with bidirectional
channels the number of network channels is $4n$. For supporting $C$ consumption channels per router and assuming equal number of injection channels, the total number of channels to be supported per router is derived as $4n + 2C$. As discussed earlier, let $v$ be the degree of virtual multiplexing in the system. The number of control bits required per physical channel for identifying the virtual channel being send or acknowledged is $2\log_2 v$ bits[25]. For a system with $W = 2$ bytes (16 bits) and $v = 4$, 20 bits are required per channel, leading to $R = 20(4n + 2C)$. Thus, a router pinout of 400 can support up to $C = 4$ in a 3D network ($n = 3$) ($4n + 2C = 20$). For 2D network, thinner data channels or fewer virtual channels, a higher $C$ can be supported. This leads us to conclude that it is reasonable to expect contemporary router pinout technology to support up to 4 physical consumption channel per router.

In the following sections we analyze various factors affecting the effective number of consumption channels required for alleviating the bottleneck.

### 4.3 Analytical model for estimating consumption collision delay?

In this section we present a contention model to estimate the probability of consumption collision and consumption delay at a node for a given number of incoming network channels, message rates on these channels, and number of physical consumption channels. Our model is based on an analysis by Dally for estimating wormhole contention delay in [72]. To keep the model simple we do not consider the impact of virtual channels here. Such impact is discussed in a later section. Trends based on the model are validated with accurate simulation modeling in Sec. 4.6.
Let us consider a node with \( m \) incoming network channels into a router with a) one consumption channel and b) multiple consumption channels, as shown in Fig. 27.

For convenience, the following is a summary of notation used in this analysis.

- \( m \): The number of incoming network channels.
- \( C \): The number of physical consumption channels at each node.
- \( \lambda_i \): The rate of arrival of terminating messages along incoming network channel \( i \) in messages/cycle.
- \( \lambda \): Total message arrival rate at a node in messages/cycle.
- \( L \): Message length in bytes.
- \( W \): Channel width in bytes.
- \( T_c \): Service time seen by a message at a consumption channel.
- \( d_i \): Duty factor of a network channel with respect to terminating messages.
- \( p_i \): Probability of a terminating message along channel \( i \) encountering a consumption collision.
- \( \Delta T_i \): Expected consumption delay seen by a terminating message along channel \( i \).
- \( \Delta T \): Expected consumption delay seen by a terminating message along any channel. This is a weighted average of delays along all incoming channels.

We make the following assumptions in our model:

1) All message packets are of length \( L \).

2) All consumption channels have the same message service rate, \( T_c = L/W \) cycles.

Consider an incoming network channel \( i \) with arrival rate of terminating messages, \( \lambda_i \). An arriving terminating message reserves a consumption channel and is consumed in \( T_c \) cycles. Thus, the incoming network channel is busy servicing a terminating
message for a period of $T_c$ every $1/\lambda_i$ cycles, leading to a duty factor of $d_i = \lambda_i T_c$. Clearly, by definition $d_i \leq 1.0$. We denote $\lambda = \sum_{j=1}^{m} \lambda_j$ as the total terminating message rate at a node. The total bandwidth along $C$ consumption channels imposes the constraint $\lambda \leq C/T_c$. In a later section we demonstrate the constraining impact of other factors on the message arrival rate at a node.

4.3.1 Estimating probability of consumption collision

Let us compute, $p_i$, the probability of a terminating message along channel $i$ encountering a collision delay. We first compute $p_i$ for a node having one consumption channel, denoted by $p_i^1$, and then generalize it to $C > 1$ consumption channels, denoted by $p_i^C$. The probability of collision is the single consumption channel being busy handling a message along another channel. Based on the analysis in [72] the probability of such an event is the total duty factor on other channels, leading to:

$$p_i^1 = d_1 + \cdots + d_{i-1} + d_{i+1} + \cdots + d_i$$

$$= T_c(\lambda_1 + \cdots + \lambda_{i-1} + \lambda_{i+1} + \cdots + \lambda_m). \quad (4.1)$$

In general with $C$ consumption channels, the probability of a consumption collision along channel $i$ is that of all consumption channels being busy with messages along other channels. With $C = m$, every incoming channel has a separate consumption channel leading to $\forall_i(p_i^m = 0)$. For $C < m$, $p_i$ is the summation of probabilities of the $(C-1)$ possible ways a message along channel $i$ can collide with messages along the other $m-1$ network channels on $C$ consumption channels. This probability is
derived as:

\[ p_i^C = \Sigma d_j, d_j \cdots d_{j_c} \]

\[ = (T_c)^c \Sigma \lambda_j, \lambda_{j_2} \cdots \lambda_{j_c}, \]  

(4.2)

where the summation is over all terms obtained as \( \forall q=1 \) \((j_q\in[1,m]-i)\) and any such term \( \lambda_j, \lambda_{j_2} \cdots \lambda_{j_c} \) has \( j_{q_1} \neq j_{q_2} \) for \( q_1 \neq q_2 \). Equation 4.1 can be observed to be a special case of the above equation obtained with \( C = 1 \). By comparing \( p_i^C \) for different values of \( C \) leads us to:

**Observation 4** For a given set of message arrival rates along \( m \) channels, \((\lambda_1, \lambda_2, \cdots, \lambda_m)\), the probability of consumption collision along any channel \( i \) \( (p_i^C) \) reduces as the number of consumption channels, \( C \), is increased from 1 to \( m \). For \( C \geq m \), \( \forall_i(p_i = 0) \).

This observation supports the intuition that having more consumption channels reduces consumption contention. This leads to lower expected collision delay as shown below.

**4.3.2 Estimating collision delay**

Given that we have a collision, let us compute the expected collision delay along a channel \( i \). The waiting time has a uniform distribution from 0 to \( T_c \). With no queuing at the switch, the average waiting time is half the consumption service time, \( T_c/2 \). Thus, expected consumption collision delay seen by a terminating message along channel \( i \), \( \Delta T_i \), is:

\[ \Delta T_i = p_i^C T_c/2 \]  

(4.3)
The collision delay seen by a message entering this node to be consumed, denoted by $\Delta T$, is given as a weighted average of delays along all channels:

$$\Delta T = \frac{1}{\lambda} \sum_{j=1}^{C} (\Delta T_j \lambda_j)$$

$$= \frac{1}{\lambda} T_c / 2 \left( \sum_{j=1}^{C} p_j^C \lambda_j \right)$$

(4.4)

A higher value of collision delay leads to greater chance of other messages in the network blocking on resources held by consumption blocked message, aggravating the impact of a domino-blocking effect. Based on Observation 4 and the above expression we conclude that higher number of consumption channels reduce consumption delay and hence alleviate the problems from the domino-blocking effect leading to higher network performance.

### 4.3.3 Parameters affecting collision delay and benefits from more consumption channels

Using the expression for $p_i^C$ derived earlier in Eqn. 4.2 and applying it to Eqn. 4.4 leads us to the following expression for $\Delta T$ in a node with $C$ consumption channels:

$$\Delta T = \frac{1}{\lambda} (T_c / 2)(T_c)^C (C + 1) (\sum_{j=1}^{C} \lambda_j \lambda_{j_2} \cdots \lambda_{j_{C+1}}).$$

(4.5)

where the summation is over all terms obtained as $\forall_{q=1}^{C+1} (j_q \in [1, m])$ and any such term $\lambda_{j_1} \lambda_{j_2} \cdots \lambda_{j_{C+1}}$ has $j_{q_1} \neq j_{q_2}$ for $q_1 \neq q_2$. Eqn. 4.5 leads us to:

**Observation 5** Given a fixed total terminating message rate at a node ($\lambda = \sum_{j=1}^{m} \lambda_j$), the expected collision delay, $\Delta T$, is minimized when $\lambda_j = \lambda$ for some $j = i$ and $\lambda_j = 0$ for $j \neq i$. Similarly, $\Delta T$ is maximized when message rates are balanced, $\forall_{i=1}^{m} \lambda_i = \lambda/m$. 
Thus, the maximum collision delay, denoted by $\Delta T_{\text{max}}$, for given $C$, $T_e$, $m$, and $\lambda$ (assuming balanced rate on channels), is derived as:

$$\Delta T_{\text{max}} = \left( \frac{T_e \lambda}{m} \right)^{C+1}(C + 1)\binom{m}{C+1}/(2\lambda)$$  \hspace{1cm} (4.6)

The parameter $\Delta T_{\text{max}}$ being a function of $C$, $\lambda$, $m$, and $T_e$, can be expressed as $\Delta T_{\text{max}}(C, \lambda, m, T_e)$. For $C \leq m$, it can be shown that the expression $(\Delta T_{\text{max}}(C - 1, \lambda, m, T_e) - \Delta T_{\text{max}}(C, \lambda, m, T_e))$ is an increasing function with respect to $\lambda$, $m$, and $T_e$. Thus, at higher values of these parameters the decrease in consumption delay with more consumption channels is expected to be higher. Based on the above discussion we make the following summary on expected trends on benefits from more consumption channels:

**Observation 6** *Benefits from more consumption channels at a node is significant at higher values of collision delay. Collision delay is higher with:*

1. *higher message rate into a node,*
2. *message rates being balanced along incoming channels,*
3. *higher network dimensionality, ($m = 2n$), and*
4. *longer messages ($T_e = L/W$).*

In the following section we consider the impact of the above factors in more detail.
4.4 Applying the model to determine required number of physical consumption channels

In this section we consider the impact of balanced versus unbalanced message rates, longer message length, and system dimensionality on the benefits from more consumption channels.

4.4.1 Impact of routing - balanced versus unbalanced rates

The routing algorithm used to route messages in a network can lead to messages arriving at a node along different incoming channels in a balanced or unbalanced manner. This can affect the demanded number of consumption channels at a node. Let us first consider dimension-order routing as discussed in the example in Fig. 24. It routes messages on a minimal path in order of increasing dimensions. For a given destination node, let us consider the directions along which messages from other nodes enter here as determined by routing restrictions. The $2n$ directions are identified as $\pm i$, $0 \leq i < n$. By considering all cases it can be shown in a $k$-ary $n$-cube system, the number of messages entering the destination node along direction $+i$ is $([\frac{k-1}{2}]k^i)$, while $([\frac{k-1}{2}]k^i)$ messages enter along direction $-i$. The total possible messages being $(k^n - 1)$, the relative message arrival rate along direction $+i$ is derived as $([\frac{k-1}{2}]k^i)/(k^n - 1) \approx [\frac{k-1}{2}]/k^{n-i}$. Thus, for a given total message termination rate at a node being $\lambda$, the message rate along direction $+i$, denoted here as $\lambda_{+i}$, is obtained as:

$$\lambda_{+i} = \lambda([\frac{k-1}{2}]k^{n-i}).$$ (4.7)
Similarly, message rates can be obtained along other dimensions: $0 \leq i < n$. From the above equation, a sharp imbalance can be observed in the message rates along different dimensions. Most messages enter along the highest dimensions: $\pm(n - 1)$.

A more adaptive routing is expected to lead to more balanced message arrival rates at a node. In this case, for a given total message termination rate at a node being $\lambda$, the message rate along direction $+i$ is obtained as:

$$\lambda_{+i} = \lambda / 2n. \quad (4.8)$$

Figure 28 depicts the impact of having more consumption channels on expected collision delay. The plots show collision delay, $(\Delta T)$, versus message rate, $(\lambda)$, at a node in a 16x16 (2D) system. Figure 28(a) is derived based on Eqn. 4.5 and message rates from dimension-order routing as in Eqn. 4.7. Similarly, Fig. 28(b) is derived based on Eqns. 4.5 and message rates from a more adaptive routing as in Eqn. 4.8. A message length of 64 bytes and channel width of 2 bytes leading to $T_c = 32$ cycles is assumed. The maximum allowable message rate is derived by ensuring the duty cycle of any channel not to exceed 1. From the plots it can be observed that for maintaining a small average collision delay (let us consider 2 cycles), the number of consumption channels required increases with message rate. By comparing the plots in Fig. 28(a) and (b) we observe that more consumption channels offer higher benefits with balanced rates. This leads us to the following observation:

**Observation 7** Routing algorithms with adaptivity, offering balanced message rates along incoming channels, can gain from higher number of consumption channels than non-adaptive (dimension-order) routing.
In Sec. 4.6 we compare fully adaptive [31] routing against dimension-order to determine the impact of routing on the effective number of consumption channels.

4.4.2 Impact of message length

Figure 29 depicts the impact of more consumption channels in a 16x16 system with two different message length of 64 and 256 bytes, respectively. It can be observed that with the longer message length ($L = 256$ bytes), values of average collision delay are much higher. For example, with $L = 64$ bytes and a message arrival rate of 0.02 messages/cycle (or 1.28 bytes/cycle) the collision delay is 6 cycles with $C = 1$. For $L = 256$ bytes, a similar arrival rate of 1.28 bytes/cycle occurs at 0.005 messages/cycle (obtained as 1.28/256). With $C = 1$ the collision delay at this message rate is much higher: 22 cycles. Similarly, it can be observed that the drop in collision delay with another consumption channel ($C = 2$) is also much more significant with $L = 256$ (20 cycles as compared to 5 with $L = 64$). This indicates that a system with longer messages can gain more from higher number of channels.
4.4.3 Impact of network dimensionality

Figure 30 depicts the impact of more consumption channels in two systems with different dimensionality. The plots are derived based on Eqn. 4.5 and are shown separately for a node in a 2D and 3D network system, respectively. As discussed earlier, an n-dimensional system requires at most $2^n$ physical consumption channels. Thus, a 2D system can use up to four consumption channels, while the 3D system can use up to six. At and beyond these respective number of consumption channels, the expected collision delay drops to zero based on Obs. 4. From the plots it can be observed that for the same message rate, the decrease in consumption delay with more consumption channels is higher in the 3D network, indicating higher dimensional systems can gain more from higher number of channels.

From the above discussion, it is clear that two major factors dictate the expected benefits from more consumption channels. These are: a) maximum possible message rate in a system and b) the balance among the arrival rates along different incoming
channels. The maximum message rate or throughput\(^2\) in a system is bounded by other constraints such as network bisection size, sustained available bandwidth between interface and processor, messaging overheads for sending/receiving messages, traffic patterns, and virtual channel flow control.

4.5 Constraints affecting maximum message arrival rate and number of physical consumption channels

In this section we analyze the impact of the factors mentioned above on the maximum message arrival rate at a node.

4.5.1 Network bisection

A small network bisection limits the maximal achievable system performance under uniform traffic. Let \(B\) denote the bisection size of a network. It is the minimum number of network channels to be cut so as to divide the network into two equal

\(^2\)We have used the terms message rate and throughput interchangeably.
halves. The bisection size of a $k$-ary $n$-cube network is derived\cite{24} as,$^3$

$$B = 2^{k^n-1} \text{channels} = 2^{k^n-1}W \text{bytes}$$  \hspace{0.5cm} (4.9)

where $W$, as defined earlier, denotes the channel width in bytes. Thus, at most $B = 2^{k^n-1}W$ bytes can be sent across the network bisection per cycle. For messages of length $L$ bytes the same bandwidth translates to $B/L = 2^{k^n-1}W/L$ messages/cycle.

Under uniform traffic, half the generated messages in the system need to cross the network bisection. Thus, for a message sending rate of $\lambda$ messages/cycle at each of the $k^n$ nodes in the system, the number of messages traveling across the bisection every cycle is derived as $k^n\lambda/2$. Since this has to be less than the bisection size, it leads to $((k^n\lambda/2) \leq (2^{k^n-1}W/L))$. The maximum message rate from bisection constraint, denoted by $\lambda_B$ is:

$$\lambda_B \leq (4W)/(kL) \text{ messages/cycle}$$  \hspace{0.5cm} (4.10)

This leads us to the following observation:

**Observation 8** Keeping channel width ($W$) and message length ($L$) fixed in a $k$-ary $n$-cube system, increasing network size by using larger radix ($k$) leads to a full in maximum message rate ($\lambda_B$) supportable by network bisection. Depending on the system size ($k$) it may happen that in some configurations $\lambda_B$ is constrained to be very small. This leads to probability of consumption collision also being small. In such case more consumption channels may not offer significant benefit in performance.

$^3$For toruses this number needs to be adjusted by a factor of two to $B = 4k^n-1$ channels due to the presence of wrap-around links.
It should be noted that the parameter $\lambda_B$ is only an upper limit on the maximum achievable throughput in a system. The actual maximum achievable throughput is typically lower due to network contention. Similar observations continue to hold with such throughput also [3, 24]. Network contention depends on various factors including virtual channel flow control, routing adaptivity, and message length. In Sec. 4.6 we study these aspects in detail through simulation experiments. We demonstrate that in contemporary systems, even up to 256 processors, the achievable value of throughput is high enough to merit more consumption channels.

4.5.2 Non-uniform traffic

In a non-uniform traffic scenario, hot-spots occur when some nodes in the system receive more than their fair-share of messages. Let us characterize non-uniform traffic by choosing a fraction, $f_h$, of the nodes ($N$) in the system randomly as hot nodes. Each processor in the system generates messages at a given rate and the messages are directed towards a hot node with a probability controlled by a parameter, $d$, defining the degree of hotness of traffic. The parameter $d$ can reflect various degrees of hot traffic. A value of $d = 1.0$ corresponds to uniform traffic, while higher values ($d > 1.0$) correspond to non-uniform traffic.

As the traffic becomes more non-uniform (increasing $d$), the network bisection becomes lesser constraint on achievable non-uniform throughput. Thus, at the hot nodes, message arrival rate much greater than $\lambda_B$ can be expected. In such a situation, the number of incoming network channels into a hot node becomes the dominant

---

4The terms non-uniform traffic and hot-spot traffic are used interchangeably.
constraint on performance. With higher message rate, the number of consumption channels required to alleviate the consumption bottleneck is expected to be higher.

This leads us to the following observation:

**Observation 9** Higher number of consumption channels are expected to be useful in alleviating the consumption bottleneck under hot-spot traffic than under uniform traffic.

### 4.5.3 Adaptivity and virtual multiplexing on network channels

In Sec. 4.3 we observed one impact of adaptive routing in offering more balanced message arrival rates. The other impact of adaptivity is in offering higher network throughput [4, 19] by allowing a message to circumvent points of congestion. Similar impact is also achieved with virtual channel flow control (VCFC) [25] which allows an active message to bypass blocked messages on a channel leading to lower blocking and better network utilization. This leads to the following observation:

**Observation 10** Higher adaptivity and virtual channels lead to higher network throughput and higher maximum message rate. For sustaining such higher message rate at the consumption interface, more consumption channels can be beneficial.

### 4.5.4 Messaging overheads

In point-to-point communication a message is associated with *processing overheads* from procedure calls, protection checking, checksum computing, and possibly memory copying. Let $t$, cycles/message denote such overhead. The maximum message
handling rate from such overhead, denoted by $\lambda_{sw}$, is derived as:

$$\lambda_{sw} = 1/t_s \text{ messages/cycle.} \quad (4.11)$$

Such overheads, usually in the range of few hundreds to thousands of cycles [30], can severely limit the rate at which messages can be sent or received from a node [44]. For example, a messaging overhead of 500 cycles limits maximum message rate per node to 0.002 messages/cycle. This can lead to much of the communication bandwidth offered by the expensive fast interconnect to remain unutilized. With low maximum message rate per node, the probability of consumption delay can be small leading to limited benefits from higher number of consumption channels. Several research studies are ongoing [45, 32, 44, 17] for designing messaging protocols and hardware to reduce such overheads. In the near future we expect efficient protocols to offer $t_s$ in the range of few tens to hundreds of cycles. Similarly, in hardware based implementations of Distributed Shared Memory (DSM) systems with small messages for cache coherence, the value of $t_s$ is expected to be small. In Sec. 4.6 we study the impact of such overhead on the requirement of consumption channels. We demonstrate that even with reasonably high messaging overhead, $t_s = 100$ cycles, more than one consumption channel can yield significant gain in performance.

4.5.5 Interface to processor bandwidth

Each physical consumption channel can remove messages from the network into the interface message buffers at the rate of $B_{phy} = W \text{ bytes/cycle}$, where $W$ is the channel width as defined earlier. With $C$ physical consumption channels this rate increases
proportionally to $WC$ bytes/cycle. However, this may not be the rate that can be sustained at the consumption interface. Messages at the interface buffers are transferred by DMA to the processor memory over the local bus. As discussed earlier in Sec. 4.2, only a fraction of the bus bandwidth (I-to-P bandwidth) is available to DMA operations for writing consumed messages. The parameter $B_{mem}$ bytes/cycle refers to such I-to-P bandwidth for message consumption. A cycle here refers to the network cycle time and not the cycle time of the local bus. Clearly, a small value of $B_{mem}$ can restrict the maximum sustainable message rate in the system. For example, in a system with $B_{mem} = 2$ bytes/cycle (for a cycle time of 0.01$\mu$sec this corresponds to 200 MBytes/sec) and message length 64 bytes, the maximum message rate is 0.03 messages/cycle.

Having limited I-to-P bandwidth also raises an interesting question: *Is there a need to have more than $B_{mem}/B_{phy}$ consumption channels in a system, even if this number is very small?* Let us define the parameter $C_m$ to denote the number $B_{mem}/B_{phy}$. It can be observed that by using only $C_m$ channels, the I-to-P bandwidth can be fully utilized. However, for small value of $C_m$ (1 or 2) the probability of collision on the consumption channels is high leading to high collision delays. High consumption delay leads degradation of performance due to domino-blocking effect. Only in traffic patterns like all-to-one where all messages in the system are destined to a single node, the domino-blocking effect cannot hurt the performance of messages destined to other nodes (since there are no such messages). Under such a pattern having $C_m$ consumption channels is sufficient, even if this number is small (1 or 2). However,
under other traffic patterns like uniform and non-uniform, where domino-blocking effect leads to overall degradation of performance, having more than $C_m$ consumption channels can be beneficial, especially when $C_m$ is a small number like 1 or 2. This leads to:

**Observation 11** For small values of $C_m$, having more than $C_m$ physical consumption channels does not lead to better completion time for an all-to-one traffic pattern. However, the performance of uniform and non-uniform traffic pattern can improve with more than $C_m$ channels.

### 4.5.6 Summary

It can be observed that the need for more than one physical consumption channel is dependent on the interplay of all the above factors: routing algorithm, message length, traffic pattern, network bisection, messaging overhead, interface-to-processor bandwidth, and network dimensionality. With all these factors determining consumption performance, it becomes a challenging problem to determine: *How many physical consumption channels need to be provided per node to derive the best performance?*

Intuitively, with each added channel we expect to get back diminishing returns in terms of gained performance. Thus, we are interested in a minimum or *effective* number of channels beyond which added performance gained (in terms of reduction in latency or increase in throughput) is less than some specified percentage of the performance attained with one consumption channel. Without loss of generality, in this paper we choose this number as 5%.
Earlier studies [3, 24] have proposed contention models to determine latency versus throughput performance in wormhole-switched $k$-ary $n$-cube networks. However, these studies focused mostly on dimension-order routing and assumed no consumption collision at the destination node (*consumption assumption*[3, 24, 25]). In this study our focus is on designing the consumption interface and studying the impact of various factors on such design. The large number of parameters being studied makes it difficult to come up with a tractable integrated contention model for the network together with the consumption interface in a wormhole system. Thus, we establish trends and derive guidelines through simulation by considering the whole gamut of interplay discussed above.

4.6 Simulation experiments and results

In this section we validate the guidelines proposed in earlier sections using simulation and derive effective number of consumption channels required under different network and system parameters like routing, message length, system size, messaging overheads, I-to-P bandwidth, traffic patterns, and degree of virtual multiplexing.

4.6.1 Simulation Set-up

A flit-level wormhole-routed simulator WORMULSim was used to model and evaluate the impact of multiple consumption channels for different topologies, routing, and traffic pattern. This simulator is designed using CSIM [66]. It accurately models flit-level transfer corresponding to multiple virtual channels using demand multiplexing over network and consumption channels. Similarly, DMA operations over the local
bus for transferring messages are also modeled. Without loss of generality, a network cycle time of 0.01 μsec was assumed. Deterministic [26] and fully-adaptive [31] routing schemes with variable number of virtual channels (lanes) were used. Four different system configurations, 8x8, 16x16, 4x4x4, and 6x6x6 were examined. Three different traffic models were considered: uniform traffic, hot-spot traffic, and all-to-one pattern. In each experiment the number of physical consumption channels per node was varied from 1 to 6. Each physical consumption channel supported \( v \) virtual consumption channels, where \( v \) is the degree of multiplexing on network channels. Each simulation experiment was carried out for a sufficient number of runs to ensure that the 95% confidence interval was within ±1% of the mean. Three different values of I-to-P bandwidth corresponding to \( C_m = 1, 2, \) and 8 were used. Impact of message length was observed for two different values of \( L = 64 \) and 256 bytes. Messaging overhead of \( t_{sw} = 50 \) to 200 cycles was examined. A channel width \( W = 2 \) bytes was assumed in these experiments.

4.6.2 Impact of routing adaptivity

Figure 31 shows the latency (microsecs) versus throughput per node (messages/microsec) for dimension-order routing and fully adaptive [31] routing schemes, respectively, on a 4x4x4 system. The top row of plots depict the results for uniform traffic while the bottom row of plots depict results with hot-spot traffic. A message length of 64 bytes was assumed. From Fig. 31 it can be observed that under uniform traffic with dimension-order routing, the maximum achieved throughput with one consumption channel per node \( (C = 1) \) is only 1.2 messages/μsec. It is interesting to note that with
adaptive routing and $C = 1$ exactly the same maximum throughput is attained. This indicates that the expected gains in performance are inhibited by the consumption bottleneck. With more consumption channels ($C = 2$ to 6), increased throughput is attained under both routing schemes. However, with dimension-order, the maximum throughput attained with $C \geq 2$ is limited to 1.7 messages/μsec. This indicates that $C = 2$ is effective for this system. With adaptive routing up to $C = 3$ channels are beneficial leading to much higher maximum throughput of 2.5 messages/μsec. These results validate Obs. 7.

![Figure 31: Impact of routing adaptivity and number of consumption channels on performance in a 4x4x4 system. Both uniform and non-uniform traffic were considered.](image)

Similar trends were also observed with hot-spot traffic as depicted in the bottom row of plots in Fig. 31. The message arrival rates at hot nodes is much higher with hot-traffic (the x-axis in the plots represent message generation rate, the message
arrival rate at hot nodes is \( d \) times higher). As with uniform traffic, with \( C = 1 \), the maximum throughput achieved was same (0.3 messages/\( \mu \text{sec} \)) under both routing schemes indicating consumption bottleneck inhibiting gains from adaptivity. With more consumption channels, the maximum throughput with dimension-order and adaptive routing increased to 0.8 and 1.25, respectively. Such message rates were achieved at \( C = 3 \) and 4 under dimension-order and fully-adaptive routing, respectively. This confirms that higher number of consumption channels are more beneficial with adaptive routing.

From Fig. 31 it can also be observed that for a system with a given routing, more consumption channels are beneficial with hot-spot traffic compared to uniform traffic. This validates Obs. 9.

### 4.6.3 Impact of message length

Figure 32 shows latency versus throughput obtained with uniform traffic on a 4x4x4 system with dimension-order routing. Plots are separately shown for message length of \( L = 64 \) and 256 bytes, respectively. From Fig. 32 it can be observed that with \( L = 64 \), there is no gain in throughput by using \( C = 3 \) over \( C = 2 \) channels. However, with longer message length \( L = 256 \) bytes, there is a gain of 11\% in throughput with \( C = 3 \). This indicates that with higher message length higher number of consumption channels can be beneficial validating Obs. 6. Similar trends were also observed with non-uniform traffic, adaptive routing, and other system configurations. These results are not presented here in order to limit the size of the paper.
Figure 32: Impact of message length and number of consumption channels on uniform traffic performance in a 4x4x4 system. Two different message length of $L = 64$ and 256 bytes, respectively, were considered.

### 4.6.4 Impact of system dimensionality

Figure 33 shows latency versus throughput obtained with with non-uniform traffic on two systems of different dimensionality: 8x8 and 4x4x4. From Fig. 33 it can be observed that $C = 3$ and 4 are effective in the 2D and 3D systems, respectively. This validates Obs. 6 that higher number of consumption channels can be beneficial in higher dimensional system.

Figure 33: Impact of dimensionality and number of consumption channels on system performance. Two different systems 8x8 and 4x4x4 were considered.
4.6.5 Impact of system size with uniform traffic

In Obs. 8 we derived that achievable uniform message rate decreases with system size due to network bisection constraint. Reduced message rates in larger systems can make higher number of consumption channels less beneficial. Figure 34 shows latency versus throughput results with uniform traffic on two 2D system sizes of 8x8 and 16x16 and two 3D system sizes of 4x4x4 and 6x6x6. From Fig. 34 it can be observed that $C = 2$ is beneficial even in 16x16 and 6x6x6 large sized systems. However, the relative gain obtained is smaller as compared to smaller systems of the same dimensionality. For example, it can be observed from Fig. 34 that in the larger 6x6x6 system, the maximum message throughput is 0.85 messages/μsec with $C = 1$. This is much lower than the corresponding 1.2 messages/μsec obtained on the 4x4x4 system. Similarly, 1.0 messages/μsec obtained with $C = 2$ on the 6x6x6 system is much lower than corresponding 1.7 messages/μsec obtained on the 4x4x4 system. Thus, the relative gain by using $C = 2$ on the 6x6x6 system is only 20% (lower than corresponding gain of 40% on the 4x4x4 system).

The above results indicate that under uniform traffic benefits from more consumption channels do reduce with larger system size. However, the benefits are significant to merit them even in reasonably large system sizes. Similar trends were observed on 2D systems, 8x8 and 16x16, as shown in top row of plots in Fig. 34. Although not reported here, with hot-spot traffic more consumption channels were derived to be beneficial even in large system sizes.
Impact of more virtual channels

Figure 35 shows the latency versus throughput plots on a 4x4x4 system. The number of virtual lanes is varied from \( v = 2, 4 \) and 6. The following observations can be made from Fig. 35. With \( C = 1 \), the maximum achievable message rate is constrained by 0.3 messages/\( \mu \)sec in all three systems. This indicates that the consumption bottleneck with a single channel limits the benefits from higher number of virtual lanes. The effective number of channels in the three systems are \( C = 3, 4, \) and 4, respectively. Thus, with \( v \) increasing from 2 to 4 one more consumption channel is required to attain higher message rate. However, with \( v = 6 \), no further gain is achieved with more channels. This is because benefits from more virtual lanes level off at this point. These results validate Obs. 10.
Figure 35: Impact of more virtual lanes and consumption channels on performance in a 4x4x4 system. Three different values of $v = 2, 4,$ and $6$ were considered.

### 4.6.7 Impact of messaging overheads

Figure 36 shows the latency versus throughput for a 4x4x4 system with uniform traffic. The messaging overhead was varied from $t_{sw} = 50, 100,$ and $200$ cycles. The corresponding constraint on maximum throughput per node is $2.0, 1.0,$ and $0.5$ messages/$\mu$sec (as stated earlier $1$ cycle $= 0.01$ $\mu$sec). From Fig. 36 it can be observed that the maximum achievable throughput with $t_{sw} = 50$ cycles is $1.7$ messages/$\mu$sec. In this example the network bisection constraint does not allow performance to be as high as $2.0$ messages/$\mu$sec. In this system $C = 2$ is effective. For $t_{sw} = 1.0$ and $2.0$, the maximum achieved throughput of $1.0$ and $0.5$ messages/$\mu$sec, respectively, match the predicted message rate from messaging overhead constraint. At these lower message arrival rates $C = 2$ offers no gain in throughput over $C = 1$. However, benefits
in terms of lower message latency is obtained. Such benefits lead us to conclude that $C = 2$ is effective even in systems with high message overheads. Although not reported here, these benefits were derived to be higher with longer messages.

![Figure 36: Impact of messaging overheads on maximum message rate and effective number of consumption channels. Uniform traffic performance is shown for a 4x4x4 configuration with three different messaging overheads: $t_{sw} = 50, 100, \text{ and } 200 \text{ cycles.}]

4.6.8 Impact of I-to-P bandwidth

The available bus bandwidth from the interface to processor-memory for message consumption ($C_m$) restricts the maximum sustained message rate. As demonstrated in Sec. 4.5.5 this is critical when expected message rates are high as in an all-to-one communication pattern. Here we discuss results derived for three different patterns: all-to-one pattern, uniform traffic, and non-uniform traffic. Results were derived for a
range of values of $C_m = 1, 2, \text{ and } 8$. The value of 8 corresponds to a very high I-to-P bandwidth. Such high value of I-to-P bandwidth reflects systems where it is not a limitation to maximum performance. Thus, experiments with $C_m = 8$ demonstrate the maximum value of effective number of consumption channels required for a system with I-to-P bandwidth not being a bottleneck.

**All-to-one traffic**

Figure 37 shows the all-to-one completion time in 16x16 and 6x6x6 systems. From Fig. 37 it can be observed that on a 16x16 system and $C_m = 1$, the lowest completion time is attained with $C = 1$. Higher number of consumption channels offer no better performance. This leads us to conclude that $C = 1$ is the effective number of channels for an all-to-one pattern with $C_m = 1$. Similar result was also observed on a 6x6x6 system with $C_m = 1$. Similarly, with $C_m = 2$, the effective number of consumption channels were observed to be $C = 2$ in both the systems. These results validate Obs. 11 that in systems with low values of $C_m = 1 \text{ or } 2$, having higher $C$ does not lead to better performance with all-to-one traffic.

For high value of $C_m = 8$, the performance is not constrained by I-to-P bandwidth. It can be observed from Fig. 37 that $C = 3$ and 4 are effective on the 16x16 and 6x6x6 systems, respectively. Similar trends were also obtained with smaller systems: 8x8 and 4x4x4, and different message length. These results indicate that having $C = 3$ or 4 is adequate for achieving best performance even under very hot-spot traffic.
Figure 37: Impact of increasing I-to-P bandwidth \( (C_m) \) and consumption channels \( (C) \) on the completion time of an all-to-one pattern in a 16x16 and 6x6x6 system. Three different values of \( C_m = 1, 2 \) and 8 are considered.

Uniform and Non-uniform traffic

In Obs. 11 it was derived that under uniform traffic more than one consumption channel may be necessary to deliver better performance even with limited I-to-P bandwidth \( (C_m = 1) \). This was validated in our experiments. Figure 38 depicts the latency versus throughput plots with uniform traffic on a 4x4x4 system and \( C_m = 1 \) and 8. As discussed in Obs. 11 the trends with uniform traffic are unlike the results with all-to-one pattern. From Fig. 38 it can be observed that \( C = 3 \) is effective even with a low value of \( C_m = 1 \). The plots with \( C_m = 8 \) are very similar to those with \( C_m = 1 \), indicating that with high I-to-P bandwidth there is no further gain in uniform performance. It can be observed that \( C = 3 \) is effective even with \( C_m = 8 \).

Trends with non-uniform traffic were in between those obtained with uniform traffic and all-to-one pattern. Figure 39 depicts the latency versus throughput plots with a hot-spot traffic on a 4x4x4 system and \( C_m = 1, 2, \) and 8. From Fig. 39 it can be observed that with hot-spot traffic the maximum message rate is limited by the value of \( C_m \). As \( C_m \) was increased the maximum message rate achievable (with
higher number of consumption channels) also increased. With $C_m = 1, 2,$ and 8 the effective number of consumption channels are 2, 3, and 4, respectively.

Figure 38: Impact of increasing I-to-P bandwidth ($C_m$) and consumption channels ($C$) on system performance with uniform traffic.

Figure 39: Impact of increasing I-to-P bandwidth ($C_m$) and consumption channels ($C$) on system performance with non-uniform traffic.
4.7 Summary

In this chapter we have presented various deadlock-free routing alternatives in a $k$-ary $n$-cube cluster-$c$ system. We have also identified performance degradation in wormhole $k$-ary $n$-cube inter-cluster networks due to limited number of router-to-processor consumption channels at each node. It is shown that such degradation not only exists for hot-spot traffic but also for uniform traffic in systems using deterministic/adaptive routing with multiple virtual lanes. Contrary to previous results in the literature, we have shown that the advantages of adaptive routing and virtual lanes are very much limited by using only one consumption channel per node. We have proposed a new network interface design with multiple consumption channels and virtual multiplexing to alleviate such consumption bottleneck. Our study indicates that multiple consumption channels are required at each node to realize the full benefits of routing adaptivity and virtual channel flow control. It is also shown that higher number of consumption channels are required to alleviate hot-spot effects in non-uniform patterns. By examining the interplay between various system, technological, and traffic parameters we have established design guidelines that two consumption channels are useful in most cases. Three to four consumption channels have shown to be useful for systems with higher adaptivity, low messaging overheads, high I-to-P bandwidth, and hot spot traffic. We have also shown that supporting 2-4 consumption channels is realizable under current router technology. Thus, current and future wormhole $k$-ary $n$-cube cluster-$c$ systems can achieve better performance by using multiple consumption channels.
CHAPTER V

Integrated framework for design under packaging and technological advancements

Rapid developments in the field of processor, interconnect, and packaging technologies make the task of efficient design of large multiprocessor systems a difficult one [27, 40, 69]. Design guidelines need to take into account technological changes to yield system configurations delivering best performance. Several previous studies have considered packaging constraints [6] while selecting the best system configuration. These studies include Dally’s [24] analysis of $k$-ary $n$-cube interconnection under a VLSI model with constant bisection bandwidth and Abraham and Padmanabhan’s study [2] under a constant pinout from a processing node. Agrawal’s [3] analysis of $k$-ary $n$-cube networks considers three different constraints: constant bisection width, constant channel width, and constant pinout while considering node and wire delays. However, Ranade [61, 62] and Yew [41] have argued that neither Dally’s VLSI model with limited bisection bandwidth nor the limited pinout model as proposed in [2, 3] is adequate while designing very large systems. Both models confine to only one level of packaging hierarchy whereas large systems typically employ several levels of packaging.
It was demonstrated in [62] that while designing a large multiprocessor system with multiple levels of packaging, multi-level/hierarchical/clustered architecture can be an advantage. The architectural levels can be chosen to closely match the packaging hierarchy leading to better designs. A variety of two-level hierarchical configurations have been proposed by researchers in the past to build scalable systems. Though a system design with multiple levels is a more generalized approach, it is commonly believed that two levels are sufficient to build parallel systems in the near future [62]. Moreover the design techniques which work for two-level hierarchies can be easily extended to accommodate more levels. Examples of previous work in this area include two-level systems based on hypercube and other network topologies[27, 54, 62], MINs and n-hop networks [62], and combination of bus and mesh/hypercube networks[40]. Though these designs provide alternative ways to build parallel systems, most of them do not take any packaging constraints into account. Thus, it is difficult to use these configurations to build realistic systems under varying technological and packaging constraints.

In this chapter, we propose an integrated supply-demand framework for multiprocessor system design. The overall framework is summarized in Fig. 40. The basic objective of the framework is to design the best system configuration for a given number \( N \) of processors using a two-level architecture. The family of flat \( k \)-ary \( n \)-cube topologies and their clustered variations \((k \)-ary \( n \)-cube cluster-c\) are considered to derive scalable configurations. Theoretically a large number of alternate configurations are possible to build such a system. We consider several packaging constraints
such as: varying board sizes, reasonable channel widths in multiples of a byte, limited pins from a router chip, and limited pinout from board depending on its size. The set of design-feasible configurations conforming to the packaging constraints are first derived. The demanded performance (latency and throughput) are then derived for a given processor and interconnect technology. The latency-throughput performance characteristics offered by each design-feasible configuration is estimated through analytical modeling and compared to the demanded performance. Among the good design-feasible configurations satisfying the demanded performance, the best configuration is decided on the basis of cost-effectiveness and scope for scalability. The elegance of the framework lies in its parameterized representation of different technologies. The best configuration can be derived for any set of technologies and constraints by choosing appropriate values for the parameters. We illustrate the framework by deriving best configurations to design a 1024 processor system. Using the framework we also study the impact of various packaging parameters and demanded performance on the overall design process. Among others the effect of varying maximum board size, higher router pinout, and alternate board pinout technology on the set of design-feasible/good/best configuration is analyzed and suitable design guidelines are derived to aid an architect in the design process. For a wide range of parameters, it is shown that best configurations are achieved with up to 8 processors per cluster and 3D-5D inter-cluster interconnection. The proposed guidelines and results are verified through accurate simulation modeling.
This chapter is organized as follows. The impact of processor speed and communication link speed on the demanded performance is discussed in Sec. 5.1. In Sec. 5.2 we present a representative multi-level packaging model for clustered systems. Section 5.3 discusses the trends in growth of processor board sizes, pinout, channel width, and router pinout technologies. In Sec. 5.4 we derive expressions for offered channel width and bisection bandwidth under different packaging, processor, and interconnect technologies. Section 5.5 presents a latency-throughput performance model for $k$-ary $n$-cube cluster-$c$ clustered systems. In Sec. 5.6 we present the integrated framework and discuss important considerations in choosing the best configuration. Section 5.7 illustrates the framework to design a 1024 processor system. Section 5.8 demonstrates the impact of varying packaging and demand parameters.

Table 3 provides a summary of the main symbols and notations used in this chapter. In the remaining discussion we represent a $k$-ary $n$-cube cluster-$c$ organization
using a compact notation of \((k^n, c)\). For example, an 8x8 inter-cluster network with clusters of four processors each is represented as \((8^2, 4)\) or \((8\times8, 4)\). Similarly, a configuration with mixed radix inter-cluster network with 4x4x3x3 topology and cluster size 2 is represented as \((4^2 \times 3^2, 2)\).

5.1 Demanded Network Performance

In this section we characterize the demands made on the average system throughput and average network latency of an interconnection network. These demands are characterized by parameterizing processor and interconnection speed and communication characteristics of applications. We first analyze the impact of processor speed and communication link technology on the demanded average throughput in the system. The importance of a bound on the average network latency of a message is discussed next.

5.1.1 Effect of Processor and Interconnect Speed on Demanded System Throughput

With advances in processor technology, the rate at which processors can execute instructions is going up. Let us denote the computational speed of a processor as \(D\) MFLOPS. An increase in processing power leads to a higher value of \(D\). As a general observation in applications, an amount of computation is associated with a quota of necessary communication, expressed as the computation to communication ratio \((r\text{ FLOPs/bit})\) [64]. Therefore while computing at the rate \(D\) MFLOPS a processor demands a sustained communication rate or average throughput of \((D/r)\) Mbits/sec.
Table 3: Summary of Important Symbols and Notations Used in this Chapter.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>Unit area to hold a processor chip, associated memory, and router logic</td>
</tr>
<tr>
<td>$b$</td>
<td>Capacity of a board in units of $a$ (number of processors in a board)</td>
</tr>
<tr>
<td>$b' = b/c$</td>
<td>Number of clusters in a board</td>
</tr>
<tr>
<td>$c$</td>
<td>Size of each cluster</td>
</tr>
<tr>
<td>$(k^n, c)$</td>
<td>Shorthand representation for a $k$-ary $n$-cube cluster-$c$ organization</td>
</tr>
<tr>
<td>$n$</td>
<td>Dimensionality of inter-cluster network</td>
</tr>
<tr>
<td>$p_p$</td>
<td>Pinout density under periphery pinout technology</td>
</tr>
<tr>
<td>$p_s$</td>
<td>Pinout density under surface pinout technology</td>
</tr>
<tr>
<td>$r$</td>
<td>Computation to communication ratio (indicates application characteristics)</td>
</tr>
<tr>
<td>$t$</td>
<td>Cycle time to transfer a bit across a wire (indicates communication link technology)</td>
</tr>
<tr>
<td>$u_b$</td>
<td>Percentage utilization of board area $b$</td>
</tr>
<tr>
<td>$u_p$</td>
<td>Percentage utilization of available board pinout $P_b$</td>
</tr>
<tr>
<td>$B$</td>
<td>Bisection size in the inter-cluster network</td>
</tr>
<tr>
<td>$D$</td>
<td>Computational speed of a processor (indicates processor technology)</td>
</tr>
<tr>
<td>$L$</td>
<td>Message length in bits</td>
</tr>
<tr>
<td>$N$</td>
<td>Total number of processors in a system</td>
</tr>
<tr>
<td>$N' = N/c$</td>
<td>Total number of clusters in system</td>
</tr>
<tr>
<td>$N_{boards}$</td>
<td>Total number of processor boards used in a system ($\geq N/b$)</td>
</tr>
<tr>
<td>$P_b$</td>
<td>Total pincount from a board (depends on pinout technology)</td>
</tr>
<tr>
<td>$R$</td>
<td>Maximum pinout supported from a router</td>
</tr>
<tr>
<td>$T_c$</td>
<td>Average message latency in presence of contention</td>
</tr>
<tr>
<td>$T_{max}$</td>
<td>Upper bound on $T_c$</td>
</tr>
<tr>
<td>$W$</td>
<td>Inter-cluster channel width (number of wires)</td>
</tr>
<tr>
<td>$W'$</td>
<td>Channel width supported by technology</td>
</tr>
<tr>
<td>$W_p$</td>
<td>Maximum value of $W$ constrained by board pinout $P_b$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Demanded average throughput per processor reflecting both processor and communication link technologies. $\lambda = Dt/r$</td>
</tr>
</tbody>
</table>
To allow such a demanded traffic rate the underlying interconnection network should be able to sustain a minimum of \((D/r)\) Mbits/sec per processor. A typical estimate of the range of computation to communication ratio, as suggested in [64], is \(r = 0.125\) to 1.25 FLOPs/bit. For example, in a system with \(D = 100\)MFLOPS and \(r = 0.50\) FLOPs/bit the expected throughput demand on the network is \((100/0.50)\)Mbits/sec or \(200 \times 10^6\)bits/sec per processor.

To capture the competitive growth of the processor and interconnect technologies, we represent the above processor demand on average throughput in terms of (bits/network cycle) instead of (bits/sec). Let \(t\) denote the network cycle time (seconds), the time to transfer a bit across a network wire. A reduction in \(t\) allows more data to be sent across any wire in a given time leading to higher channel bandwidth. The average throughput of \((D/r)\) bits/sec can be rewritten as \((D/r)t\) bits/cycle. Observe that an increase in the value of \(D\) captures the advancement in processor speed while a decrease in the value of \(t\) captures that in link speed. To capture both these technological advancements together, let us introduce a parameter \(\lambda = (D/r)t\) for demanded average throughput in bits/cycle. It is to be noted that if processor and link speeds increase in the same proportion then \(\lambda\) does not change. For deriving representative values of \(\lambda\) in current and future systems, we consider predictions made by Patterson [57]. As shown in Table 4, for a computation to communication ratio \(r = 0.5\) FLOPs/bit the values of \(\lambda\) broadly lie in the range of 0.5-2.5 bits/cycle. For a lower value of \(r\) the corresponding values in the range of \(\lambda\) would be higher.

The bisection size of a network is defined as the minimum number of wires that
Table 4: Sample values of $\lambda$ for various representative combinations of processor and Interconnection Link speeds. A computation to communication ratio of $r = 0.5$ FLOPs/bit[64] is assumed.

<table>
<thead>
<tr>
<th>Processor Speed $D$ (MFLOPS)</th>
<th>Demand on Network $D/r$ (bits/sec)</th>
<th>Interconnection Link Speed (MHz)</th>
<th>Channel Cycle Time $t$ (sec/cycle)</th>
<th>$\lambda = (D/r)t$ (bits/cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>$1 \times 10^8$</td>
<td>150</td>
<td>$6.6 \times 10^{-9}$</td>
<td>0.66</td>
</tr>
<tr>
<td>50</td>
<td>$1 \times 10^8$</td>
<td>100</td>
<td>$10.0 \times 10^{-9}$</td>
<td>1.0</td>
</tr>
<tr>
<td>100</td>
<td>$2 \times 10^8$</td>
<td>150</td>
<td>$6.6 \times 10^{-9}$</td>
<td>1.3</td>
</tr>
<tr>
<td>250</td>
<td>$5 \times 10^8$</td>
<td>200</td>
<td>$5.0 \times 10^{-9}$</td>
<td>2.5</td>
</tr>
<tr>
<td>250</td>
<td>$5 \times 10^8$</td>
<td>500</td>
<td>$2.0 \times 10^{-9}$</td>
<td>1.0</td>
</tr>
<tr>
<td>1000</td>
<td>$2 \times 10^9$</td>
<td>1000</td>
<td>$1.0 \times 10^{-9}$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

It needs to be cut in order to divide the network into two equal parts [24]. It limits the number of bits that can cross from one half to another half of the network. Let us analyze the impact of sustaining an average throughput of $\lambda$ bits/cycle per processor on the network bisection size in the inter-cluster network. The total traffic injected by all processors per cycle is $N\lambda$ bits. For uniform traffic, the probability of a generated message being intra-cluster is $c/N$. Similarly, the probability of a message being inter-cluster is $(1 - c/N)$. Thus, on the average $N\lambda(1 - c/N)$ bits get injected into the inter-cluster network every cycle. On the average half of this inter-cluster traffic is destined to processors on the other half of the system. These messages require to cross the inter-cluster bisection indicating a demand of $N\lambda(1 - c/N)/2$ bits across the bisection every cycle. Denoting the inter-cluster bisection size as $B$ wires, it is clear that $B = N\lambda(1 - c/N)/2$ wires are needed to sustain the demanded traffic. However,
due to contention in the network the utilization of the network channels are only a fraction of the maximum capacity [24]. To compensate for such loss in bandwidth due to contention, the actual value of $B$ required to support a throughput of $\lambda$ bits/cycle has to be greater than $N\lambda(1 - c/N)/2$.

Similar to the demand on the inter-cluster bisection size to sustain a given throughput, the traffic going in/coming out/traversing from one processor to another inside a cluster also imposes a demand on the intra-cluster bisection size. This demand is clearly proportional to the size of a cluster. Thus, to support larger sized clusters, the intra-cluster bisection size should scale linearly with size $c$. Since packaging constraints are less rigid in lower hierarchies, it is possible to provide thicker channels/buses to achieve higher bandwidth inside a cluster [62]. Hence in this chapter, we focus only on the bisection size of the inter-cluster network.

### 5.1.2 Demand on Average Network Latency

The achievable average message latency can have a direct impact on the processor efficiency in a multiprocessor system. For example, consider a parallel application in which each processor in the system executes a series of computation blocks of 400 cycles each. At the start of a block each processor sends out request messages to other processors, the reply to which is checked at the end of the block execution (can be considered as prefetching data). Assume a processor continues execution of its next block only after it receives the reply to the previous request. Let the amount of time spent by a remote processor in sending out the reply message be small. Thus, it is critical that the average one-way message latency in the network be less than 200
cycles for a round-trip latency of less than 400 cycles. Otherwise, a processor is forced to idle while waiting for a reply message leading to a fall in performance. Let $T_{\text{max}}$ denote such an upper bound on the average network latency expected/demanded in a system. Clearly the value of $T_{\text{max}}$ depends on many factors like nature of the application and state of compiler technology. For example, the extent to which prefetching of data can be employed varies from application to application. Similarly, the state of compiler technology dictates the degree to which prefetching can be exploited. For illustrative purposes in this chapter we assume typical values of $T_{\text{max}}$ to be 100-200 cycles [35].

In this section we have defined demanded network performance in terms of a) sustained average throughput ($\lambda$ bits/cycle) and b) bound on average message latency ($T_{\text{max}}$ cycles). We refer to this as the demand side to our design framework. In the following sections we develop the supply side to our framework. The next section presents a typical multi-level packaging model for designing large multiprocessor systems. The packaging technologies and constraints of these levels are discussed in Section 5.3 and their impact on the set of feasible (or supplied) configurations derived in Section 5.6.

5.2 Packaging model for clustered systems

Some earlier models, proposed to capture packaging constraints, like the VLSI model with limited bisection bandwidth as proposed by Dally [24] and the limited pincount model as proposed by Agrawal [3] deal with only one level of packaging. Large parallel
machines, on the other hand, typically employ several levels of packaging. There are inherent technological constraints that limit factors like VLSI die sizes, chip pinout, board areas, board pinout, and number of boards per card-cage.

Figure 41 shows a typical multi-level hierarchy encountered while packaging a large clustered multiprocessor system. At the lowest level are processor, memory, and router VLSI chip modules. A single processor chip, its associated share of memory, routing, and other interface logic are usually referred together as a processing node. Multiple processing nodes are organized into cluster modules and placed on processor boards. A cluster module is usually compact and does not span across multiple boards. Figure 41 shows each board in the system having four cluster modules placed on it. Multiple boards are organized into a card-cage and a system may comprise of multiple such card-cages. For example, the system in Fig. 41 requires two card-cages each containing four boards. A larger system would require more boards and card-cages.

The modules at each level of this packaging hierarchy: chips, clusters, boards, card-cages have their own characteristics in terms of maximum capacity, bisection size, available pinout, and channel width. For example, the pinout from a cluster module may be limited to a maximum of 250-300 pins. Similarly, the maximum allowable area of a processor board, depending on the size of a cluster module, may be able to accommodate only up to 4-8 cluster modules. Such packaging characteristics are analyzed in more detail in the next section.

The above packaging hierarchy has a direct impact on the connections of the
inter-cluster channels. For example, let us consider two clusters connected by an inter-cluster channel. This channel utilizes the pinout from the associated cluster modules. The two clusters may lie on the same or on different boards. A channel between two clusters on the same board is connected through wires on the board. Such wires directly connect pins of one cluster module to another. However, connecting a channel between clusters on two different boards is more difficult. Pins from the cluster modules need to be first connected to pins of the respective boards. Figure 41 shows such board pins along the periphery of each board. The channel is completed by connecting the pins between the two board modules. This inter-board part of the channel is implemented either through a) shared backplane, for boards within the same card-cage or b) connector cable, for boards across different card-cages. Figure 41 shows examples of these two types of inter-board channels.

Large system size and multiple levels of packaging have significant impact on the
length of connecting wires in the system. The length of wires to connect channels depends on the distance between the connecting boards. Longer wires can lead to longer propagation delays and hence longer channel cycle times. However, this problem is alleviated by applying pipelining techniques over long wires [69]. In this study we assume such techniques being used to limit channel cycle time.

5.3 Parameterizing packaging technologies

The characteristics and limitations of each level of packaging has an extreme impact on the set of achievable or feasible configurations and their cost and performance. In this section we discuss important trends in board capacity, board pinout, channel width, and router pinout technologies. These technologies are parameterized and their impact on design constraints are analyzed.

5.3.1 Processor Board Technology

Processor boards cannot be arbitrarily large in size. The physical size of a board is restricted by electrical, mechanical, and board fabrication constraints. In terms of physical dimensions board sizes, being used in recent multiprocessor systems, vary from 6" × 4" to an aggressive 26" × 21" used in the J-machine [53]. The largest board size available to a system designer usually varies with technology and over a period of time. In this study we therefore do not present guidelines restricted to a particular largest board size. The available maximum board size is treated as a parameter in the framework. Depending on the size of a given board we can fit only a limited number of cluster modules or processing nodes on it. For example, consider a design
problem with board sizes of $12'' \times 6''$ and two processing nodes per cluster module. Let
the size of a processor chip be $2'' \times 2''$, the area occupied by associated local memory
of 16MB (say) be $6'' \times 4''$, and required intra-board estate for router, memory and
address buses and other support logic be 8 square inches. In such a design we require
around 36 square inches per processing node leading to 72 square inches per cluster
module. This implies that we can fit exactly one cluster module or two processing
nodes on each board.

It is natural to express board area or capacity in terms of absolute units such as
square inches. However, for ease of discussion, it is more convenient to express it in
relation to the area of a processing node. For example, consider a given board with
capacity specified as 8 processing nodes. This implies that such a board is also capable
of holding 4 clusters of 2 processing nodes each, or similarly, 2 clusters of 4 processing
nodes each. As discussed earlier we define the size of a processing node to include
the area required by a processor chip, its local memory, router, and other associated
interface logic. Such a combined area is denoted as parameter $a$. The magnitude of
$a$ depends mainly on two factors: a) level of integration in the VLSI chips and b)
the size of memory associated with each processor. A higher VLSI technology leads
to smaller $a$. Similarly, supporting larger memory per processor leads to larger $a$. A
designer can choose an appropriate value for $a$ depending on chip sizes and amount
of memory per processor and use this as an unit of board capacity. A board with
physical area $(ba)$ is defined to have a capacity of $(b)$. Thus, in the example in the last
paragraph $a = 36$ and a $12'' \times 6''$ board has a capacity of $b = 2$. Future advancements
in VLSI technology can lead to higher processor and memory integration. Let this be reflected in our design problem by choosing a suitably smaller value of \( a = 24 \). This implies that the same 12" \( \times 6" \) board in terms of the new value of \( a \) has a capacity of \( b = 3 \) processing nodes. In further discussion, unless otherwise mentioned, we use the terms board size, board area, and board capacity in an interchangeable manner.

Assume a cluster module containing \( c \) processing nodes require an area \( ca \). Thus, a board with capacity \( b \) can hold up to \( b' = b/c \) clusters on it. To build a system with \( N \) processing nodes a total board area of \( Na \) is necessary. A total board area larger than \( Na \) may be used but this clearly leads to wastage of precious board estate. Thus, we suggest using a total board area close to \( Na \). We denote the maximum board size discussed earlier as parameter \( b_{\text{max}} \). For illustrative purposes, we later consider a \( b_{\text{max}} \) of 8 and smaller in this chapter. However, the framework is valid for any value of \( b_{\text{max}} \) being offered by technology. All boards used in a given system are assumed to be of the same size and board capacity satisfies the constraint, \( 1 \leq b \leq b_{\text{max}} \). Total number of boards used in designing an \( N \)-processor system is defined as \( N_{\text{boards}} = N/b \).

### 5.3.2 Board Pinout Technologies

The pin-count \( P_b \) out of a board has a direct influence on the data volume that can flow in/out of a given processor board. Currently two different types of technologies are being employed by the computer industry:

a) **Peripheral pinout:** This is the traditional technology [70, 74] where the periphery of a board is used for external connections. The exact relationship between the
size of board periphery and pin-count depends on the pin connector technology. However, it is reasonable to expect this relationship to be linear. Thus, a larger board, having a longer periphery, can support more pins. Typically only one or two sides of a board are used for periphery pinout. Without loss of generality let us assume only one side being used for pinout as shown earlier in Fig. 41. Let $p_p$ denote the peripheral pinout density, the pin-count that can be supported from an unit board length. Thus, a larger square board with capacity $b$ having each side of length $\sqrt{b}$ units, can support a total of $P_b = p_p \sqrt{b}$ pins.

b) Surface pinout: This is representative of a more aggressive pinout technology. The surface of the board is utilized for external connections. Representative examples are electronic interconnections using elastomeric connectors [53] and optical interconnections [65]. Let $p_s$ denote the surface pinout density, the pin-count that can be supported from a board of unit capacity. Assuming a linear relation between board area and surface pinout, the pin-count supportable from a board of capacity $b$ is $P_b = b \times p_s$.

Figure 42 shows the growths of pincount with board area under the respective surface and peripheral technologies for different relationships between $p_s$ and $p_p$. The vertical clipping line in each graph reflects that the board size cannot grow continuously and is limited by some maximum size. As expected, for large board sizes the surface technology always supersedes the peripheral technology. However if $p_s < p_p$, then for smaller board sizes, the peripheral technology can beat the surface technology. Representative values of $p_p$ and $p_s$ derived from the current design trends [23, 53]
Figure 42: Growth of pincount with board area under peripheral and surface technologies shown for two different relationships between \( p_p \) and \( p_s \). The parameters \( p_p \) and \( p_s \) reflect the pincount that can be supported from a board of unit capacity under periphery and surface pinout technologies, respectively.

are: \( p_p = 128 - 256 \) and \( p_s = 64 - 128 \). Future values of these parameters will depend on the advances made in the respective technologies. In the next section we compare the impact on designing clustered systems under both these pinout technologies.

5.3.3 Channel Width Technology

Most current parallel machines have 8 and 16-bit data channels. This corresponds to a channel width of \( W \approx 12 \) and 24, respectively, including control, acknowledgment, and parity wires. For a simpler interface design, the width of data lines in a channel is expected to maintain an integral relationship with that of processor and memory which are typically in multiples of a byte. Further, factors like path-width inside routers, and connector technology restrict channel widths from being arbitrarily large. Most previous studies while proposing design guidelines, did not consider such constraints on channel width. In our framework we account for these and denote a channel width supportable by technology as \( W' \). For example, current technology supports \( W' = 12 \) and 24. In the near future it is expected that a wider channel width technology with ability to carry 32-bit data \([13, 57]\), corresponding to \( W' \approx 40 \), would
be feasible. In the next section we investigate the impact of such supportable fixed channel width \((W')\) on the set of design-feasible configurations.

### 5.3.4 Router Pinout Technology

Similar to the restriction on maximum pinout from a processor board, the pinout from a router is also limited. The available number of pins from a router restricts the number of channels and channel widths that can be supported[3]. It is to be noted that as discussed in Sec. 5.2 the router is a part of the cluster module. Thus, in this organization router pinout is synonymous with cluster module pinout. Let \(R\) denote the maximum pinout from a router (cluster module) being available for inter-cluster channels. Let us consider the required pinout for inter-cluster channels from a router in a \((k^n, c)\) system. In a multi-dimensional system a router has to support at least two channels along any dimension, one for an incoming and another for an outgoing channel. For a system with bidirectional channels this number is four per dimension. The number of channels to be supported from a router assuming unidirectional channels is \(2n\). Supporting \(W\) bit wide channels in such a system requires a pinout of \(2nW\) from each router. Clearly, this imposes a constraint of

\[
2nW \leq R.
\]  

\((5.1)\)

As an illustration consider a representative value of \(R \approx 250\). Using the above equation let us derive the maximum dimension \((n)\) that can be supported in an interconnection. For various current and future channel width technologies: \(W' = 12, 24,\) and 40, the maximum dimension supported are \(n = 10, 5,\) and 3, respectively.
Thus, it can be observed that even with a conservative pinout technology and a channel width technology of $W = 40$, up to 3D systems are feasible. For thinner channels the maximum supportable dimension can be even higher.

5.4 Impact of varying architectural parameters

In this section we analyze the impact of packaging constraints on the channel width and bisection size of $k$-ary $n$-cube cluster-$c$ configurations. The maximum offered channel width is derived based on board size and pinout constraints. We then analyze the impact of maintaining a fixed supportable channel width on system dimensionality and cluster size. It is then shown that a realistic design can lead to under-utilization of board resources. Minimizing such under-utilization is necessary to minimize system cost. Based on this, a process of deriving system configurations minimizing under-utilization is proposed. We then analyze the impact of increasing system and cluster size on the bisection size of a system while maintaining a fixed supportable channel width. Most results presented in this section hold for both periphery and surface pinout technologies. In case of differences these are specifically mentioned. All results are derived for arbitrary cluster sizes ($c \geq 1$) and the corresponding results for flat architectures can be obtained by choosing $c = 1$.

5.4.1 Offered Inter-Cluster Channel Width

Let the clusters in a system be placed on multiple boards and interconnected by channels to yield a $N' = N/c = k_1 \times k_2 \times \cdots \times k_n$ inter-cluster topology as discussed earlier in Section 5.2. Each board in this system holds $b'$ clusters. The clusters on
any board together with the inter-cluster channels between them can be visualized as a sub-topology of the overall inter-cluster topology. Let this sub-topology be represented as \((b' = b_1 \times b_2 \times \cdots \times b_n)\), where \(\forall_{i=1}^{n}(b_i < k_i)\) and some \(b_i\)'s may be 1 to depict that only one cluster exists on the board along that dimension. The condition \(\forall_{i=1}^{n}(b_i < k_i)\) is assumed instead of \(\forall_{i=1}^{n}(b_i \leq k_i)\) to reflect a typical constraint that any inter-cluster dimension is large enough such that it can not fit on a single board. This is reasonable to expect for large system sizes. Figure 43 shows an example system with 48 clusters being placed on 12 boards each with a capacity for holding four processor clusters. The overall inter-cluster topology is assumed to be \((N' = 4 \times 4 \times 3)\). The sub-topology on each board is assumed to be \((b' = 2 \times 2 \times 1)\). Let the

![Diagram](image)

**Figure 43:** A system with 48 clusters being placed on 12 boards each with a capacity for holding 4 processor clusters. The overall inter-cluster topology is 4x4x3. The sub-topology placed on each board is 2x2x1.

number of rows of processor clusters in dimension \(i\) of the sub-topology on a board be denoted as \(r_i\). This can be derived as \(r_i = b'/b_i\). For example, the number of such rows on each board in Fig. 43 along dimensions marked 1, 2, and 3 are 2, 2, and 4, respectively. Note that inter-board channels go out/come in from the two ends of each
row leading to $2r_i$ inter-board channels\(^5\) along dimension $i$. Hence, the total number of inter-board channels required to go out/come into a board from all $n$ dimensions is derived as $2\sum_{i=1}^{n}r_i = 2b\sum_{i=1}^{n}(1/b_i)$. As discussed earlier in Section 5.2 inter-board channels between boards in the same card-cage can be connected through a shared backplane. The connections from a board to the backplane are made using the pinout from the board. In large systems with many boards and multiple card-cages a single backplane cannot be used to connect all boards. Thus, cable connectors also need to be employed to connect the inter-board channels across card-cages. Here too, the connections from the board to cable connectors are established using the pinout from the board. Thus, the inter-board channel width is constrained by the available board pinout. To support a channel width of $W$, a total pincount of $2b/W\sum_{i=1}^{n}(1/b_i)$ is required from each board. Thus, the maximum supportable inter-board channel width from pinout restrictions, is derived as:

$$W = \frac{P_b}{2b\sum_{i=1}^{n}(1/b_i)}$$  \hspace{1cm} (5.2)

Assuming that the sub-topology on the board is regular\(^6\), i.e. $(\forall i : b_i = b_i^{1/n})$ and using $b' = b/c$, this can be simplified to,

$$W = \frac{P_b}{2nb'(n-1)/n} = \frac{P_b}{2n(b/c)^{(n-1)/n}}$$  \hspace{1cm} (5.3)

\(^5\)This is true for unidirectional channels. For full duplex bidirectional channels this number would be $4r_i$.

\(^6\)Assumption of regular topology wherever made in this section is only for formula simplification and providing a more intuitive understanding of the interplay between various parameters. However, the proposed framework is also valid for the mixed radix topologies. Design solutions together with simulation results are presented later with mixed radix topologies.
Similar to the inter-board inter-cluster channels connecting clusters across different boards, intra-board inter-cluster channels are required to connect the clusters on a single board to maintain the \((b_1 \times b_2 \times \ldots \times b_n)\) sub-topology on a board. These channels can be implemented using on-board wires. For an on-board implementation it is intuitive to expect the available board bisection to limit the size of such intra-board channels. However, with multi-layered boards, it is reasonable to assume on-board connection density to be much higher than off-board density. Hence the inter-cluster channel width is mainly limited by the inter-board channel width determined by the pinout constraint as shown in Eqn. 5.3.

From the expression for \(W\) in Eqn. 5.3, it can be observed that the channel width is determined by board size, cluster size, pinout technology from board, and dimensionality of the inter-cluster network. It does not depend on the total system size. For a given pinout technology and board size let us consider the impact of varying the inter-cluster dimensionality while keeping cluster size fixed. In order to support larger dimensional networks, we need more channels. This leads to thinner channels with a fixed pinout from a board. This observation matches with a similar result shown in [3, 24] while designing flat systems. Similarly, for a given pinout technology and board size, decreasing the cluster size while keeping inter-cluster dimensionality fixed also leads to thinner channels. A smaller cluster size results in larger number of clusters on board. This implies that pinout from the board gets shared among more clusters leading to thinner channels. These observations can be summarized as:

**Result 5** For a fixed pinout from a board of given size, the offered channel width
(W) falls with increase in the inter-cluster network dimensionality (n) while keeping cluster size (c) fixed. Similarly, channel width (W) falls as the inter-cluster dimensionality (n) remains fixed and the cluster size (c) reduces.

Let us consider the impact of board size on channel width W. Under periphery pinout technology, with \( P_b = p_p\sqrt{b} \) we can simplify Eqn. 5.3 as:

\[
W = \frac{p_p c^{(n-1)/n}}{4 b^{(n-2)/(2n)n}}.
\]

(5.4)

Thus, for a given cluster size and inter-cluster dimensionality \( n > 2 \), an increase in board size \( b \) leads to a fall in channel width. For a 2D inter-cluster network \( (n = 2) \) the channel width remains constant while that in an 1D network rises sharply. Under surface pinout technology, with \( P_b = p_s b \), a similar simplification leads to:

\[
W = \frac{p_s c^{(n-1)/n} b^{1/n}}{4 n}.
\]

(5.5)

Here, for any given cluster size and inter-cluster dimensionality, an increase in board size \( b \) leads to a rise in channel width. However, this rise is not very significant for higher dimensions. This leads to:

**Result 6** Under periphery pinout technology, keeping cluster size c fixed and dimensionality of inter-cluster network fixed at \( n > 2 \), an increase in board size results in channel width to fall. However, under surface pinout technology, for any fixed inter-cluster dimensionality it leads to a rise in channel width.

### 5.4.2 Supporting a Fixed Inter-Cluster Channel Width

Most of the prior studies on system design, while proposing guidelines under different constraints like constant bisection bandwidth [24], did not impose any restrictions
on the values that the channel width can take while satisfying other constraints. As discussed in Section 5.3.3 supporting an arbitrarily large channel width is difficult. Further the data lines are expected to be integral multiples of bytes for easier interfacing with processors and memories. Representative values of supportable channel width ($W'$) as discussed earlier in Section 5.3.3 are 12, 24, and 40.

In Equations 5.4 and 5.5, we showed that offered channel width $W$ is a function of system parameters $n$, $b$, $c$, and $p_p$ or $p_s$. Given a channel width technology $W'$ to be supported, an obvious design objective while selecting values for $n$, $b$, and $c$ is to ensure that the offered channel width ($W$) is equal to the supportable channel width ($W'$). Based on Equations 5.4 and 5.5 we analyzed the relationships among these parameters and the impact of varying one parameter on another was studied while maintaining a fixed $W$ (= $W'$). Table 5 summarizes some of the important interplays. An entry in the table: fixed, ↑, or ↓ corresponds to the respective parameter being kept fixed, increased, or decreased, respectively. For example, the first row of the table indicates that for a given fixed board size and pinout density, increasing (↑) cluster size requires the inter-cluster dimensionality to be also increased (↑). Similarly, second and third rows denote the impact of increasing board pinout density on inter-cluster dimensionality and cluster size, respectively. With increase (↑) in board pinout density the supported pinout from a given board is higher. The increased pinout can be used to support more inter-cluster channels from a board while maintaining a given channel width. The extra channels can support a) higher (↑) dimensional inter-cluster networks requiring more channels or b) smaller (↓) clusters. With smaller clusters,
building a system of given size requires more clusters. Interconnecting a larger number of clusters, while maintaining the inter-cluster dimensionality and channel width, requires more channels in the system. Increased pinout from boards make it possible to support the needed extra channels. It may be noted that relationships obtained by reversing the sense of all arrows in Table 5 also hold true.

Table 5: Summary of Interplay Between Board Size ($b$), Pinout Density ($p_p$ or $p_s$), Cluster Size ($c$), and Inter-Cluster Dimensionality ($n$) to Maintain a Fixed Channel Width ($W'$).

<table>
<thead>
<tr>
<th>Board Size</th>
<th>Pinout Density</th>
<th>Cluster Size</th>
<th>Inter-Cluster Dimensionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>fixed</td>
<td>↑ fixed</td>
<td>↑</td>
</tr>
<tr>
<td>fixed</td>
<td>↑</td>
<td>fixed</td>
<td>↑ fixed</td>
</tr>
<tr>
<td>↑</td>
<td>fixed (periphery)</td>
<td>↑ fixed</td>
<td>fixed</td>
</tr>
<tr>
<td>↑</td>
<td>fixed (surface)</td>
<td>↓ fixed</td>
<td>fixed</td>
</tr>
</tbody>
</table>

5.4.3 Designing with Under Utilization of Resources

The expression in Eqn. 5.3 yields a channel width $W$ assuming full utilization of board area and pinout resources. However, while building a real machine it is unlikely that both board area and pinout resources get fully utilized. For example, a fraction of the board pinout capacity may remain unutilized if a larger board size is required to fit a desired topology but the resulting larger pinout is not required. To make our design framework more realistic, we allow such under-utilization of board and pinout
capacities. However, in order to minimize system cost we also add a design objective to minimize such under-utilization of resources. Let us denote parameters $u_p$ and $u_b$ as the percentage utilization of board pinout and board area, respectively. Now consider supporting a channel width of $W = W'$ with marginal under-utilization of board area and pinout resources being allowed. Based on Eqn. 5.3 we can derive,

$$W' = \frac{P_b u_p}{4 n u_b (b/c)^{(n-1)/n}} = W \frac{u_p}{u_b}$$

(5.6)

where $P_b u_p$ is the utilized pincount from board, $u_b b'$ is the actual number of clusters placed on a board (out of the maximum $b' = b/c$), and $W$ is the channel width value obtained assuming full-utilization of resources. We assume a reasonable bound on both under-utilizations such that $u_p, u_b \geq u_{\text{min}}$ for some $u_{\text{min}}$ closer to 1. Thus, observations made earlier assuming full-utilization ($u_{\text{min}} = 1$) continue to hold. For illustrative purposes, in this chapter, we choose $u_{\text{min}} = 0.9$. Given a pinout density and a channel width technology, we formulate the following search problem to determine $(n, b, c)$ tuples which satisfy the following inequalities.

Allowing only under-utilization of pinout, we have $0.9 \leq u_p \leq 1$ and $u_b = 1$. This leads to:

$$W = W' (u_b / u_p) \Rightarrow W \leq 1.1 W'$$

(5.7)

Allowing only under-utilization of board area, we have $0.9 \leq u_b \leq 1$ and $u_p = 1$. This leads to:

$$W = W' (u_b / u_p) \Rightarrow W \geq 0.9 W'$$

(5.8)
The derived solution configurations, in terms of \((n, b, c)\), comply with supported channel width \((W')\) technology while utilizing board resources maximally. In Section 5.6 we demonstrate that a subset of these configurations, while complying with other packaging constraints like maximum router pinout and maximum board size, form the set of design-feasible or package-able configurations.

### 5.4.4 Offered Inter-Cluster Bisection

Now let us consider the impact of packaging on offered inter-cluster bisection size. As discussed earlier in Section 5.1.1 the bisection size of the inter-cluster network \(B\), or more specifically bisection size per processor \(B/N\), is indicative of the supportable average throughput per processor in the system [24]. Observations derived in this subsection for the bisection size per processor reflect similar trends on system throughput. These observations are expected to be broad guidelines to aid the design decision process. A more accurate modeling of the average throughput in presence of contention is presented in the next section.

Given a \((l_1 \times l_2 \times \ldots \times l_k)\) mesh/torus inter-cluster network with a given channel width, \(W\), the size of the inter-cluster bisection can be computed in the following manner. A \((k_1 \times k_2 \times \ldots \times k_n)\) mesh/torus can have various bisections which divide this network into two halves. For example, in a 3D torus with \(x\), \(y\), and \(z\) dimensions, we can have three possible bisections: one orthogonal to \(x\) dimension along \(yz\) plane, one orthogonal to \(y\) dimension along \(xz\) plane, and so on. We are interested in the size of the smallest bisection in the system because it maximally constrains the performance of the system under random traffic. Clearly, the smallest bisection has to be
orthogonal to the dimension having the largest radix, given by \( k_{\text{max}} = \max_{i=1..n}(k_i) \).

The number of nodes on either side of such a bisection is given by \( N'/k_{\text{max}} \), where \( N' \) as defined in Sec. 5.4.1 denotes the total number of nodes (clusters) in the system. Let each node on one side of the bisection be connected to exactly another across the bisection using an unidirectional channel of width \( W \). This leads to the offered bisection size, \( B \), being \( \frac{N'W}{k_{\text{max}}} \) wires. Thus, the expression for \( B/N \), the bisection size per processor, is derived as \( \frac{N'W}{Nk_{\text{max}}} \) wires. The presence of wrap-around channels in torus doubles this number\(^7\) to \( \frac{2N'W}{Nk_{\text{max}}} \). The expression can be simplified by assuming the inter-cluster network to be regular i.e. \( k_{\text{max}} = (N/c)^{1/n} \) and \( N'/N = 1/c \) to:

\[
\frac{B}{N} = \frac{2N'W}{Nk_{\text{max}}} = \frac{2W}{c(N/c)^{1/n}} = \frac{2W}{N^{1/n}c(n-1)/n}
\] (5.9)

Based on the above equation the following observations can be made:

**Result 7** For a given channel width \( W = W' \) and inter-cluster dimensionality \( n > 1 \), the offered bisection size per processor \( (B/N) \) falls with increase in the system size \( N \) while keeping cluster size \( c \) fixed. Similarly, the bisection size per processor \( (B/N) \) also falls as the system size \( N \) remains fixed and cluster size \( c \) is increased.

Figure 44 shows the bisection size per processor for a system size of \( N = 1024 \) processors, a given channel width of \( W' = 24 \), and different inter-cluster dimensionality \( (n = 1 - 4) \) as cluster size is varied \( (c = 1 - 16) \). It can be observed that the fall in bisection size per processor is more appreciable at smaller cluster sizes and higher inter-cluster dimensionalities.

\(^7\)For bidirectional duplex channels this value needs to be multiplied by another factor of 2.
Figure 44: Offered inter-cluster bisection per processor \((B/N)\) as a function of cluster size and inter-cluster dimensionality while maintaining a channel width \(W' = 24\) for a 1024 processor system. Note that for a given pinout technology, the board sizes are not fixed and are changed with cluster size to maintain \(W' = 24\).

It is also interesting to observe the impact of board size on offered bisection size per processor under two different pinout technologies. Figure 45 shows the plots of \(B/N\) for a \(N = 1024\) system. The trends are shown with respect to periphery and surface pinout technologies as board size is increased. To analyze these trends we

Figure 45: Offered inter-cluster bisection per processor \((B/N)\) in a system of 1024 processors as a function of board size and different inter-cluster dimensionalities under two different pinout technologies: a) periphery pinout with \(p_p = 128\) and b) surface pinout with \(p_s = 128\).
first simplify Eqn. 5.9 by using Eqn. 5.3 to replace \( W \), leading to:

\[
\frac{B}{N} = \frac{2}{N^{1/n} c_{n-1}^{1/n}} \frac{P_b}{2n b^{(n-1)/n}}
\]

(5.10)

For a periphery pinout technology with \( P_b = p_p \sqrt{b} \), this can be further simplified to

\[
\frac{B}{N} = \frac{P_b}{N^{1/n} b^{(n-2)/(2n)}}. 
\]

Thus, for a given system size \( (N) \) and inter-cluster network dimensionality \( (n) \), the bisection size per processor \( (B/N) \) falls with increase in board size \( (b) \) for \( n > 2 \). This can be explained by the fact that under periphery pinout the total board pin-count in the system reduces as board sizes are increased. For \( n = 2 \), the value of \( B/N \) remains fixed and for \( n = 1 \) it rises slowly. The key point is that under periphery pinout technology, it is ideal to work with smaller boards as they offer higher bisection sizes and hence performance. Similarly, under surface pinout technology with \( P_b = p_s b \), we can simplify Eqn. 5.10 as \( B = \frac{P_s b^{1/n}}{N^{1/n}} \). Here, it can be observed that for a given system size \( (N) \) and inter-cluster network dimensionality \( (n) \), the size of the bisection increases with \( n \). These lead to:

**Result 8** Under periphery pinout technology, for a given system size \( (N) \), keeping the dimensionality of inter-cluster network \( (n) \) fixed at a value greater than 2, an increase in board size \( (b) \) leads to a fall in the inter-cluster bisection size per processor. However, under surface pinout technology, as board size \( (b) \) is increased, bisection size per processor increases for all inter-cluster dimensionalities.

Bisection size per processor has direct impact on the maximum traffic that can be supported in a system. Thus, trends similar to those presented in this section with respect to bisection size per processor are expected in the maximum value of
the offered average system throughput. In the following section we present a simple analytical model for \((k^n, c)\) systems to estimate their performance in presence of contention. We use such a model in Sec. 5.6 to determine a more accurate trend about the average system throughput and offered average message latency while deriving configurations supporting a demanded performance.

### 5.5 Latency-throughput performance model for \(k\)-ary \(n\)-cube cluster-\(c\) systems

#### 5.5.1 The Model

We develop a simple analytical model to predict performance in \((k^n, c)\) systems with small cluster sizes. This model considers network contention and determines latency and throughput parameters for a given traffic load. Our model is an extension of the model proposed by Agrawal [3] to predict performance in flat \(k\)-ary \(n\)-cube networks with dimension-order [24] virtual cut-through routing. As shown in [3] the average message latency in the presence of contention through a \(k\)-ary \(n\)-cube network can be derived as:

\[
T_c = \left[ 1 + \frac{mF^2}{(1 - mFd)} \frac{(d - 1)}{d} \left( 1 + \frac{1}{n} \right) \right] nd + F, \tag{5.11}
\]

where \(T_c\) denotes the average message latency expressed in network cycles. Network cycle, as defined earlier in Sec. 5.1.1, is the time to send a flit across one hop in the system. The parameter \(d\) represents the average number of hops taken by a message in a dimension. For a network with unidirectional channels and wrap-around connections, \(d = (k - 1)/2\). For bidirectional channels this value is \((k - 1)/4\). The message size expressed in flits is \(F = L/W\), where \(L\) is the message length in bits and
$W$ the channel width. The parameter $m$ is the message injection rate by a processor in terms of messages/cycle. From the above equation, it can be observed that $nd + F$ cycles is the minimum latency suffered by a message. This happens at a very small value of message injection rate.

In a clustered $(k^n, c)$ system depending on whether a message is intra-cluster or inter-cluster it encounters different latencies. For relatively small clusters ($c \leq 8$) it is reasonable to assume advanced packaging, faster interconnect, and wider buses inside a cluster leading to very high intra-cluster message bandwidth. With such high bandwidth it is reasonable to expect small delays inside clusters. Inter-cluster messages on the other hand need to travel across clusters. The average delay for inter-cluster traversal is relatively much larger than average intra-cluster delay.

Our objective is to derive the average message latency as a function of average message throughput. The average message latency in a clustered system is determined as a weighted sum of the intra-cluster and inter-cluster message latencies. The weights are the relative frequency of these messages being generated by a processor. Under uniform traffic, the probability of an intra-cluster message occuring is given by the expression $(c/N)$. Clearly, for reasonably large systems and relatively small cluster sizes the value of $c/N$ is negligible. For example, in a system with $N = 1024$ processors and a cluster size of $c = 8$ the value of $c/N = 0.781\%$. The probability of a message being inter-cluster, $(1 - c/N)$, is therefore very close to 1. This indicates that most messages generated in the system are inter-cluster. Under such an assumption, the
average message latency in the system is closer to the average latency of an inter-cluster message. In this analysis we therefore focus on deriving the average latency of an inter-cluster message. An uniform traffic model was chosen because it is considered more representative while designing a general purpose machine. No prior knowledge is assumed about the nature of the applications to be run on it.

Let us consider the latency of an inter-cluster message. It has three components $T_{\text{intra}_1}, T_{\text{inter}},$ and $T_{\text{intra}_2}$. These terms denote the delays from the source processor to source-CI through the source intranet, source-CI to destination-CI through the $k$-ary $n$-cube internet, and destination-CI to the destination processor through the destination intranet, respectively. The exact expressions for $T_{\text{intra}_1}$ and $T_{\text{intra}_2}$ depend on the cluster topology, cluster size, and rate of traffic. However, from earlier discussion we know that these latency components inside clusters, $T_{\text{intra}_1}$ and $T_{\text{intra}_2}$, are much smaller as compared to the inter-cluster component $T_{\text{inter}}$. For star-based clusters which is a popular trend in current multi-processor systems as discussed earlier these components are negligible. Assuming a clustered system offering small intra-cluster latencies, the average latency seen by a message, $T_c$, is dominated by the average inter-cluster latency. To estimate this factor we modify the model in [3] in the following manner.

Given the message injection rate from each processor to be $m$ messages/cycle, the combined traffic generated by all the processors in a cluster is $mc$ messages/cycle. Under uniform traffic assumption, most of this traffic is inter-cluster. We assume
this resultant traffic to be injected by the CI into the internet. Assuming the resultan
tant traffic stream to be Poisson, the inter-cluster latency can be predicted from
Equation 5.11 by replacing the parameter \( m \) by the expression \( mc \) leading to:

\[
T_c = \left[ 1 + \frac{mcF^2}{(1 - mcFd)} \frac{(d - 1)}{d} \left( 1 + \frac{1}{n} \right) \right] nd + F. \tag{5.12}
\]

Equation 5.12 can be rewritten to express injection rate \( m \) as a function of \( T_c \)
and other parameters as:

\[
m = \frac{(T_c - F - nd)d}{cFd((n + 1)(d - 1)F + (T_c - F - nd)d)} \tag{5.13}
\]

### 5.5.2 Comparing Performance of the Model with Simulations

We validated the above model through simulations against a wide range of clustered
systems with varying inter-cluster topology and cluster size with variety of workload
parameters. It was found that the model closely predicted the simulated performance.
For example, a comparison of simulation results with analytical prediction from the
above model are depicted in Fig. 46. In Fig. 46(a) as message length \( F \) is varied
from 2-12 flits in a 4-ary 3-cube cluster-4 \((4^3, 4)\) system, the model closely matches
the simulation results. Similar matches were obtained with varying cluster sizes from
1 through 8, as depicted in Fig. 46(b). Our clustered simulation testbed models
star-clusters, routes messages through the inter-cluster network in dimension-order,
and generates statistics such as average message latency and observed message rates.
Each simulation was run until the 95% confidence interval of a data point was within
5% of its mean. In the next section we demonstrate the use of the above model in
deriving good configurations.
Figure 46: Comparing the analytical latency-throughput model with simulations for: a) 4-ary 3-cube cluster-4 system with message length being varied from $F = 2$ to 12 flits and b) 4-ary 3-cube inter-cluster network with cluster size being varied from $c = 1$ to 8 while message length is kept fixed at $F = 8$ flits. Lines shown without points correspond to model predictions.

5.6 Design Framework Methodology

Let us put together all the components of our supply-demand framework, as discussed earlier. For a given system size, our goal is to derive the best configuration. This configuration should satisfy packaging and demand constraints in the most cost-effective manner. We formulate this as a search problem of selecting the best configuration from amongst the various configurations possible in the design space. The problem is solved in three phases.

Phase I: Deriving Design-Feasible Configurations

The set of configurations satisfying all packaging constraints are derived in this phase. These packaging constraints, as discussed earlier, are:

- All board sizes are less than a maximum board size ($b \leq b_{\text{max}}$).
• Pinout from a board is bounded by board size and pinout density ($P < P_b$, where $P_b = p_p \sqrt{b}$, under periphery pinout technology and $P_b = p_s b$, under surface pinout technology).

• Intended channel width is supportable by channel width technology ($W = W'$, where $W' = 12, 24$, and 40).

• Pinout from a router is less than maximum supportable pinout $R$.

For a given set of packaging parameters: $a$, $b_{max}$, $p_p$, $p_s$, $W'$, $R$, and desired system size $N$, the design-feasible solutions can be easily derived through a computer-aided search. We refer to this set of design feasible configurations as the supply side.

**Phase II: Deriving Good Configurations**

The set of good configurations is a subset of the design-feasible configurations satisfying the demanded performance requirements. The demanded performance requirements refer to the upper bound on average network latency ($T_{max}$ cycles) and a minimum average throughput ($\lambda$ bits/cycle) as discussed earlier in Sec. 5.1. For ease of discussion we represent the demanded minimum average throughput in terms of $\lambda/L$ messages/cycle, where $L$ denotes the length of a message in bits. For each design-feasible configuration, using Eqn. 5.13 we derive the value of maximum average throughput that can be sustained with average message latency being less than $T_{max}$. Such a maximum value of average throughput is denoted by $m_{max}$ messages/cycle. A design-feasible system configuration offering $m_{max} \geq \lambda/L$ meets both latency and
throughput demands on performance and is therefore marked as a good configuration. Any member from the set of good configurations can be used to build a machine under packaging constraints while offering the desired performance.

**Phase III: Deriving the Best Configuration**

The final phase of the problem is to select the best configuration from the set of good configurations. Cost-effectiveness, offered maximum throughput, and scope for future scalability are three important considerations for deciding the best configuration. For each good configuration our framework provides information such as exact board size used, number of inter-board connectors required, and total volume of wires. Such information can be used by a system designer to estimate the cost of a given configuration. However, a cost comparison is dependent on the exact cost model used. Thus, the search for the configuration with lowest cost may not be a very clear-cut process. An alternative approach is to select the good configuration offering the maximum value of $m_{\text{max}}$ as the best configuration. Another factor considered is the potential of a configuration for future scalability to larger sizes to meet higher computing needs. Let us analyze this scalability aspect in choosing the best configuration.

As an example, let us consider the design-problem of building a system with $1K$ processors. Once a configuration (say $n = 4D$, $c = 4$, $b = 8$, and $W = 24$) is chosen to fabricate a machine, these system parameters are fixed. To scale a given system to a larger size more processors need to be connected. However, to have a homogeneous scaled system the values of the parameters $n$, $b$, $c$, and $W$ must be maintained. Otherwise, the system requires complete redesign and refabrication. Let $(n, b, c, W)_{1K}$
denote the set of all $n$, $b$, $c$, and $W$ values that represent good configurations to build a system of $N = 1^K$ processors. Similarly, let $(n, b, c, W)_{2^K}$ denote such a set for system size of $N = 2^K$ processors. The intersection set $(n, b, c, W)_{1^K} \cap (n, b, c, W)_{2^K}$ denotes good configurations under both $N = 1^K$ and $2^K$ processors. Such configurations can be used to build a system with $1^K$ processors which can be scaled up to $2^K$ processors while still meeting demanded performance. We emphasize on such scalability in our framework to derive best configurations.

5.7 Applying the framework

In this section we illustrate the above three phase process by deriving the best configuration to design a system with $N \approx 1024$ processors for a representative set of packaging and technological parameters. The packaging parameters used are $b_{max} = 8$, $p_p = 128$, and $R = 250$. A demanded performance of $T_{max} = 200$ cycles, $\lambda = 3.0$ bits/cycle, and a message length of $L = 192$ bits are assumed.

5.7.1 Deriving Design-Feasible Configurations

Table 6 summarizes various design-feasible configurations obtained through the framework for three different values of $W' = 12, 24$ and $40$. The results are organized in columns with respect to number of clusters per board ($b'$). For example, column 3 in Table 6 summarizes these solutions with one cluster per board ($b' = 1$). Similarly, columns 4, 5, and 6 in Table 6 present solutions obtained with allowable number of clusters per board of $b' = 2, 4$, and $8$, respectively. Each entry is either a single cluster size or a range of cluster sizes. Such a cluster size coupled with the value of
dimension \((n)\), indicated on the row of the entry, represents a feasible configuration. For example, the entry of \(c = 2\) in column 3 and the row corresponding to \(W' = 24\) and \(n = 4\) represents a feasible configuration 4D c-2. Blank entries in the table indicate that there was no valid configuration for that inter-cluster dimensionality, pinout technology, channel width, and board area. For a given value of \(W'\), the inter-cluster dimensionality was varied up to a maximum dictated by the router pinout constraint of \(R = 250\), as discussed in Sec. 5.3.4.

Table 6: Valid ranges of cluster size \((c)\) for packaging constraints of \(b_{\text{max}} = 8\), \(p_p = 128\), \(R = 250\), and three different supportable channel widths of \(W' = 12\), 24, and 40. Number of clusters/board \((b')\) is varied from 1 - 8.

<table>
<thead>
<tr>
<th>(W' = 12)</th>
<th>n</th>
<th>cluster size ((c))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(b' = 1)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(W' = 24)</th>
<th>n</th>
<th>cluster size ((c))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(b' = 1)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3 - 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(W' = 40)</th>
<th>n</th>
<th>cluster size ((c))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(b' = 1)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3 - 4</td>
</tr>
</tbody>
</table>

From Table 6 we observe that for thinner supportable channel width \((W' = 12)\), configurations with lower dimensionality are not design-feasible (hence not depicted in the table). This limits wastage of board resources as discussed in Sec. 5.4.3. Similarly, for wider channel width \((W' = 40)\), configurations with higher dimensionality are not
design-feasible in order to remain within the router pinout limit. From Table 6 it can also be observed that for wider channel width technologies ($W' = 24$ and $40$) clustered organizations ($c > 1$) are feasible while flat organizations are not. Note that the above representation of design-feasible solutions is independent of the total processors in the system. Given a system size of $N$ processors, the above representation of feasible configurations can be expanded as explained in the next paragraph. In the remaining discussion we continue the illustration of the framework for only one channel width of $W' = 24$. The framework can be similarly applied to other values of $W'$.

For each of the design-feasible configuration in Table 6 we first derive the exact inter-cluster topology to realize a system with $N \approx 1024$ processors. For example, consider the entry in Table 6 corresponding to $n = 4$, $b' = 1$, $c = 2$, and $W' = 24$. The desired number of clusters in this system with 1024 processors is $1024/2 = 512$. Let us consider selecting a 4D ($n = 4$) inter-cluster topology to interconnect 512 clusters. The closest configuration offering 4D inter-cluster is $5^3 \times 4$. The resultant system configuration has 600 clusters or 1200 processors which is more than the desired number of 1024. Without loss of generality we allow such deviations in system size up to reasonable limits. Similarly, other configurations corresponding to $W' = 24$ are also expanded. All design-feasible configurations corresponding to $W' = 24$ in Table 6 are presented in the top half (corresponding to $N = 1024$) of Table 7. It can be observed that configurations with identical values for $n, c, W'$ but different $b'$ lead to same inter-cluster topology. For example, consider the two entries in Table 6 corresponding to $n = 3$, $c = 2$, $W' = 24$ but different $b' = 2$ and 4, respectively. Both
lead to the same 8x8x8 inter-cluster topology. However, the sizes of the boards being used in the two configurations are different, \( b = b'c = 4 \) and 8, respectively. In a cost-model where board size is a factor in system cost, a careful distinction may be necessary among these configurations. However, to avoid making our study sensitive to a specific cost-model, in the remaining part of the chapter we do not make such a distinction.

5.7.2 Deriving Good Configurations

From among the design-feasible configurations presented in the left half of Table 7 we can derive a set of good configurations. Let us assume a demanded performance of \( T_{\text{max}} = 200 \) cycles, \( \lambda = 3.0 \) bits/cycle, and a message length of \( L = 192 \) bits. This leads to \( \lambda/L = 0.015 \) messages/cycle. For each configuration in Table 7, the value of maximum throughput sustainable (\( m_{\text{max}} \)) while maintaining average message latency less than \( T_{\text{max}} \leq 200 \) cycles is shown next to it. These values are derived using Eqn. 5.13. Some of the values for very low radix systems were derived through simulation experiments. This is because Agrawal's model [3], on which our model is based, does not hold for very low radix configurations. The performance plots depicting the average message latency versus average throughput for all feasible configurations to build a \( N \approx 1024 \) processor system are shown in Fig. 47. These were obtained through actual simulation experiments. These plots demonstrate similar comparative trends between configurations as derived by the analytical model. The configurations in Table 7 offering \( m_{\text{max}} \geq \lambda/L \) are good configurations and depicted in boldface. For example, the configurations 4D c-2, 4D c-3, 5D c-3, and 5D c-4 were derived to
be good.

Table 7: Design Feasible Configurations to Build Systems with $N \approx 1024$ and 4096 Processors under the Packaging Parameters $p_p = 128$, $W' = 24$, $R = 250$, and $b_{max} = 8$. Good Configurations are Shown in Boldface and these Offer $m_{max} \geq \lambda/L = 0.015$ with $T_{max} = 200$ cycles.

<table>
<thead>
<tr>
<th>Internet topology</th>
<th>cluster size ($c$)</th>
<th>Maximum Average Throughput ($m_{max}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 1024$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D: 8x8x8</td>
<td>2</td>
<td>0.0100</td>
</tr>
<tr>
<td>4D: 5x5x5x4</td>
<td>2</td>
<td>0.019</td>
</tr>
<tr>
<td>4D: 5x4x4x4</td>
<td>3</td>
<td>0.015</td>
</tr>
<tr>
<td>4D: 4x4x4x4</td>
<td>4</td>
<td>0.013</td>
</tr>
<tr>
<td>5D: 4x3x3x3x3</td>
<td>3</td>
<td>0.024</td>
</tr>
<tr>
<td>5D: 3x3x3x3x3</td>
<td>4</td>
<td>0.018</td>
</tr>
<tr>
<td>$N = 4096$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D: 13x13x12</td>
<td>2</td>
<td>0.006</td>
</tr>
<tr>
<td>4D: 7x7x7x7</td>
<td>2</td>
<td>0.012</td>
</tr>
<tr>
<td>4D: 6x6x6x6</td>
<td>3</td>
<td>0.009</td>
</tr>
<tr>
<td>4D: 6x6x6x5</td>
<td>4</td>
<td>0.007</td>
</tr>
<tr>
<td>5D: 5x4x4x4x4</td>
<td>3</td>
<td>0.015</td>
</tr>
<tr>
<td>5D: 4x4x4x4x4</td>
<td>4</td>
<td>0.012</td>
</tr>
</tbody>
</table>

5.7.3 Deriving the Best Configuration

Let us first consider the best configuration derived based on selecting the good configuration offering the maximum value of $m_{max}$. From Table 7 this best configuration is 5D c-3 ($4 \times 3^4, 3$) with a $m_{max} = 0.024$ messages/cycle. Next let us analyze the best configuration with potential to scale up to a system size of $N \approx 4096$ processors.

The design-feasible configurations and the good configurations were also derived for a larger system size with $N \approx 4096$ processors for a similar set of packaging and
Figure 47: Comparing the performance, obtained through simulation, of the design-feasible configurations shown in Table 7 to build a system with $N \approx 1024$ processors.

performance parameters. These are shown in the bottom half of Table 7. For a set of packaging parameters the set of design-feasible configurations (in terms of $n$ and $c$) are similar irrespective of size of the system being designed. However, the size of the inter-cluster needs to be larger to accommodate more clusters. For example, consider the 3D $c$-2 ($8 \times 8 \times 8, 2$) feasible configuration, shown in first row of Table 7 to build a $N \approx 1024$ system. This configuration needs to be scaled to a larger 3D $c$-2 ($13 \times 13 \times 12, 2$) feasible configuration to build a system with $N \approx 4096$ processors. However, from Result 7 we know that the inter-cluster bisection size per processor does not scale linearly with system size. This is also reflected in a fall in the value of supportable throughput $m_{\text{max}}$, derived using Eqn. 5.13, from 0.010 to 0.006 messages/cycle as shown in Table 7. Similarly, for other configurations a fall in the offered value for $m_{\text{max}}$ was observed with increasing system size. It can be observed that only one configuration, 5D $c$-3, is capable of meeting the demanded performance of $\lambda/L = 0.015$ messages/cycle to build a $N \approx 4096$ processor system. This leads us to conclude that 5D $c$-3 ($5 \times 4^4, 3$) is the only good configuration to build a system.
with 4K processors. The intersection of this set with the set of good configurations for a system with 1K processors again leads us to the 5D c-3 as the best configuration. This configuration has potential to scale up to 4K processors while still offering the minimal demanded performance.

Similar to the above illustration our framework can be applied to derive the best configuration under different sets of packaging and demand parameters. Considering the above example as a base case we also studied the impact of varying different packaging and demand parameters on design. Further results on best configurations are derived by choosing the good configuration offering maximum value of $m_{\text{max}}$.

5.8 Impact of varying packaging and demand parameters on the design process

In this section, we illustrate the impact of changing various packaging and demand parameters on the set of design-feasible and good configurations. In all following tables the derived good configurations are shown in boldface and the best configuration shown preceded by a + sign.

5.8.1 Processor and Interconnect Technology

For a given set of packaging technologies the set of design-feasible solutions is fixed. However, the set of good configurations is dependent on the demanded values of $T_{\text{max}}$ and $\lambda$. In Section 5.1.1 it was indicated that the value of $\lambda$ depends on the relative advancements in processor and interconnect technologies. We analyzed the impact of varying $\lambda$ on the set of good configurations. Table 8 shows the same design-feasible
configurations as shown in Table 7 to build a system with $N \approx 1024$ processors. The top half of Table 8 shows the good configurations (depicted in boldface) for a lower value of $\lambda = 2.0$, corresponding to $\lambda/L = 0.010$ messages/cycle. In this example it was observed that all design-feasible configurations become good. The bottom half of Table 8 similarly shows the good configurations obtained with a higher value of $\lambda = 4.0$, corresponding to $\lambda/L = 0.020$ messages/cycle. In this case only the 5D c-3 and 5D c-4 configurations were derived to be good. Under both values of $\lambda$ the best configuration, based on maximum value of offered $m_{\text{max}}$, is 5D c-3. This best configuration is depicted in Table 8 preceded by a + sign.

Table 8: Impact of varying demanded throughput ($\lambda$) on the design of a system with $N \approx 1024$ processors. Parameters $p_p = 128$, $W' = 24$, $R = 250$, and $b_{\text{max}} = 8$ are assumed.

<table>
<thead>
<tr>
<th>Internet topology</th>
<th>cluster size ($c$)</th>
<th>Maximum Average Throughput ($m_{\text{max}}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda = 2.0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D: 8x8x8</td>
<td>2</td>
<td>0.010</td>
</tr>
<tr>
<td>4D: 5x5x5x4</td>
<td>2</td>
<td>0.019</td>
</tr>
<tr>
<td>4D: 5x4x4x4</td>
<td>3</td>
<td>0.015</td>
</tr>
<tr>
<td>4D: 4x4x4x4</td>
<td>4</td>
<td>0.013</td>
</tr>
<tr>
<td>+ 5D: 4x3x3x3x3</td>
<td>3</td>
<td>0.024</td>
</tr>
<tr>
<td>5D: 3x3x3x3x3</td>
<td>4</td>
<td>0.018</td>
</tr>
<tr>
<td>$\lambda = 4.0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D: 8x8x8</td>
<td>2</td>
<td>0.010</td>
</tr>
<tr>
<td>4D: 5x5x5x4</td>
<td>2</td>
<td>0.019</td>
</tr>
<tr>
<td>4D: 5x4x4x4</td>
<td>3</td>
<td>0.015</td>
</tr>
<tr>
<td>4D: 4x4x4x4</td>
<td>4</td>
<td>0.013</td>
</tr>
<tr>
<td>+ 5D: 4x3x3x3x3</td>
<td>3</td>
<td>0.024</td>
</tr>
<tr>
<td>5D: 3x3x3x3x3</td>
<td>4</td>
<td>0.018</td>
</tr>
</tbody>
</table>
5.8.2 Increasing Supported Channel Width

We studied the impact of wider channel width of $W' = 40$ while designing a system with $N \approx 1024$ processors. The other parameters were maintained at $p_p = 128$, $R = 250$, and $b_{max} = 8$. The resultant feasible configurations are shown in Table 9. The performance plots depicting the average message latency versus average throughput for these configurations obtained through actual simulation are shown in Figure 48. The simulation results again conform to the latency-throughput trends indicated by the analytical model. Let us compare these configurations with those obtained in Table 7 for a similar design problem with $W' = 24$. It can be observed that higher dimensional systems ($n = 4, 5$) feasible earlier under $W' = 24$ are no longer feasible under $W'' = 40$. Similarly, for a given inter-cluster dimension under both channel widths, the cluster size is larger with $W' = 40$. For example, the configuration 3D c-2 is feasible with $W' = 24$ while configurations with larger cluster sizes, 3D c-3 and 3D c-4, become feasible with $W' = 40$. These results confirm the observations in Result 5. The best configuration in this case was derived as $(7 \times 7 \times 7, 3)$.

5.8.3 Increasing Board Pinout Density

As observed in Table 5 in Section 5.4.2, an increase in board pinout density makes higher dimensional inter-cluster networks feasible. Similarly, smaller cluster sizes also become feasible with fixed inter-cluster dimensionality. These impacts of increasing pinout density to $p_p = 192$ and 256 were analyzed while designing a system with $N \approx 1024$ processors. The other parameters were maintained at values used in
Table 9: Impact of increased channel width technology ($W' = 40$) on the design of a system with $N \approx 1024$ Processors. Parameters $p_p = 128$, $R = 250$, and $b_{max} = 8$ are assumed.

<table>
<thead>
<tr>
<th>Internet topology</th>
<th>cluster size (c)</th>
<th>Maximum Average Throughput ($m_{max}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W' = 40$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D: 32x32</td>
<td>1</td>
<td>0.010</td>
</tr>
<tr>
<td>2D: 23x22</td>
<td>2</td>
<td>0.007</td>
</tr>
<tr>
<td>+ 3D: 7x7x7</td>
<td>3</td>
<td>0.017</td>
</tr>
<tr>
<td>3D: 7x6x6</td>
<td>4</td>
<td>0.015</td>
</tr>
</tbody>
</table>

Figure 48: Comparing the performance, obtained through simulation, of the design-feasible configurations shown in Table 9 to build a system with $N \approx 1024$ processors.
Sec. 5.7: $R = 250$, $W' = 24$, and $b_{max} = 8$. The resultant feasible configurations obtained are shown in Table 10. We observed 3D c-2 to be a feasible configuration under $p_p = 128$ in Table 7. With increasing pinout density, $p_p = 192$, the feasibility set shifted to support a smaller cluster size, a 3D c-1 configuration, as shown in the first row of Table 10. With even higher pinout density, $p_p = 256$, a 3D topology cannot utilize the increased pinout effectively because the cluster size can not be smaller than one. Thus, a 3D configuration is no longer feasible and hence not depicted in Table 10. Although higher dimensional (n > 5) systems become supportable with increased pinout density, the maximum router pinout ($R = 250$) constrains systems with n > 5 from being feasible. The best configurations derived with $p_p = 192$ and 256 were $(6 \times 6 \times 6 \times 5, 1)$ and $(4 \times 4 \times 4 \times 4 \times 4, 1)$, respectively.

Table 10: Impact of increasing pinout ($p_p$) on the design of a system with $N \approx 1024$ processors. Parameters $W' = 24$, $R = 250$, and $b_{max} = 8$ are assumed.

<table>
<thead>
<tr>
<th>Internet topology</th>
<th>cluster size (c)</th>
<th>Maximum Average Throughput ($m_{max}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_p = 192$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D: 10x10x10</td>
<td>1</td>
<td>0.015</td>
</tr>
<tr>
<td>+ 4D: 6x6x6x5</td>
<td>1</td>
<td>0.029</td>
</tr>
<tr>
<td>4D: 5x5x5x4</td>
<td>2</td>
<td>0.019</td>
</tr>
<tr>
<td>5D: 4x3x3x3x3</td>
<td>3</td>
<td>0.024</td>
</tr>
<tr>
<td>$p_p = 256$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4D: 6x6x6x5</td>
<td>1</td>
<td>0.029</td>
</tr>
<tr>
<td>+ 5D: 4x4x4x4x4</td>
<td>1</td>
<td>0.049</td>
</tr>
<tr>
<td>5D: 4x4x4x3x3</td>
<td>2</td>
<td>0.029</td>
</tr>
</tbody>
</table>
5.8.4 Increasing the Maximum Router Pinout

The impact of increasing maximum router pinout to $R = 500$ was observed in designing a system with $N \approx 1024$ processors. The other parameters were maintained at $p_p = 128$, $W' = 40$, and $b_{\text{max}} = 8$. The resultant feasible configurations are shown in Table 11. The feasible configurations derived for a similar design problem with $R = 250$ were presented in Table 9. By comparing Tables 9 and 11, it can be observed that the maximum dimensionality of feasible configurations increased from $n = 3$ to $5$ as $R$ was raised from 250 to 500. Larger dimensional systems were also associated with larger cluster sizes: 4D configurations with $c = 6$ and 7 and 5D configuration with $c = 8$. The best configuration derived with $R = 500$ was $(3 \times 3 \times 3 \times 2 \times 2, 8)$.

Table 11: Impact of increased router pinout ($R = 500$) on the design of a system with $N \approx 1024$ Processors. Parameters $p_p = 128$, $W' = 40$, and $b_{\text{max}} = 8$ are assumed.

<table>
<thead>
<tr>
<th>Internet topology</th>
<th>cluster size $(c)$</th>
<th>Maximum Average Throughput $(m_{\text{max}})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R = 500$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D: 32x32</td>
<td>1</td>
<td>0.010</td>
</tr>
<tr>
<td>2D: 23x22</td>
<td>2</td>
<td>0.007</td>
</tr>
<tr>
<td>3D: 7x7x7</td>
<td>3</td>
<td>0.017</td>
</tr>
<tr>
<td>3D: 7x6x6</td>
<td>4</td>
<td>0.015</td>
</tr>
<tr>
<td>4D: 4x4x4x3</td>
<td>6</td>
<td>0.019</td>
</tr>
<tr>
<td>4D: 4x4x3x3</td>
<td>7</td>
<td>0.018</td>
</tr>
<tr>
<td>+ 5D: 3x3x3x2x2</td>
<td>8</td>
<td>0.027</td>
</tr>
</tbody>
</table>
5.9 Summary

The impact of varying other packaging constraints like maximum board size and surface board pinout technology was also considered. These results are available in [13]. Based on representative current and expected future technologies our analysis indicated that flat configurations may not be design-feasible under all packaging technologies. On the other hand, clustered configurations demonstrate higher potential in offering design-feasible configurations. For a wide range of technological parameters, we observed that best configurations are achieved with up to 8 processors per cluster and 3D-5D inter-cluster interconnection.
CHAPTER VI

Interplay between cluster organizations and collective communication algorithms

In previous chapters our analysis for designing processor-cluster based systems has emphasized mostly on packaging and cost-efficiency aspects [8, 16]. While deriving best topologies we have primarily emphasized on the load on network bisection and evaluated different topologies with respect to average message latency and throughput. However, the performance of a real system depends on several other factors like locality of reference, communication overheads, and performance of frequently-used communication patterns. The impact of intra-cluster organization on system performance has also been ignored.

The intra-cluster organization heavily depends on the programming model intended for a system. In order to support shared memory programming it has been shown to be beneficial to allow shared memory between processors inside a cluster [51]. However, it is not clear whether distributed memory systems need to provide shared-memory access between sibling processors (processors within a cluster)? From a naive point of view such sharing does not seem to be beneficial due to the message passing programming model. However, with a closer look it can be observed
that in the absence of such sharing the benefits of clustering to exploit locality of reference are minimal. If four processors in a processor-clustered system are placed on a single board it is natural that faster communication between these processors can be achieved by providing some portions of shared memory between them (even though higher-level communications will be message-passing). This will lead to faster message passing between these processors and the applications can take advantage of locality. Otherwise processor-clustering may not lead to any performance gain. If we believe that such shared memory access are required the next questions are: 1) which intra-cluster organization will provide the best benefit? and 2) can existing collective communication algorithms take advantage of such memory sharing to deliver better performance?

In this chapter we take on such a challenge to provide answers to the above questions. We study the interplay between intra-cluster organizations and the performance of collective communication operations, which are frequently used operations on distributed memory systems. We first analyze alternative intra-cluster organizations (star, bus, direct and crossbar) to study their respective capabilities to provide faster communication between sibling processors. We demonstrate that such faster communication can also lead to faster global operations like broadcasting. Our results indicate that bus and crossbar organizations can provide better performance improvement for broadcasting. We then demonstrate that existing collective communication algorithms (U\_mesh, designed for non-clustered systems) may not be sufficient to take advantage of faster communication between sibling processors. We
propose a new broadcast algorithm (Clus_mesh) [15] to deliver better performance on processor-clustered systems capable of outperforming U_mesh by up to 20-25%.

This chapter provides significant insight on how future processor-clusters should be designed for supporting distributed-memory paradigms.

6.1 Processor-cluster organizations and communication costs

In this section we present four different cluster organizations as shown in Fig. 49, two loosely coupled: star and direct-network and two tightly coupled: bus and x-bar. The inter-cluster network connecting clusters is kept the same when considering different cluster organizations. Let us analyze the communication delay to transfer $m$ words to a processor's own memory from the memory of another processor. This basic operation is similar to the $\text{shmem.get}$ primitive [23] supported on the Cray T3D system. In the following discussion we use parameters as summarized in Table 12. Explanation of these parameters and discussion on the different clustered organizations are detailed in [14]. The $\text{shmem.get}$ transfer across clusters requires inter-cluster message communication. For popular wormhole routing a cost penalty of $(t_s + mt_p)$ cycles can be derived for this operation independent of the cluster organization. The time for a $\text{shmem.get}$ operation from a memory inside the same cluster depends on the cluster organization. The communication delay for $\text{shmem.get}$ for transferring $m$ words in the four different cluster organizations can be derived as: star $(t_s + mt_p)$, direct $(t_s + mt_p')$, bus $(t_s' + mt_p)$, and crossbar $(t_s' + mt_p')$. Detailed derivations and
Table 12: Important Symbols Used in Paper.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>Bus bandwidth in a bus-based cluster (words/cycle)</td>
</tr>
<tr>
<td>$c$</td>
<td>Size of each processor-cluster</td>
</tr>
<tr>
<td>$i$</td>
<td>Number of injection channels per cluster to router</td>
</tr>
<tr>
<td>$t_s$</td>
<td>Startup overhead for inter-cluster messages. In loosely coupled clusters (star and direct-network) it also represents startup overhead for intra-cluster messages</td>
</tr>
<tr>
<td>$t'_s$</td>
<td>Startup overhead for intra-cluster memory transfers in tightly coupled clusters (bus and x-bar)</td>
</tr>
<tr>
<td>$t_p$</td>
<td>Propagation delay per word in inter-cluster network and also in star intra-cluster network</td>
</tr>
<tr>
<td>$t'_p$</td>
<td>Propagation delay per word inside tightly coupled clusters</td>
</tr>
<tr>
<td>$t''_p$</td>
<td>Propagation delay per word inside direct-network based clusters</td>
</tr>
<tr>
<td>$N$</td>
<td>Total number of processors in a system</td>
</tr>
<tr>
<td>$N_C$</td>
<td>($= N/c$), total number of clusters in system</td>
</tr>
</tbody>
</table>
explanations for these expressions are presented in [14]. In this paper we assume typ­
ical values of \( t_s = 200 \) to 1000 cycles, \( t'_s = 20 \) to 100 cycles, \( t_p = 10 \) to 40 cycles/word, \( t'_p \approx t''_p = 5 \) to 20 cycles/word, and \( m = 128 \) words.

![Diagram of processor-cluster organizations](image)

Figure 49: Four alternate processor-cluster organizations.

6.2 Evaluation of different processor-cluster organizations

The \( U_{-mesh} \) algorithm has been proposed in [50] as an optimal algorithm to achieve multicaasts and broadcasts in non-clustered mesh interconnected systems. To per­
form a multicast/broadcast to \((N - 1)\) destinations the algorithm orders the \( N - 1 \) destinations and source into a dimension-ordered chain and achieves the operation in \([\log_2 N]\) contention-free steps. We extend the same \( U_{-mesh} \) algorithm for broad­
casting to clustered meshes. To have contention-free steps we label the clusters in the system on a dimension ordered chain along the dimensions of the inter-cluster network. The \( c \) processors in a cluster labeled \( i \) are numbered from \( ci \) to \( c(i + 1) - 1 \).
6.2.1 Deriving the time for broadcast

We derive expressions for the time taken by the U_mesh algorithm to achieve a broadcast in clustered systems with different cluster organizations. The U_mesh algorithm naturally divides into two phases: an inter-cluster phase followed by intra-cluster phase in each cluster. In the inter-cluster phase the broadcast data is sent to one processor in each cluster. For a given system with $N$ processors and cluster size $c$, there are $N/c$ clusters. This phase requires $\lceil \log_2 (N/c) \rceil$ steps of inter-cluster message communication. Assuming the broadcast data consists of $m$ words, the time for the inter-cluster phase is $(\lceil \log_2 (N/c) \rceil) (t_s + mt_p)$.

The time for the intra-cluster phase is a function of the cluster organization. For the four different cluster organizations discussed earlier, the third column in Table 13 summarizes the time required for this phase. The basic approach for broadcasting inside a cluster is to employ, whenever possible, a U_mesh-like divide and conquer strategy. Let us derive the broadcasting time required in a star cluster. Recall that in a star cluster even an intra-cluster communication needs to go through the network router using an injection channel. Without loss of generality, in this discussion we assume $i$ and $c$ to be powers of 2 and $c \geq i$. At the beginning of the intra-cluster phase one processor in the cluster has the data. In step 1, using one injection channel, the data is transferred to a second processor. Similarly, in step 2 the two processors now having the data after step 1, concurrently transfer it to two more processors using two injection channels. In step 3 the four processors now having the data after step 2, concurrently transfer it to four more processors using two injection channels.
Similarly, we can show that in step $\log_2 i$, $i/2$ processors having the data concurrently transfer it to another $i/2$ processors using $i/2$ injection channels. Thus, after $\log_2 i$ steps, $i$ processors have the data while $c - i$ processors still have to get it. We assign one injection channel to each of the $i$ processors having the data. These $i$ processors now broadcast the data to $(c - i)/i$ processors each, using the injection channel assigned to them. Thus, this part requires $c/i - 1$ steps. The total number of intra-cluster steps used are $(\log_2 i + c/i - 1)$ and the time required is $(\log_2 i + c/i - 1)(t_s + mtp)$ cycles.

Table 13: The time required in network cycles for the inter- and intra-cluster phases of broadcasting a data of size $m$ words in a clustered mesh using the U.mesh algorithm. The number of clusters in the system is $N/c$. The time for the intra-cluster phase of the broadcast varies with the cluster organization. The relationships $i \leq c$ and $b \leq c$ are assumed.

<table>
<thead>
<tr>
<th>Cluster organization</th>
<th>Inter-cluster broadcast time</th>
<th>Intra-cluster broadcast time</th>
</tr>
</thead>
<tbody>
<tr>
<td>star</td>
<td>$(\lceil \log_2 N/c \rceil)(t_s + mtp)$</td>
<td>$(\lceil \log_2 i \rceil + \lceil c/i \rceil - 1)(t_s + mtp)$</td>
</tr>
<tr>
<td>direct</td>
<td>$(\lceil \log_2 N/c \rceil)(t_s + mtp)$</td>
<td>$\lceil \log_2 c \rceil(t_s + mtp')$</td>
</tr>
<tr>
<td>bus</td>
<td>$(\lceil \log_2 N/c \rceil)(t_s + mtp)$</td>
<td>$(\lceil \log_2 b \rceil + \lceil c/b \rceil - 1)(t'_s + mtp')$</td>
</tr>
<tr>
<td>crossbar</td>
<td>$(\lceil \log_2 N/c \rceil)(t_s + mtp)$</td>
<td>$\lceil \log_2 c \rceil(t'_s + mtp')$</td>
</tr>
</tbody>
</table>

The number of steps for broadcasting in a bus organization with a cluster bus supporting at most $b$ parallel transfers can be similarly derived to be $(\lceil \log_2 b \rceil + c/b - 1)$. The time for an intra-cluster communication in a bus organization being $(t'_s + mtp')$, as shown earlier, the time for a broadcast in this organization requires $(\lceil \log_2 b \rceil + c/b - 1)(t'_s + mtp')$ cycles. The number of steps for broadcasts in direct and
crossbar organizations, following a U-mesh-like algorithm can be easily derived to be $[\log_2 c]$.

### 6.2.2 Comparision of broadcast times

Figure 50 depicts plots comparing the time for broadcasting in a given system with a total of $N = 256$ processors for different cluster organizations. The impact of increasing cluster size from $c = 1$ to 16 processors, number of injection channels per router from $i = 1$ to 4, and system size to $N = 1024$ processors was also studied.

From Fig. 50 we observe that as the cluster size is increased, the broadcasting time in a star cluster remains constant while $c \leq 2i$, beyond which it rises sharply. This happens because until $c = 2i$, the intra-cluster broadcast can be achieved in $\log_2 c$ steps leading to a total number of $(\log_2 N/c + \log_2 c) = \log_2 N$ steps, a constant for a given system size. However, with $c > 2i$, an additional $(\lceil c/i \rceil - 1)$ steps are necessary to complete the broadcast.

For a given system size of $N = 256$ processors, an increase in cluster size leads to a fall in the number of clusters in the system. This leads to fewer steps being necessary to achieve the inter-cluster phase of the broadcast. This gets compensated by the greater number of steps in the intra-cluster network. Among the cluster organizations, the direct, bus, and crossbar offer a reduced intra-cluster communication time over the inter-cluster communication. This gain is higher in the tightly-coupled bus and crossbar organizations because $t'_{s} < t_{s}$. These organizations therefore demonstrate a substantial fall in the broadcast time as the cluster size is increased. For a bus cluster the cluster bus bandwidth can limit this gain in performance. The depicted plots in
Figure 50: Comparing the performance of U_Mesh broadcast algorithm on four different clustered organizations as the cluster size is increased from $c = 1 - 16$. The total number of processors in the system was maintained at $N = 256$ and $N = 1024$ in the left and right columns of plots, respectively. Typical values of $t_s/t_p = 10$, $t_s/t_p = 128$, $t_p/t_p = 2$, $b = 4$, and $m = 128$ flits were assumed.
Fig. 50 were derived for $b = 4$. For cluster sizes higher than 8, this limitation becomes evident from the plots as the bus organization cannot match the crossbar performance. Observations similar to those made above were also made with a larger system size of $N = 1024$ processors. From the plots we also observe that the tightly coupled bus and crossbar configurations consistently outperform the direct and star configurations by a margin of 5% to 20%. This demonstrates that tightly-coupled cluster configurations are indeed necessary to offer the faster intra-cluster communication required to fully exploit application locality and fully harness the power of processor-cluster based systems.

### 6.3 Broadcast on clustered meshes

In the last section we demonstrated that having faster intra-cluster communication leads to better performance for broadcast operations. It was recommended that clustered systems should be designed with cluster organizations like the bus or crossbar allowing processors to directly access the memory of other processors within the same cluster. In this section we demonstrate that with such differential communication costs, existing algorithms designed to be optimal under the assumption of flat communication cost may not remain optimal in clustered systems. We analyze the U_Mesh[50] algorithm for multicasting/broadcasting in terms of the required number of inter-cluster and intra-cluster steps. We propose an alternate clus.mesh algorithm that requires more overall steps but outperforms the U_Mesh algorithm under the assumption of faster intra-cluster communication.
6.3.1 Single port model with multiple injection channels

In Chapter IV it was shown that current and future router pinout technologies can support more than one injection and consumption channel from a processor-cluster to a router. With such pinout technologies $i = 2, 4$ injection/consumption channels will be commonly available in clustered systems. The Cray T3D currently supports $i = 2$ from each processor-cluster. In literature [39] injection channels have often been referred to as communication ports. In this discussion although we have multiple injection channel per cluster, we allow a processor to be injecting at most one message at a time. Thus, from a single processor's perspective ours is not a multi-port model. However, in a cluster with $i$ injection channels, up to $i$ processors within a cluster can concurrently inject a message each into the router through separate injection channels. The one-port model reflects the limitation that can arise from the limited memory bandwidth available for message injection. When a message is injected, the memory needs to be read at a rate matching the network channel speed. Otherwise, the network channels can remain under-utilized leading to a fall in network performance [12]. In this paper we assume memory bandwidth per processor to support only one message injection, that is, a single-port model. However, different processors in a cluster can inject messages concurrently since these are supported from different memories.
6.3.2 Illustrating U\_mesh in terms of inter- and intra-cluster steps

The U\_mesh algorithm achieves a contention-free broadcast in $n$-dimensional meshes. A broadcast in a system with $N$ processors is achieved in $\lceil \log_2 N \rceil$ steps. The input to the U\_mesh algorithm is a dimension-ordered chain\cite{50} of all the processors. Let us analyze the application of the U\_mesh algorithm for achieving a broadcast in a system with 9 clusters, $C0 - C9$, each having $c = 4$ processors, as shown in Fig. 51.

In the remaining discussion we denote an inter-cluster communication latency as $S$ (for slow) cycles and an intra-cluster communication latency as $F$ (for fast) cycles. We denote the latency to perform a broadcast in a cluster, when one processor within the cluster already has the data, as parameter $B$. Representative expressions for $B$ for different cluster organizations were presented in Table 13. For example, $B$ inside a crossbar based cluster requires $\log_2(c).F$. Thus, for a cluster with $c = 4$ processors, $B = 2F$ cycles. In the example broadcast in Fig. 51(a) the source processor 16 is in the center cluster, $C4$, while in the broadcast in Fig. 51(b) the source processor 0 is not in the center cluster. U\_Mesh requires 6 steps to complete the broadcast in both examples. Each arc representing a communication is numbered by the step in which it occurs and by $S$ or $F$ depicting inter- or intra-cluster communication, respectively. The completion times are derived by tracing the longest arc-path to the last destination. As derived in Fig. 51, the completion time for the broadcast with source in center cluster is $3S + 3F$ cycles, while for the broadcast with source not in the center cluster, the completion time is $4S + 2F$ cycles. All communications in the
same step are contention-free.

![Diagram](https://via.placeholder.com/150)

Figure 51: Example demonstrating the communication steps to perform a broadcast using the U.mesh algorithm in a 9 cluster linear array (1D mesh). The source in figures (a) and (b) is chosen as processor 16 and 0, respectively. Each cluster is assumed to have two injection channels. All communication steps are contention-free.

The U.mesh algorithm can also be applied to higher dimensional mesh of clusters. Figure 52 depicts the same broadcasts as in Fig. 51 on a 3x3 2D mesh of clusters. It can be shown that the communications within the same step are contention-free leading to similar broadcast times as in the case of the linear array.

### 6.3.3 Usage of injection channels by U.mesh

From Figures 51(b) and 52(b) it can be observed that U.mesh mostly uses only one injection channel from each cluster. The U.mesh broadcast can be divided into two distinct phases. In the first phase referred to by us as the inter-cluster phase, one
Figure 52: Example demonstrating the communication steps to perform a broadcast using the U\_mesh algorithm in a 3x3, 2D mesh of clusters. The source in figures (a) and (b) is chosen as processor 16 and 0, respectively. Each cluster is assumed to have two injection channels. All communication steps are contention-free and the broadcast times are similar to those obtained on a 9 cluster linear array.

processor in each cluster is reached. This is followed by the second phase, the intra-cluster phase, where the remaining processors are reached. Two injection channels are used concurrently by U\_mesh only in the special case when the source lies in the center cluster as shown in Figures 51(a) and 52(a). However, this usage is limited only to the center cluster. This leads us to the following observation:

**Observation 12** The U\_mesh algorithm for broadcasting when applied to clustered systems does not effectively use more than one injection channel from each cluster.

### 6.3.4 Illustrating clus\_mesh

We propose a new broadcast algorithm clus\_mesh for clustered systems which effectively exploits a) faster intra-cluster communication ($F < S$) and b) more injection channels from each cluster. The algorithm presented in this paper is designed for systems with two injection channels per cluster. However, a similar approach can be
used to design an algorithm for systems with four injection per processor. We first convey the main idea behind the clus_mesh algorithm using the following example.

Figure 53 depicts the communications performed to achieve broadcast in a 9 cluster linear array from processor 16 in the center cluster, C4. In the first step processor 16 performs an intra-cluster communication to involve another processor 17 in the same cluster. In step 2 processors 16 and 17 perform inter-cluster communications to processors 4 (in C1) and 28 (in C7), respectively. In step 3, both processors 4 and 28 perform an intra-cluster communication to involve another processor in their own clusters, respectively. In the same step, processors 16 and 17 perform inter-cluster communications to processors 12 (in C3) and 20 (in C5), respectively. In step 4 cluster C1 communicates with C0 and C2 concurrently, while C7 communicates with C6 and C8. The last clusters to be reached C0, C2, C6, and C8 perform an intra-cluster broadcast using two more intra-cluster steps. The total broadcast time as derived in Fig. 53 is $2S + 2F + B = 2S + 4F$ cycles (assuming $B = 2F$). Let us compare this to the $3S + 3F$ cycles for a similar broadcast using U_mesh as shown earlier in Fig. 51(a). We derive that the clus_mesh outperforms U_mesh if $S > F$.

The main idea behind clus_mesh for a system with 2 injection channels, is as follows. The first processor in any cluster to receive the broadcast data performs an intra-cluster communication to involve one more processor in the same cluster. The two processors then exploit the two injection channels to send out two concurrent inter-cluster messages in two different directions to cover two more clusters. The potential gain arises from the faster step within the cluster which allows more clusters
to be covered in lesser time. The algorithm offers contention-free performance if started from the center cluster. Thus, when the source processor is not in the center cluster an extra inter-cluster communication is required to send the broadcast data to the center cluster. The total time required in the example shown in Fig. 53, when the source cluster is not in center, is therefore $3S + 4F$ cycles. Comparing this to the $4S + 2F$ cycles for a similar broadcast using $U_{\text{mesh}}$ derived earlier in Fig. 51(b), it can be observed that the $\text{clus}_{\text{mesh}}$ outperforms $U_{\text{mesh}}$ if $S > 2F$. We also observe that the total number of steps obtained by adding the $S$ and $F$ factors may be higher for the $\text{clus}_{\text{mesh}}$ algorithm. Typically, $\text{clus}_{\text{mesh}}$ has fewer $S$ steps and more $F$ steps compared to the $U_{\text{mesh}}$ expression. However, with $S > F$ $\text{clus}_{\text{mesh}}$ has potential to offer a smaller broadcast time.

![clus_mesh broadcast algorithm on 9 clusters requiring: $3S + 4F$ cycles.](image)

Figure 53: Example demonstrating the communication steps to perform a broadcast using the novel $\text{clus}_{\text{mesh}}$ algorithm in a 9 cluster linear array (1D mesh). Each cluster is assumed to have two injection channels. All communication steps are contention-free.

### 6.3.5 The $\text{clus}_{\text{mesh}}$ Algorithm

Let us consider the algorithm for $\text{clus}_{\text{mesh}}$ on a linear array with $N_C$ clusters. The broadcast data is first sent to the center cluster. The linear array is then recursively partitioned into 9 parts. At each recursive step we apply the algorithm presented in
the example in Fig. 53 for broadcasting to 9 clusters. The algorithm in Fig. 53 being contention-free and the communications in the recursive partitions not interfering with each other, the overall algorithm is also contention-free. For partitions smaller than 9 clusters Table 14 summarizes the minimal cost for achieving a broadcast in these special cases. The clus_mesh algorithm is easily extendible to higher dimensional meshes of clusters. For example, Fig. 54 demonstrates the basic 9 partitioning in a 2D mesh. The time complexity of the clus_mesh algorithm can be easily shown to be \((S + (2S + 2F) \log_9 N_C + B)\). The time complexity for a similar broadcast using \(U\_mesh\) is \((S \log_2 N_C + B)\). Thus, clus_mesh outperforms \(U\_mesh\) only if the condition 
\((S + (2S + 2F) \log_9 N_C) < (S \log_2 N_C)\) is satisfied by the respective values of \(S\) and \(F\). We propose a hybrid clustered algorithm which for a given system size, and values for \(S\) and \(F\), evaluates the time for both clus_mesh and \(U\_mesh\) and uses the faster

<table>
<thead>
<tr>
<th>Number of Clusters</th>
<th>Source belongs to center cluster</th>
<th>Source does not belong to center cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(S + B)</td>
<td>(S + B)</td>
</tr>
<tr>
<td>3</td>
<td>(S + F + B)</td>
<td>(2S + B)</td>
</tr>
<tr>
<td>4</td>
<td>(2S + B)</td>
<td>(2S + B)</td>
</tr>
<tr>
<td>5</td>
<td>(2S + F + B)</td>
<td>(2S + F + B)</td>
</tr>
<tr>
<td>6</td>
<td>(2S + F + B)</td>
<td>(3S + B)</td>
</tr>
<tr>
<td>7</td>
<td>(2S + F + B)</td>
<td>(3S + B)</td>
</tr>
<tr>
<td>8</td>
<td>(2S + 2F + B)</td>
<td>(3S + B)</td>
</tr>
</tbody>
</table>
algorithm. While comparing the performance of U_mesh versus the new algorithm, we present results derived using the hybrid approach.

6.4 Comparison of Clus_mesh and U_mesh performance

In this section we compare the performance of Clus.Mesh and U_mesh algorithms for various system sizes, cluster sizes, and different values of $S$ and $F$.

6.4.1 Impact of increasing number of clusters

Figure 55 compares the time taken for clus_mesh and U_mesh to achieve a broadcast on a system having $N_C$ clusters. The parameters $N_C$ was varied from 2-1024 clusters to study the impact of increasing number of clusters. The experiment was performed...
for three different relative values of $S$ and $F$, $S/F = 2$, 5, and 10. As discussed earlier the clus.mesh achieves a trade-off by reducing number of $S$ steps with increased $F$ steps. This trade-off pays off when the $S/F$ is above a certain value. From Fig. 55 it can be observed that for $S/F = 2$, the clus.mesh is not able to deliver significantly better performance than U.mesh. As $S/F$ is increased to 5 the gains from clus.mesh become apparent. The relative gain over U.mesh was observed to be up to 20% for a wide range of system sizes. These gains are further increased with a higher $S/F = 10$. The relative gain over U.mesh in this case was observed to be up to 25% for a wide range of system sizes. We also observed that the gains are higher as the number of clusters is increased.

![Figure 55](image)

Figure 55: Comparing the performance of clus.mesh and U.mesh broadcast algorithms with the number of clusters in the system being increased from $N_C = 2 - 1024$. The cluster size is kept fixed at $c = 2$. The three plots depict results for three different values of inter-cluster communication step ($S$) to intra-cluster communication step cost ($F$) of $S/F = 2$, 5 and 10, respectively.
6.4.2 Impact of increasing the cluster size

We first analyzed the impact of increasing cluster size while keeping the number of clusters in the system fixed. These results are presented in Figure 56. The number of clusters was kept fixed at $N_C = 81$. It can be observed that the relative performance remains almost constant as the cluster size is increased from 2 - 16. This is because in both clus_mesh and U_mesh the cluster size only affects the time taken towards the end of the respective algorithms to achieve the intra-cluster broadcasts. This time denoted by $B$ earlier, is same for both algorithms.

![Figure 56: Comparing the performance of Clus_mesh and U_Mesh broadcast algorithms with increasing cluster size. The number of clusters is kept fixed at $N_C = 81$. An inter-cluster communication step of $S = 1000$ cycles and intra-cluster communication step of $F = 100$ cycles are assumed.](image)

We also analyzed the impact of increasing cluster size while keeping the total number of processors in the system fixed at $N = 256$. These results are presented in Figure 57. As the cluster size is increased the number of clusters in the system falls. This can leads to a fall in the relative gains of clus_mesh over U_mesh as depicted in
Fig. 57. However, we observed that for realistic cluster sizes and large system sizes as expected in the near future, clus\_mesh outperforms U\_mesh.

![Graph comparing performance of Clus\_Mesh and U\_Mesh broadcast algorithms.](image)

Figure 57: Comparing the performance of Clus\_Mesh and U\_Mesh broadcast algorithms. The impact of increasing cluster size is shown in two different systems with \( N = 256 \) and 1024 processors. An inter-cluster communication step of \( S = 1000 \) cycles and intra-cluster communication step of \( F = 100 \) cycles are assumed.

### 6.5 Summary

In this chapter we have analyzed various cluster organizations for designing processor-cluster based multiprocessor systems. Using U\_mesh broadcasting algorithm we have demonstrated the need for tightly coupled clusters to take advantage of higher integration available inside processor-clusters. With differential communication costs in the intra- and inter-cluster, we have also demonstrated that the U\_mesh algorithm does not remain optimal on clustered systems. We have proposed a better algorithm, Clus\_mesh, for broadcasting on clustered meshes.
CHAPTER VII

Conclusions and suggested future research

7.1 Summary of Research Contributions

The contributions of this research cover various aspects of design and use of clustered parallel systems. The overall objective has been to provide a general framework to design such systems in a balanced and cost-effective manner. Investigation has been carried out in the areas of packaging constraints, efficient communication and interface design, and collective communication algorithms in clustered systems. The key original contributions of this thesis are summarized below:

- Analyzing the benefits of emerging processor-cluster technology for designing high-performance and cost-effective parallel systems. Using a new design-space-graph framework which considers processor-clustering, messaging overheads, and network performance in an integrated manner, it was established that:
  1) with messaging overheads constraining performance in a system, processor clustering can be used to build a) an equal-sized system with a smaller network or b) a larger system with an equal-sized network; 2) with messaging overheads not being a constraint, a combination of processor clustering and wider channels
can be used to build a range of larger-sized systems. Chapter III emphasized this aspect.

- Developing a supply-demand framework for system design which takes into account advancements in packaging, processor, and interconnection technologies while deriving cost-effective and scalable system configurations. We demonstrated that flat configurations may not be design-feasible under all packaging technologies. On the other hand, clustered configurations demonstrate greater potential in offering design-feasible configurations. For a wide range of technological parameters, it is shown that best configurations are achieved with up to 8 processors per cluster and 3D-5D inter-cluster interconnection. These conclusions are based on analytical and simulation results presented in Chapter V.

- Proposing a deadlock-free wormhole routing scheme on direct-network based clusters. This allows the flexibility for an efficient no-absorb at interface routing for such processor-cluster organizations. This scheme is discussed in section 4.1.2.

- Demonstrating that having only a single consumption/injection channel from a network router to its cluster can become a bottleneck on system performance. To alleviate this problem a new network-cluster interface for supporting a larger number of consumption channels has been proposed. It was established that even with large cluster size, 3D inter-cluster network, fully adaptive wormhole routing, and high-degree of virtual multiplexing in the network channels, having
up to 2 to 4 consumption channels can alleviate consumption bottleneck. For other system and routing configurations, these demands were shown to be even more reasonable. We presented analytical and simulation results in Chapter IV to support these claims.

- Establishing that tightly coupled cluster organizations with shared access to memory offer faster intra-cluster communication. We demonstrated that such faster intra-cluster access in clustered systems can be exploited to design better collective communication algorithms. Based on this, a new broadcasting algorithm on clustered meshes - clus.mesh has been proposed which outperforms existing algorithms. Chapter VI discussed these issues.

Overall, this research has established the fact that the processor-cluster based approach is a feasible, dominant, and attractive option to design future large scale parallel systems. Currently some commercial systems like Cray T3D, Intel Paragon, Convex SPP are designing clustered parallel systems in an ad hoc manner. Though such systems are being built, there exists no systematic framework to guide such designs for building future large scale systems. This research directly attempts to provide such a framework. Thus, the guidelines developed in this research have immediate applicability to parallel computer industries to build cost-effective massively parallel systems.
7.2 Suggestions for future research

No research is complete at any time. During the course of this research, we have encountered several interesting problems. We have addressed some of these problems in this thesis. At this stage of thesis completion, we provide a list of suggestions for future research. Some of these can be treated as continuation to the present work. The remaining ones are long term problems and need sufficient research.

7.2.1 Short-Term problems

- Proving the concept of processor-clusters for implementing a currently popular Distributed Shared Memory (DSM) based system over the basic \( k \)-ary \( n \)-cube cluster-c framework. In this thesis we have used a combination of uniform traffic, hot-spot traffic, collective communication patterns, and representative message passing applications to validate our results. With a DSM implementation on a clustered system, we will have the flexibility of also using a large set of shared memory applications.

- In Chapter VI we discussed the broadcast operation on clustered meshes and proposed a more efficient algorithm for it. However, similar ideas can be extended to other collective communication operations like gather, reduction, and barrier synchronization. These operations can be treated as "reverse broadcast" or combination of broadcast and "reverse broadcast" operations. It will be useful to design algorithms for implementing these collective communication operation efficiently on clustered systems.
7.2.2 Long-Term problems

- Extending the study to processor-cluster systems using a MIN inter-cluster network. Besides, meshes/toruses, MINs are also commonly employed to implement parallel system interconnection networks. Typically MIN based architectures connect a processor to the interconnection network through a single link. With a single processor replaced by a cluster of processors, this single link will become a bottleneck. The cluster based MIN architectures proposed will have to solve this problem. It will also be interesting to observe how the packaging constraints affect such design.

- Investigating the advantages of extending the concept of multideestination messages [56] on clustered systems. It will allow multiple messages from different processors in one cluster to be combined into a single message and sent over the inter-cluster network. At a receiving cluster, the message will be decombined and distributed among the destined processors. Such a scheme has the potential for reducing the number of message startups seen at the source cluster and thus reducing overheads of communication, leading to better performance. It will make interesting study to investigate the architectural enhancements required to implement such a scheme with respect to the obtained gains in performance.

- Designing algorithms to map computation and data on clustered systems. The problem of mapping has been studied extensively on flat systems. The basic objective is to minimize messages and exploit locality of communication to improve
performance. In processor-cluster based systems, the intra-cluster communication is expected to be faster than inter-cluster communication. New mapping schemes need to be designed to exploit this aspect of clustered systems.
BIBLIOGRAPHY


