CLOUDWAVE: A CLOUD COMPUTING FRAMEWORK FOR MULTIMODAL ELECTROPHYSIOLOGICAL BIG DATA

by

CATHERINE PRAVEENA JAYAPANDIAN

Submitted in partial fulfillment of the requirements For the degree of Doctor of Philosophy

Dissertation Advisor: Dr. Guo-Qiang Zhang
Co-advisor: Dr. Satya S. Sahoo

Department of Electrical Engineering and Computer Science
CASE WESTERN RESERVE UNIVERSITY

August 2014
We hereby approve the thesis/dissertation of

Catherine Praveena Jayapandian

(candidate for the Ph. D. degree *).

(signed) Guo-Qiang Zhang

(Chair of the committee)

Xiang Zhang

Satya S. Sahoo

Samden D. Lhatoo

(date) June 30, 2014

We also certify that written approval has been obtained for any proprietary material contained therein.
# Contents

1 Introduction .................................................. 1
   1.1 Biomedical Big Data: Multimodal Electrophysiology Signals ........ 2
     1.1.1 The Basics of Electrophysiology ............................... 3
     1.1.2 Advantages of Electrophysiology .............................. 4
   1.2 Role of Electrophysiology in Clinical Medicine ....................... 5
     1.2.1 The Human Connectome Project ................................. 5
     1.2.2 Electroencephalography in Epilepsy ............................ 6
     1.2.3 Electrophysiology in Cardiology and Sleep Medicine .......... 7
   1.3 Challenges in Electrophysiology Data Management ...................... 8
     1.3.1 Challenge 1: Process and Store ................................. 10
     1.3.2 Challenge 2: Query and Retrieve ............................... 11
     1.3.3 Challenge 3: Visualize and Analyze ............................ 12
   1.4 Contributions and Dissertation Outline .............................. 14
      1.4.1 Organization of the Dissertation .............................. 15

2 Background and Related Work .................................... 17
   2.1 The PRISM Project ........................................... 18
      2.1.1 Multi-modality Epilepsy Data Capture and Integration System 19
      2.1.2 Use Cases for Electrophysiological Signal Analysis ........... 21
   2.2 Technology used in Cloudwave .................................. 23
<table>
<thead>
<tr>
<th>2.2.1</th>
<th>European Data Format (EDF)</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.2</td>
<td>Epilepsy and Seizure Ontology (EpSO)</td>
<td>25</td>
</tr>
<tr>
<td>2.2.3</td>
<td>JSON Data-interchange Format</td>
<td>26</td>
</tr>
<tr>
<td>2.2.4</td>
<td>Apache Hadoop Project</td>
<td>27</td>
</tr>
<tr>
<td>2.2.5</td>
<td>Agile Web Development using Ruby on Rails</td>
<td>29</td>
</tr>
<tr>
<td>2.2.6</td>
<td>HighCharts: An Interactive JavaScript Charting Library</td>
<td>30</td>
</tr>
<tr>
<td>2.2.7</td>
<td>PhysioToolkit: Physiologic signal processing and analysis APIs</td>
<td>31</td>
</tr>
</tbody>
</table>

### Cloudwave Architecture and Function 32

<table>
<thead>
<tr>
<th>3.1</th>
<th>System Overview</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Key Components and Workflow</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Cloudwave Integration to MEDCIS</td>
<td>36</td>
</tr>
</tbody>
</table>

### High Performance Data Processing Workflow using MapReduce 38

<table>
<thead>
<tr>
<th>4.1</th>
<th>Phase 1: Uploading EDFs to HDFS</th>
<th>39</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Phase 2: Processing EDF to CSF using MapReduce</td>
<td>39</td>
</tr>
<tr>
<td>4.2.1</td>
<td>EDFProcessor MapReduce Algorithm and Workflow</td>
<td>40</td>
</tr>
<tr>
<td>4.2.2</td>
<td>EDFProcessor Classes for Processing EDF files</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Results</td>
<td>45</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Processing EDF to Binary Signal Files</td>
<td>47</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Processing EDF to CSF</td>
<td>49</td>
</tr>
</tbody>
</table>

### Cloud-based Web Interface for Signal Visualization 50

<table>
<thead>
<tr>
<th>5.1</th>
<th>Cloudwave Web Interface Design Parameters</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Web Server Communication with Hadoop HDFS</td>
<td>53</td>
</tr>
<tr>
<td>5.3</td>
<td>Main Features of the Web Interface</td>
<td>53</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Application of Montages to Signals</td>
<td>54</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Multimodal Time-synchronized Signals displaying Clinical Seizure Events</td>
<td>56</td>
</tr>
</tbody>
</table>
9 Conclusion and Future Work

9.1 Future Work .................................................. 103

Appendices

A Hadoop MapReduce classes for processing EDF to CSF 105

A.1 Class: EDFProcessor ............................................. 105
A.2 Class: EDFMapper ............................................... 106
A.3 Class: EDFReducer ................................................. 107
A.4 Class: EDFFileInputFormat ................................. 107
A.5 Class: EDFRecordReader ...................................... 108
A.6 Class: EDFFileOutputFormat ............................... 109
A.7 Class: EDFRecordWriter ....................................... 109
List of Tables

4.1 Characteristics of patient enrolled in PRISM SUDEP study .... 46
4.2 Details of electrophysiological dataset used in comparative evaluation of Cloudwave ............................................. 46
6.1 Comparing Signal Data Models (CSF, EDF and MEF) ......... 68
8.1 Use of keywords in template creation ............................ 93
# List of Figures

1.1 Big Data in Biomedical Research [2] ........................................ 2

1.2 Electrophysiology is an example of ‘Big Data’ ................................ 9

1.3 Electrophysiology Data Growth in UH-CMC Epilepsy Center from 
    2011 to 2013 [81] ................................................................. 9

2.1 PRISM Workflow ................................................................. 20

2.2 Schema of European Data Format .......................................... 24

2.3 Epilepsy and Seizure Ontology (EpSO). ..................................... 26

2.4 HDFS Architecture [24] ....................................................... 28

3.1 Cloudwave Three-Layered Architecture .................................. 33

3.2 Cloudwave System Overview ............................................... 35

3.3 Cloudwave Components and Workflow ................................... 36

3.4 Cloudwave Integration to MEDCIS VISAGE Query Interface ...... 37

4.1 Uploading EDF Files to HDFS .............................................. 40

4.2 EDFProcessor MapReduce Algorithm .................................... 41

4.3 Cloudwave MapReduce Workflow ........................................ 41

4.4 Cloudwave MapReduce Example ......................................... 42

4.5 EDF Classes using Hadoop API .......................................... 44
4.6 Comparative evaluation of performance of Cloudwave for processing EDF to Binary signal files versus a standalone electrophysiological data processing application ............... 47

4.7 The average time taken by Cloudwave and the standalone application to process various categories of signal data ............... 48

4.8 Comparative evaluation of performance of Cloudwave for processing of EDF to CSF ................................. 49

5.1 The Cloudwave User Interface showing the signal visualization, signal filtering, and montage features. ..................... 52

5.2 (A) Referential Montage, (B) Bipolar Montage ....................... 54

5.3 Study, Montage and Signal Selection ............................... 55

5.4 Selection of Events .................................................. 57

5.5 Selection of Filters .................................................. 57

5.6 Total time to render signals with increasing number of signals/page for increasing length of EEG recordings. ............... 59

5.7 Average Time for loading and rendering 60-min EEG recording for increasing number of signals/page. ............... 60

6.1 EpSO defines the attribute values of complex epilepsy syndromes using OWL constructs, such as class level restrictions on object properties. ........................................ 62

6.2 The Cloudwave Signal Format consists of two segments storing the signal metadata and data values. ....................... 65

6.3 The CSF ontology annotation segment illustrating the mapping to EpSO classes. .................................................. 66
6.4 The advantages of storing data in Cloudwave storage module as compared to a desktop file system for: (a) a single ECG channel, and (b) 20 signal channels. 69

6.5 The results of comparative evaluation of CSF epoch-based and channel-based data partitioning. 70

6.6 The access time for CSF files is clearly less than the binary format EDF files for 30 seconds epoch data segments across all the six standard montages. 71

7.1 Cardiac measurement workflow implemented in Cloudwave to identify QRS complexes and compute RR intervals and instantaneous heart rate (IHR) values from ECG signal data. BPM, beats/min; EDF, European Data Format; HRV, heart rate variability. 75

7.2 (A) Cloudwave MapReduce algorithm for cardiac measurements from the ECG data. (B) Implementation of the algorithm with specialized Cloudwave classes corresponding to the Map phases and Reduce phases. EDF, European Data Format; HDFS, Hadoop Distributed File System; IHR, instantaneous heart rate. 78

7.3 Comparative evaluation of computing three cardiac measures of QRS complex detection, RR intervals, and instantaneous heart rate (IHR) values using a desktop computer and two implementations of Cloudwave over (A) 640 MB data from one European Data Format (EDF) file, (B) 3.2 GB data from five EDF files and (C) 10 min data segments (optimized for use by Cloudwave interface), and (D) scalability of multi-node Cloudwave implementation for one to four ECG channels. 80

8.1 EDF De-identification and Normalization 83
8.2 EDF Editor Workspace Overview. 1) Menubar, 2) Toolbar, 3) Task Navigator, 4) Work area, 5) Source File and Task Summary information, 6) Log and Incompliance area, 7) Status bar . . . . . . . . . . . . 85
8.3 Selecting EDF Files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
8.4 EDF Identity Attributes . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
8.5 Applying Identity Attributes . . . . . . . . . . . . . . . . . . . . . . . 90
8.6 Applying Identity Attribute Template . . . . . . . . . . . . . . . . . . 91
8.7 EDF Signal Attributes . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
8.8 Applying Signal Attributes . . . . . . . . . . . . . . . . . . . . . . . . 96
8.9 EDF Signal Attributes Template . . . . . . . . . . . . . . . . . . . . . . 98
8.10 Tools: Select Find EDF Header Errors . . . . . . . . . . . . . . . . . . 99
8.11 Tools: View Header Incompliances to EDF Specification . . . . . . . . 100
8.12 Parameter Selection for EDF Error Fix Component . . . . . . . . . . . 101
Acknowledgements

I thank God for His favor and courage during the last decade, through every step of my learning and advancement. This is truly a humbling experience to mature in trust that He will direct my steps.

I am so honored and privileged to be mentored by my advisor Dr. Guo-Qiang Zhang. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. Dr. Zhang has created an atmosphere of learning, collaboration, responsibility and leadership in all aspects of work. He has provided the research staff and students a creative and productive environment with start-of-art technologies and cloud computing resources. This infrastructure has been the foundational motivation for my dissertation.

I am also thankful to Dr. Satya S. Sahoo, my research co-advisor, whose enthusiasm and persistence for innovation was contagious and motivational for me, even during tough times in the Ph.D pursuit. Dr Sahoo’s insights have helped shape my research. I am extremely happy to work with Dr. Sahoo on several publications.

For this dissertation, I would also like to thank my committee members Dr. Samden Lhatoo and Dr. Xiang Zhang for their time, interest, and helpful comments. I would like to sincerely thank Dr. Samden Lhatoo for his leadership in the PRISM project. His vision for treatment and cure of epilepsy, and his passion for patient care has inspired me truly. Without his support of integrated informatics in this domain, my dissertation would not have been successful.

The members of Dr. Zhang’s group have contributed immensely to my personal and professional time at Case Western Reserve University. The group has been a source of friendships as well as good advice and collaboration. I am especially grateful for the fun group of original group members, who did coursework with me in grad school: Remo Mueller, Van Ann Tran, Fangping Huang, and Jacek Szymanski. I also want to acknowledge my group members who have done
foundational work for my thesis and co-authors on my publications, Zendrix Ng, Sulabh Jain, Meng Zhao, Chien-Hung Chen and Annan Wei for their enthusiasm, intensity and willingness to do user and performance evaluations. Other past and present group members that I have had the pleasure to work with or alongside of are grad students: Dr. Licong Cui, Dr. Lingyun Luo, Joseph Teagno, Priya Ramesh, Shiqiang Zhao, Jeno Moses, Gang Shu, Wei Wang, Yang Chen, and Yash Gunapati. I want to thank the Dr. Zhang’s staff who have always very supportive: Jeralynne Androski, Sridevi Cherukuri, Mike Warfe, and Hadrian Djobari.

Lastly, I would like to thank my family for all their love and encouragement. For my parents who raised me with a love of science and supported me in all my pursuits. For my siblings, who have constantly motivated me for innovation. For my three children, who were very understanding and patient at all times of my study. For my extended family members who have always been considerate of my well being. And most of all for my loving husband for his faithful support during the whole tenure of this program. Thank you.

My academic research at Case Western Reserve University was funded through grants from the NCRR (NCRR-94681DBS78), CTSC (NIH UL1-RR024989), NIH-NINDS (P20NS076965-01), and NIH-NHLBI (R24HL114473-01A1).
List of Abbreviations

- AJAX: Asynchronous JavaScript and XML
- AWS: Amazon Web Services
- BRAIN: Brain Research through Advancing Innovative Neurotechnologies
- CARMEN: Code analysis, Repository and Modeling for e-Neuroscience
- CRCNS: Collaborative Research in Computational Neuroscience
- CSF: Cloudwave Signal Format
- EEG: Electroencephalography
- ECG/EKG: Electrocardiography
- ECoG: Electrocorticography
- EDF: European Data Format
- EMG: Electromyography
- EMU: Epilepsy Monitoring Unit
- EOG: Electrooculography
- EOS: Earth Observing System
- EpiDEA: Epilepsy Data Extraction and Annotation
- EpSO: Epilepsy and Seizure Ontology
- EP: Evoked Potential
- ERPs: Event-Related Potentials
• fMRI: Functional Magnetic Resonance Imaging
• HCP: Human Connectome Project
• HDF: Hierarchical Data Format
• HDFS: Hadoop Distributed File System
• HEDP: Hadoop Electrophysiological Data Processing
• HIPAA: Health Insurance Portability and Accountability Act
• HITECH: Health Information Technology for Economic and Clinical Health
• HRV: Heart Rate Variability
• iEEG: Intracranial Electroencephalography
• IEEG: International Epilepsy Electrophysiology Portal
• IHR: Instantaneous Heart Rate
• ILAE: International League Against Epilepsy
• INCF: International Neuroinformatics Coordinating Facility
• JSON: JavaScript Object Notation
• MAF: Multiscale Annotation Format
• MEDCIS: Multi-modality Epilepsy Data Capture and Integration System
• MEF: Multiscale Electrophysiology File
• MEG: Magnetoencephalography
• MRI: Magnetic Resonance Imaging
• MVC: Model View Controller
• NCS: Nerve Conduction Studies
• NIH: National Institute of Health
• NLM: National Library of Medicine
• NINDS: National Institute of Neurological Disorders and Strokes
• OPIC: Ontology-driven Patient Information Capture
• OWL: Web Ontology Language
• PET: Positron Emission Tomography
• PLMD: Periodic Limb Movement Disorder
• PRISM: The Prevention and Risk Identification of SUDEP Mortality
• PSG: Polysomnography
• REM: Rapid eye movement
• REST: Representational State Transfer
• RoR: Ruby on Rails
• SPECT: Single Positron Emission Computed Tomography
• SUDEP: Sudden Unexpected Death in Epilepsy Patients
• SVA: Signal Visualization and Analysis
• UH-CMC: University Hospital - Case Medical Center
• VISAGE: VISual AGgregator and Explorer
• XML: extended Markup Language
Cloudwave: A Cloud Computing Framework for Multimodal Electrophysiological Big Data

Abstract

by

Catherine Praveena Jayapandian

Multimodal electrophysiological data, such as electroencephalography (EEG) and electrocardiography (ECG), are central to effective patient care and clinical research in many disease domains (e.g., epilepsy, sleep medicine, and cardiovascular medicine). Electrophysiological data is an example of clinical ‘big data’ characterized by volume (in the order of terabytes (TB) of data generated every year), velocity (gigabytes (GB) of data per month per facility) and variety (about 20-200 multimodal parameters per study), referred to as ‘3Vs of Big Data.’ Current approaches for storing and analyzing signal data using desktop machines and conventional file formats are inadequate to meet the challenges in the growing volume of data and the need for supporting multi-center collaborative studies with real-time and interactive access. This dissertation introduces a web-based electrophysiological data management framework called Cloudwave using a highly scalable open-source cloud computing approach and hierarchical data format. Cloudwave has been developed as a part of the National Institute of Neurological Disorders and Strokes (NINDS) funded multi-center project called Prevention and Risk Identification of SUDEP Mortality (PRISM).

The key contributions of this dissertation are:

1. An expressive data representation format called Cloudwave Signal Format
(CSF) suitable for data-interchange in cloud-based web applications;

2. Cloud based storage of CSF files processed from EDF using Hadoop MapReduce and HDFS;

3. Web interface for visualization of multimodal electrophysiological data in CSF; and

4. Computational processing of ECG signals using Hadoop MapReduce for measuring cardiac functions.

Comparative evaluations of Cloudwave with traditional desktop approaches demonstrate one order of magnitude improvement in performance over 77GB of patient data for storage, one order of magnitude improvement to compute cardiac measures for signal-channel ECG data, and 20 times improvement for four-channel ECG data using a 6-node cluster in local cloud. Therefore, our Cloudwave approach helps addressing the challenges in the management, access and utilization of an important type of multimodal big data in biomedicine.
Chapter 1

Introduction

The unprecedented rate of data collection across scientific, business, and social networking domains is transforming research, education, and decision-making through data-driven insights and knowledge discovery tools. This research paradigm provides significant ‘added value’ and is complementary to traditional hypothesis-driven approaches [88]. Data-driven approaches are taking advantage of the increasing volume of data generated at a rapid rate in a variety of modalities. Datasets characterized by 3Vs of variety, velocity and volume are called ‘big data’. Big data presents a number of computational challenges that include efficient capture, curation, storage, sharing, query, analysis and visualization of high-throughput large volumes of heterogenous datasets [107]. The healthcare domain is currently facing a number of these data management challenges.

There is a growing need to adopt an approach of using big data to derive healthcare intelligence through near real-time processing of healthcare data to support preventive care, personalized medicine, and improved treatment outcome [86]. Healthcare big data are exemplified by electronic health records, published literature, electrophysiology, imaging, proteomics and genomics data as shown in Figure 1.1. Biomedical big data are generated and shared by multiple institutions.
with differing business logic and data formats. Also these big data applications need to preserve patient privacy and ensure easy data exchange when data becomes publicly available for collaboration [81].

1.1 Biomedical Big Data: Multimodal Electrophysiology Signals

Large-scale, high-resolution electrophysiological recordings are used in the diagnosis of neurological disorders across multiple disease domains, such as epilepsy and sleep medicine [18]. These recordings that monitor the electrical activity up to single neuron spatial resolution pose several challenges for data acquisition, storage, analysis and visualization. Using conventional data representation formats,
the size of such recordings can be as large as terabytes (TB) for an EEG study. Also,
traditional storage technologies circumvent the big data challenges by reducing the
duration, number of channels, and resolution of recordings [8], which represents a
lost opportunity.

1.1.1 The Basics of Electrophysiology

Neurons are specialized cells for the integration and propagation of electrical events
among each other as well as with muscles and other end organs [58]. An under-
standing of electrophysiology, the science that pertains to the flow of ions in bio-
logical tissues [105], is fundamental to study the function and dysfunctions of the
human nervous system. The brain, heart and skeletal muscles are main sources
of electric and magnetic fields that can be recorded and the resulting patterns can
give insight into the ailments of a patient. Electrophysiology has a very important
role in ensuring accurate clinical diagnoses for investigation of abnormal electrical
signals in the body’s tissues. It provides quantitative data to clinicians, supporting
diagnostic processes and evaluating treatment success [104].

Classical electrophysiology techniques involve placing electrodes into biologi-
cal tissue generating intra- and extra- cellular recordings to measure the electro-
magnetic signals of the body [84]. These techniques are named according to the
data being measured and the anatomical locations of the sources, such as EEG
for measuring the brain activity through the surface of the brain, EKG for heart,
electromyography (EMG) for skeletal muscles, electrooculography (EOG) for eyes,
magnetoencephalography (MEG) for measuring the magnetic fields produced by
brain’s electrical activity, polysomnography (PSG) for recording biophysiological
changes during sleep, nerve conduction study (NCS) for measuring the electrical
conduction velocity of the nerves in the body, event-related potential (ERPs) to lo-
cate the precise position in the brain neurons that have been activated in response
to sensory, motor or cognitive events, and evoked potential (EPs) for measuring activity in application to an external stimulus [3]. EEG and ERP are most often used to diagnose brain diseases such as epilepsy and dementia, which cause obvious abnormalities in EEG recordings [7]. It is also used to diagnose brain lesions, sleep disorders, coma, encephalopathies, and brain death [100]. EEG is the first-line method for diagnosis for tumors, stroke and other focal brain disorders. EMG and NCS are effective in diagnosis of essential tremors [63], spasticity, multiple sclerosis, diabetic polyneuropathy [109] and Parkinson’s disease [66]. EKG is used for diagnosis of general symptoms in myocardial infarction, pulmonary embolism, cardiac murmurs, syncope, seizures and cardiac dysrhythmias [64].

1.1.2 Advantages of Electrophysiology

Electrophysiology continues to be a valuable tool for research and diagnosis due to significantly lower hardware costs for recording as compared to most other techniques [98]. Electrophysiology is widely used due to the high temporal resolution (order of milliseconds rather than seconds) and sampling rates ranging from 250 to above 20,000 Hz [23]. These techniques are relatively tolerant to subject movement [77], silent, do not involve high intensity magnetic fields [85] and exposure to radioligands [110], unlike most other neuroimaging techniques. Electrophysiology techniques, such as EEG and EMG are noninvasive and favorable in subjects who cannot make a motor response [27], and powerful for tracking brain changes during different phases of life. However, noninvasive methods have low spatial resolution and poor signal-to-noise ratio and can be improved by adding more electrodes for recording [96]. High spatial and temporal resolution recordings are obtained invasively using intracranial EEG (iEEG) otherwise referred to as electrocorticography (ECoG). ECoG offers a temporal resolution of approximately 5 ms and a spatial resolution of 1 cm [103]. Electrophysiology combined with anatomi-
1.2 Role of Electrophysiology in Clinical Medicine

Clinical utilization of electrophysiology is limited mainly due to large accumulation of data that needs thorough analysis and interpretation - a highly time-consuming task, and still depend overwhelmingly dependent on the experience of skilled physiologists who diagnose a variety of neurological disorders by visual assessment [55]. However, in the recent years, there has been a growing interest for electrophysiology in neuropathology due to great expansion in the knowledge about the electrical activity in various organs, development of acquisition hardware and precise methods of electrophysiological evaluations, and insights into sensory, perceptual and cognitive processes for treatment of disorders in certain kinds of patients where verbal cooperation is impossible [20].

1.2.1 The Human Connectome Project

The goal of the NIH Human Connectome Project is to build a ‘network map’ that will shed light on the anatomical and functional connectivity within the healthy human brain, as well as to produce a body of data that will facilitate research into brain disorders such as dyslexia, autism, Alzheimer’s disease, and schizophrenia [39]. Electrophysiology plays a important role in the Human Connectome Project (HCP) providing a window onto neurological processes underlying sensory, motor, and cognitive functions at a temporal scale inaccessible to fMRI [97]. MEG and EEG respectively detect external magnetic fields and scalp potentials arising from neuronal activity within the brain with millisecond-level temporal resolution. Despite the limited spatial resolution, the richness of temporal information obtained...
by MEG/EEG enables assessment of how brain rhythmical activity relates to resting and task-evoked brain connectivity [97].

1.2.2 Electroencephalography in Epilepsy

Electrophysiology has been a tool for understanding Epilepsy almost from the discovery of EEG in the early twentieth century [5]. Epilepsy is the most common serious neurological condition characterized by recurrent seizures [49]. Epilepsy can be caused by different conditions that affect a person’s brain, such as stroke, head trauma, complications during childbirth, infections and genetic disorders that may generate abnormality in brain wiring along with a combination of imbalance of nerve signaling chemicals called neurotransmitters [49]. Noninvasive scalp EEG and invasive intracranial EEG are the most accurate methods for localizing onset of seizures, monitoring for seizure progression and interictal activity between seizures [108]. EEG has been used to identify seizures that begin near structural abnormalities of the brain regions. EEG can detect multiple seizure types with different ictal patterns even though the subject has only a single lesion on imaging. EEG can also detect focal seizures showing no abnormality in imaging [5]. Therefore, EEG is called the gold standard for monitoring epileptic seizures [67].

The International Epilepsy Electrophysiology Portal (IEEG) is a NINDS funded initiative that seeks to advance research towards the understanding of epilepsy by providing a platform for sharing intracranial data, tools and expertise between researchers [99]. The IEEG-Portal stores all electrophysiology datasets using the Multiscale Electrophysiology File format (MEF). This file-format enables the storage of high-bandwidth multichannel data using lossless compression [9].

The PRISM project is a NINDS initiative to advance hypothesis-driven, prospective, multi-center research into sudden unexpected deaths in epilepsy patients (SUDEP). SUDEP accounts for 1% of epilepsy deaths every year. [91]. This initiative provides
a platform to investigate and quantify seizure induced brainstem dysfunction by collecting and analyzing multi-modal physiological seizure data including EEG, EKG, autonomic, cardiovascular, respiration, sleep, endocrine and evoked potential features along with phenotypic and clinical data obtained from epilepsy monitoring unit discharge summaries [59].

1.2.3 Electrophysiology in Cardiology and Sleep Medicine

Cardiac Electrophysiology is a branch of the medical specialty of clinical cardiology and is concerned with the study and treatment of rhythm disorders of the heart. These recordings are performed invasively or non-invasively consisting of spontaneous electrical activity, as well as cardiac responses to programmed electrical stimulation to assess arrhythmias, elucidate symptoms, evaluate abnormal electrocardiograms, assess risk of developing arrhythmias in the future, and design treatment [102].

Polysomnography is a comprehensive recording of the biophysiological changes that occur during sleep and is used to diagnose many types of sleep disorders including narcolepsy, idiopathic hypersomnia, periodic limb movement disorder (PLMD), REM behavior disorder, parasomnias, and sleep apnea. It is usually performed at night, when most people sleep. The PSG monitors many body functions including brain (EEG), eye-movements (EOG), muscle activity or skeletal muscle activation (EMG) and heart rhythm (ECG) during sleep [106].

National Sleep Research Resource (NSRR) is a National Institute of Health (NIH) funded initiative of the National Heart, Lung, and Blood Institute (NHLBI). NSRR aims to establish a comprehensive, easily accessible and well-annotated national repository of sleep data. This project will make data from more than 50,000 sleep studies available to sleep researchers across the country [38]. The PhysioNet project offers free web access to large collections of recorded physiologic signals (Phys-
ioBank) and related open-source software (PhysioToolkit) to stimulate investigations in the study of cardiovascular and other complex biomedical signals. It currently includes databases of multi parameter cardiopulmonary, neural and other biomedical signals from subjects with health implications including life-threatening arrhythmias, congestive heart failure, sleep apnea, neurological disorders, and aging [22].

1.3 Challenges in Electrophysiology Data Management

Electrophysiology is a data intensive domain and poses several data management challenges. By addressing these challenges for such large scale recordings, we can expand our understanding of electrical activity of neurons at very high spatial and temporal resolution [61]. The University Hospital-Case Medical Center (UH-CMC) Epilepsy Monitoring Unit (EMU) generates about 5-10 GB of data for a single patient visit. About 100-150 patients are admitted every year in a single EMU [53] generating 5TB of data each year. The number of signals in each recording varies from 20-200 channels with high temporal resolution of 200 samples/sec as shown in Figure 1.2.

Also Figure 1.3 shows the steady growth of data in the UH-CMC Epilepsy center from 2011 to 2013 [81] generating about 10TB in 2 years. Therefore electrophysiology is an example of ‘clinical big data’ characterized by high-throughput or velocity, large volume, and diverse variety. In the next 3 sections, we explain the challenges in existing data management systems for processing and storing such large datasets, querying and retrieving clinically relevant signal segments, and analyzing and visualizing these datasets from multiple centers for collaborative research.
Figure 1.2: Electrophysiology is an example of ‘Big Data’

- Multimodal: 20-200 channels
- High Temporal Resolution: 200 samples/sec
- Diverse Variety
- High Velocity
- Large Volume

Each patient visit: 5-10GB
Number of patients/yr: 100-150
Total data/year: ~5TB

Figure 1.3: Electrophysiology Data Growth in UH-CMC Epilepsy Center from 2011 to 2013 [81]
1.3.1 Challenge 1: Process and Store

Modern recording devices generate large amounts of electrophysiology data at a rapid pace. These datasets are massive in volume and need to be hosted on remote storage devices to be accessed by signal analysis and visualization applications. Currently in many labs, archiving and long-term storage of data are done locally on DVDs or externally hard drives. Because of this, old data is practically never used for further analysis. Also this data is not protected from hardware failures. Such traditional storage methods hinder the spread of scientific knowledge [26].

Also conventional data formats, such as European Data Format (EDF), store multimodal signals in a single file generating very large datasets (100s of GB). These data formats are not designed for storing very large multimodal datasets and accessing subsets of interested signals. Over the last two decades, there has been a competing market for development of tools and data formats to facilitate access of remote data and partitioning data into smaller segments. Some of the key vendors are Compumedics [41], Nihon Kohden [48], Respironics [50], DataXpress [47], EEGLAB [10], eConnectome [25], MANTA [17], TRENTOOL [60], ErgExplorer [92], and FieldTrip [76]. However, developments of such tools results in incompatible data formats between vendors making it difficult to exchange data, which then resulted in researchers spending lot of time and effort converting data between different formats for scientific collaboration.

Also recent big data initiatives envision the deployment of big data management platforms for different types of neurological data. The ConnectomeDB informatics infrastructure provides a customized installation of XNAT for storing imaging data, clinical evaluations, behavioral data and electrophysiology in the Human Connectome Project [97]. The International Neuroinformatics Coordinating Facility (INCF) has an inventory of electrophysiology data sharing initiatives, such as CARMEN [40], CRCNS [31], G-Node [44], and Neurodatabase [34]. The
IEEG Portal aims to allow multiple participants to upload de-identified patient data, which will be stored on Amazon Simple Storage Service (S3) and further analyze the datasets using desktop tools. IEEG Portal also envisions to provide a web-based interface for MEF datasets, in addition to the IEEG Toolkit that can be downloaded to interoperate with MATLAB based tools [9].

1.3.2 Challenge 2: Query and Retrieve

Electrophysiological big data do not have an associated schema that represents the modeling semantics of the data. A conceptual model, such as domain ontology, is required by electrophysiology big data applications to represent terms, relations between terms, define constraints to support domain semantics, support efficient storage, and fast retrieval of data. Existing data representation formats were not designed for use in big data applications since their storage schema does not support efficient data partitioning. For example, EDF is a widely used signal representation model that stores the signal values in binary format that need to be converted into numeric values for use by signal query and visualization applications. In addition, many signal tools use data corresponding to a single channel (for example, EEG or ECG), but an EDF file stores data from all channels as interleaved segments. Also EDF does not store clinical events along with the data, which makes it difficult to query EDF files for data corresponding to these events. Users cannot access segments of signal from remote storage using conventional data representation formats like EDF [51].

In the recent decade, there are many research initiatives for representing biomedical signals using hierarchical data models that are designed to organize very large amounts of numerical data, that can be compressed for network transfer. BiosignalML is a new XML standard for encoding and storing of biomedical time-varying signals (EEG, EMG, ECoG, EOG, BP, etc.) [43]. Neo is a Python object model for
neuroscience data [19]. Neuroshare is an initiative to develop a standard for accessing neurophysiological data from any vendor’s acquisition device or software [90]. The Neurophysiology Data Translation Format (NDF) provides a vendor-independent mechanism for translating between raw and processed neurophysiology data in the form of time and image series [30]. MEF was developed for management of large-scale intracranial EEG recordings. Hierarchical Data Format (HDF4/HDF5) are also emerging hierarchical data models for very large and complex datasets, such as data of Earth Observing System (EOS) in the order of petabytes [37].

An expressive signal representation format that uses domain ontology as a conceptual model is required for

- representing multi-modal electrophysiological time-series data;

- partitioning data to facilitate querying large volumes of data from remote distributed storage; and

- accessing data over the internet by remote web clients implementing the web interface [51].

1.3.3 Challenge 3: Visualize and Analyze

The increasing need for multi-center clinical research studies exacerbates ‘Big Data’ management challenges by introducing the need to share, interoperate, and integrate signal data in real time. In addition, collaborative access to electrophysiological data requires reconciling heterogeneous data formats, cross-platform applications, and integrated environment that keep track of process used to generate reproducibility. Two approaches for developing these tools are: (a) Stand-alone tools that need to be downloaded and installed after appropriate configuration of
the host computer, and (b) Web-based tools that are accessible through ubiqui-
tous Web browsers (e.g., Internet Explorer, Google Chrome, and Mozilla Firefox).
Stand-alone tools for signal visualization and analysis are usually proprietary soft-
ware applications or have been developed and extended in an ad-hoc manner with
minimal documentation.

Traditional approaches to signal data management involving standalone tools
(e.g. Nihon Kohden [48], and Lifelines iEEG [33]) are not suitable for collabora-
tive research. Researchers face a number of issues in using standalone tools for
managing large scale, multi-modal electrophysiological data, such as:

1. No support for synchronous visualization and interaction with shared elec-
trophysiological datasets across institutions by multiple researchers;

2. Limited re-usability of tools across different institutions and projects due to
heterogeneous computing environments, such as operating systems, hard-
ware configuration, and software libraries; and

3. Difficulty in maintaining software deployed at multiple sites over the life
cycle of the tool/application [52].

In contrast, Web-based applications not only address the above challenges with
easy accessibility through the use of ubiquitous Web browsers (e.g. Microsoft In-
ternet Explorer), but also have the ability to be transparently integrated with cloud
computing resources to support extreme scalability, high fault tolerance, and high
rate of service availability with low cost [79]. Scientific data management tools
are rapidly adopting the cloud computing paradigm, which involves Web-based
access to both storage and computing resources [75]. The cloud-computing infra-
structure is also ideally suited for managing ‘big data’ in clinical settings with strict
data security and accessibility features. For example, Amazon Web Services (AWS)
and Microsoft Azure platform provide a reliable, scalable, and inexpensive computing platform ‘in the cloud’ that can support health care customer applications in a manner consistent with HIPAA and HITECH [28] [29]. The use of Web-based signal visualization and analysis tool is ideally suited for multi-center clinical studies with multi-user collaborations [53].

1.4 Contributions and Dissertation Outline

This dissertation presents Cloudwave, a big data management framework for large multimodal time-series datasets, such as electrophysiology. ‘Cloudwave’ as the name denotes, is for storing massive time-series signal ‘waveforms’, such as EEG, EKG, and BP in ‘cloud’ storages. The key contributions of the Cloudwave framework are:

1. Provide *scalable storage* solutions for large and increasing volumes of data. Cloudwave stores data on high-performance distributed file systems that are reliable, fault tolerant and high availability. Storage can be added on the fly to enable storage of increasing data volumes. Cloudwave also provides parallel computing approach using MapReduce programming model for processing large multimodal signal files and store them in simpler data representation. A single large file is split into smaller signal files for fast network transfer;

2. Use *domain ontology* as conceptual model for partitioning large datasets for query and storage. Cloudwave defines a JSON-based signal format that defines the schema for metadata (patient and recording information, signal metadata, clinical event annotations) and signal data in time-series format. This data model uses domain ontology to facilitate query, data partitioning and reconciling event label heterogeneity;
3. Web based access to high performance distributed data stores. Cloudwave provides a Web interface for signal visualization, query and analysis. The web application is a single point of access to Cloudwave for uploading and retrieving electrophysiological data; and

4. Computational processing using Mapreduce for signal storage and analysis. Cloudwave provides parallel computational classes using MapReduce for electrophysiology signal analysis, such as cardiac measures for ECG signal.

Cloudwave data processing and analysis APIs can be extended to store time-series data in other domains such as control engineering, weather forecasting, pattern recognition, mathematical finance, communications engineering, and econometrics.

1.4.1 Organization of the Dissertation

The dissertation is organized into the following chapters:

- **Chapter 2** describes the background of Cloudwave and how it integrates with the components of the PRISM project, and also describes the various methods and materials used by the Cloudwave framework

- **Chapter 3** presents the architecture and functions of key components of the Cloudwave framework.

- **Chapter 4** describes the Cloudwave Signal Format for storing large scale multimodal time series waveforms and ontology-driven annotations.

- **Chapter 5** describes the Cloudwave Signal Library for processing EDF files using parallel computing approach for storing on distributed data stores on the cloud.
• **Chapter 6** describes the Cloudwave Signal Interface for visualization of multimodal time series data.

• **Chapter 7** describes the Cloudwave Signal Analytics service for time and frequency based analysis on time series signals.

• **Chapter 8** enumerates the preprocessing tools for EDF files for de-identification and curation of metadata of these multimodal datasets.

• **Chapter 9** concludes this dissertation and discusses future work and possible improvements.

The research presented in this thesis and related work has been presented in conference papers [52], [53], [51], [54] and journal papers [81], [82]. The JAMIA article was selected as Editor’s choice for the special issue and was highlighted in the press with a story posted in the ‘Cardiovascular Business Magazine’ May 10, 2014 issue [11].
Chapter 2

Background and Related Work

In many critical care and neurological monitoring applications, large volumes of physiological data, including EEG from scalp and implantable intracranial electrodes, pulse oximetry (SpO2), ECG, respiratory and sleep signals are collected. This rapidly growing volume of multimodal electrophysiological signal data is playing a critical role in patient care and clinical research across multiple disease domains, such as epilepsy and sleep medicine. Data management challenges are increasing with improvements in neuroimaging techniques, the integration of data from high-throughput ‘omics pipelines, and the increasing availability of multi-channel neurophysiologic datasets [82]. In this dissertation, Cloudwave addresses the electrophysiology data management needs in the epilepsy disease domain for advancing research of SUDEP and can be extended to other disease domains.

This chapter introduces the data management challenges in PRISM project. The chapter also provides description of materials and methods used in Cloudwave development, such as the Epilepsy and Seizure Ontology (EpSO), EDF, and Apache Hadoop platform. The chapter also provides an overview of related work for electrophysiology big data management.
2.1 The PRISM Project

The PRISM project, introduced in Chapter 1 is a multi-center study to examine the risk factors underlying SUDEP. SUDEP is defined as a sudden, unexpected, witnessed/un-witnessed, non-traumatic, and non-drowning death of persons with epilepsy [72]. Its causes are not well understood and there is no effective prevention [89]. PRISM is part of the ‘SUDEP Center Without Walls’ initiative. Multiple EMUs need to collaborate, share data for building a large cohort of potential SUDEP patients. The project involves the EMUs at UH-CMC, Ronald Reagan Medical Center (University of California, Los Angeles), Northwestern Memorial Hospital (NMH, Chicago), and National Hospital for Neurology and Neurosurgery (NHNN, London). An informatics infrastructure is being developed for the needs of PRISM project to manage data from all participating EMUs, including:

- A standardized tool to enter patient information at different points of care;
- An integrated signal processing tool that allows clinicians to seamlessly interface between signal data and patient information;
- An epilepsy-focused natural language processing (NLP) tool to extract structured information from clinical free text in patient records; and
- A secure query environment to identify patient cohorts across multiple study centers.

These informatics tools are underpinned by a domain ontology that is also closely aligned with existing epilepsy classification systems [82].
2.1.1 Multi-modality Epilepsy Data Capture and Integration System

Multi-modality Epilepsy Data Capture and Integration System (MEDCIS) [111] is a novel, integrated, multi-modal data acquisition and management system for collection of large scale, multicenter, prospective cohort of epilepsy patients undergoing seizure monitoring. MEDCIS creates a prospective surveillance registry of SUDEP, and also establish capability for comprehensive comparative studies of SUDEP and near-SUDEP cases versus cohort survivors to ascertain differences in clinical epilepsy and multi-modal physiological seizure data including EEG, EKG, autonomic, cardiovascular, respiration, sleep, endocrine and evoked potential features in order to characterize and quantify seizure induced brainstem dysfunction. A SUDEP brain bank and genetics database is also envisioned to be established within MEDCIS adding to an existing, substantial collection of material to investigate genetic influences and serotonergic brain dysfunction [42]. Figure 2.1 shows the key components and workflow of MEDCIS.

Patient information is captured at multiple stages during a patient’s stay in the EMU, such as patient demographics, past medical and surgical history, past and current anti-epileptic medications, physical and neurological examinations, classification of seizure episodes, and multi-modal electrophysiological recordings that include EEG, sleep, breathing, and EKG. MEDCIS provides an integrated solution for storing, processing, analyzing and visualizing these datasets as shown in Figure 2.1. MEDCIS is underpinned by EpSO, a formal domain ontology for the integration of structured, unstructured and signal data into a coherent structure for patient care and clinical research [82]. EpSO is used in various informatics tools in PRISM project, such as Ontology-driven Patient Information Capturing (OPIC) system for patient data entry [83], Epilepsy Data Extraction and Annotation (Epi-DEA) platform for epilepsy focusses clinical text processing from discharge sum-
Figure 2.1: PRISM Workflow
maries [14], ViSuAl AGgregator and Explorer (VISAGE) which is a query interface for patient cohort identification for SUDEP study [112], and cloud-based management system for large-scale electrophysiological signal waveforms (Cloudwave) [52].

The electrophysiological datasets used in this dissertation are collected at the UH-CMC EMU. These signal files are acquired in Nihon Kohden proprietary data format and converted to EDF. The EDF files are de-identified and curated using desktop applications, such as EDF Preprocessing Tools Suite developed at Case Western Reserve University (discussed in Chapter 8). 176 consented (out of 964 recruited) prospective SUDEP patients have been processed since January 2011 and large scale recordings are obtained from scalp and intracranial EEG from undergoing epilepsy surgical evaluations. A typical electrophysiological scalp recording is performed for a period of 5 days, and intracranial recording for 1 month. Due to technological limitations, the data is acquired as 6-hr segments, generating several data files per recording (20-30 data files for scalp, and 100-500 data files for intracranial). The size ranges from 5-10 GB for scalp recording, as compared to 100-500 GB for intracranial recording [53]. About 5.25 TB of data have been accumulated from patients recruited for the PRISM project from 2011-2013 (Figure 1.3). The rate of data collection in the EMU is increasing every year. For example, the volume of data at the end of 2012 was 2.7 TB, but 5.25 TB of data had already been collected by 2013 [81]. A smaller number of electrodes are used for scalp EEG recording (12-15 channels). Intracranial EEG (iEEG) recordings have more electrodes (100s of channels). Other channels represent multimodal physiological parameters such as EKG, heart rate, blood oxygen, respiration, blood pressure, sleep.

2.1.2 Use Cases for Electrophysiological Signal Analysis

The Cloudwave framework was developed using four design principles:
1. Fast access to individual signals stored in a multimodal data store;

2. Ability to partition signals into meaningful ‘segments’ based on seizure events;

3. Signal metadata-driven query for identification of matching patient cohorts; and

4. Visualization of multiple signals on a single screen for visual analysis by human reader.

A set of use cases were defined and systematically documented to identify appropriate features to be implemented in Cloudwave, such as the ability to:

1. Search for seizure event information in signal data, including the time of event occurrence or the time duration between start and end of an event. For example, occurrence of ‘Sign-of-Four’ lateralizing sign event, time duration between ‘onset of jittery phase and end of jittery phase’;

2. List patients with Cardiac Arrhythmia who also have irregular heartbeat rates. This can be further classified as Bradycardia or Tachycardia, with markings on EKG signal when heartbeat rate is below or above a threshold, such as 60 beats per minute (BPM);

3. Measure Heart Rate Variability (HRV) for selected patients, which is a physiological phenomenon of variation in the time interval between heartbeats;

4. List patients with respiratory arrhythmia with markings on the respiration signal where the rate of respiration is above or below a threshold (e.g. 5/minute); and

5. Provide EEG suppression information for patients, including time duration of ‘EEG Suppression’ in patients after a seizure occurrence and time duration of ‘EEG suppression to return to baseline’ event.
2.2 Technology used in Cloudwave

This section provides an overview of all the background resources that are adapted for specific needs of Cloudwave.

2.2.1 European Data Format (EDF)

EDF is a simple and flexible format for exchange and storage of multimodal biological and physical signals. EDF was published in 1992. Since then, EDF became the de-facto standard for EEG and PSG recordings in commercial equipment and multi-center research projects [32]. One EDF data file contains one uninterrupted digitized polygraphic recording. A data file consists of a header record followed by data records. The variable-length header record identifies the patient and specifies the technical characteristics of the recorded signals. The data records contain consecutive fixed-duration epochs of the polygraphic recording as shown in Figure 2.2.

Limitations of EDF for Visualization on Web

EDF partitions multimodal recordings as interleaved segments of all signals in temporal order. For large-scale recordings, this data format is very cumbersome due to its size and complexity. For example, a intracranial recording has more than 100 signals/channels. For a 6-hr recording, the size of the file is in the order of gigabytes (sometimes few terabytes). Transferring such large datasets to clinicians console, and extracting individual channels needs preprocessing before transferring the data to the web interface for visualization. Chapter 6 discusses CSF as a data format to enable real-time and interactive visualization on web, for efficient parsing of signal partitions using temporal order and event localization heuristics. This format using EpSO for data partitioning.
HEADER RECORD (we suggest to also adopt the 12 simple
8 ascii : version of this data format (0)
80 ascii : local patient identification (mind item 3 of the additi
80 ascii : local recording identification (mind item 4 of the ad
8 ascii : startdate of recording (dd.mm.yy) (mind item 2 of the
8 ascii : starttime of recording (hh.mm.ss)
8 ascii : number of bytes in header record
44 ascii : reserved
8 ascii : number of data records (-1 if unknown, obey item 10
8 ascii : duration of a data record, in seconds
4 ascii : number of signals (ns) in data record
ns * 16 ascii : ns * label (e.g. EEG Fpz-Cz or Body temp) (mi
ns * 80 ascii : ns * transducer type (e.g. AgAgCl electrode)
ns * 8 ascii : ns * physical dimension (e.g. uV or degreeC)
ns * 8 ascii : ns * physical minimum (e.g. -500 or 34)
ns * 8 ascii : ns * physical maximum (e.g. 500 or 40)
ns * 8 ascii : ns * digital minimum (e.g. -2048)
ns * 8 ascii : ns * digital maximum (e.g. 2047)
ns * 80 ascii : ns * prefiltering (e.g. HP:0.1Hz LP:75Hz)
ns * 8 ascii : ns * nr of samples in each data record
ns * 32 ascii : ns * reserved

DATA RECORD
nr of samples[1] * integer : first signal in the data record
nr of samples[2] * integer : second signal
..
..
nr of samples[ns] * integer : last signal

Figure 2.2: Schema of European Data Format
2.2.2 Epilepsy and Seizure Ontology (EpSO)

The use of a common terminological system for annotation and query of signal data is essential to reconcile data heterogeneity in multi-center clinical studies. As part of the PRISM project, EpSO was developed as a formal knowledge model representing epilepsy domain knowledge through close collaboration between epileptologists, including members of the International League Against Epilepsy (ILAE), and ontology engineers. EpSO has been modeled using the description logic-based Web Ontology Language (OWL2). EpSO uses the well-known ‘four-dimensional classification of epileptic seizures and epilepsies’ [65] [62] to model concepts describing seizures (abnormal electrical activity in brain), location of the seizures in brain, cause of seizures, and other medical conditions. EpSO currently models key concepts to effectively annotate signal data, including EEG patterns, both scalp and intracranial electrodes, their placement scheme (e.g. 10-10 system), and detailed brain anatomy to correlate signal events with their location [82]. Figure 2.3 illustrates a snapshot of the EpSO class hierarchy describing the various types of EEG patterns, such as abnormal, benign, and normal EEG patterns. EpSO currently has more than 1000 concepts and their related properties.

Role of Ontology in Cloudwave

In Cloudwave, EpSO classes are used to describe clinical seizure events identified in signal data, which ensures commonality of clinical event descriptors across different EMUs. These ontology annotations on signal data are used to support efficient data query and retrieval in Cloudwave. Also EpSO is used as conceptual model to define a new JSON-based representation model for signal data that supports optimal data partitioning for storage and efficient network transfer. This domain ontology-based signal representation model is described in Chapter 6.
2.2.3 JSON Data-interchange Format

The JavaScript Object Notation (JSON) is a light-weight, platform-independent, and extensible representation format that can be adapted to model valid data segments conforming to domain ontology. JSON was developed for efficient data transfer in client-server applications by using the Javascript programming language [26] and is defined by RFC 7159 as well as ECMA 404 standards. As compared to XML, JSON reduces the space requirements for datasets, but without constraining the expressivity or extensibility of the representation format. JSON uses nested objects of ‘attribute-value’ pairs with a set of standard syntactic elements that are easily parsed by JSON parsers, which are implemented in multiple languages [13]. Hence, Cloudwave uses JSON to define CSF that combines the flexibility of JSON with use of EpSO as reference schema.
JSON vs. XML

JSON uses Unicode and is self-documenting and does not need a schema unlike XML. JSON’s syntax is significantly simpler, so parsing is more efficient and has the ability to represent general computer science data structures: records, lists and trees. While there are transformations which allow XML to express these structures, JSON expresses them directly. JSON’s simple values are the same as used in programming languages [46]. JSON objects does not need any restructuring as their structures are similar to conventional programming language structures. JSON doesn’t have namespaces, as every object is a namespace. JSON uses context to avoid ambiguity, just as programming languages do. It is possible to add new tags and attributes, but it is not possible to extend XML to add expressive syntax for arrays and objects and numbers and booleans. JSON is flexible because fields can be added to existing structures without obsoleting existing programs.

2.2.4 Apache Hadoop Project

The Apache Hadoop project is an open-source software framework for large-scale processing and storage on reliable and scalable distributed file systems [15]. It is licensed under Apache License 2.0. This framework is composed of many modules, such as Hadoop Common, Hadoop Distributed File System (HDFS), Hadoop YARN which is the resource management platform, and Hadoop MapReduce that provides the programming model for large scale data processing. Hadoop software is designed to detect hardware failures, thereby delivering a highly available service. Cloudwave uses two main components of Apache Hadoop framework namely HDFS [87] that provides high-throughput access to application data and MapReduce [93] for parallel processing of large data sets.
Hadoop Distributed File System: Foundation and Features

HDFS is a distributed, highly fault-tolerant file system written in Java for the Hadoop framework. It is designed to run on low-cost commodity hardware and provides high-throughput access to application data and is suitable for applications with large data sets, like terabytes and petabytes. It is inspired by the Google File System.

A HDFS cluster consists of a single namenode and a cluster of datanodes. The namenode manages file system namespace and regulates client access to files. The datanodes serve blocks of data over the network as shown in Figure 2.4.

Hadoop MapReduce Programming Model

MapReduce is a framework for processing parallelizable problems across huge datasets using a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogenous hardware) [101].
‘Map’ step: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node.

\[ Map(k_1, v_1) \rightarrow list(k_2, v_2) \] (2.1)

‘Reduce’ step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output to answer the problem it was originally trying to solve.

\[ Reduce(k_2, \text{list}(v_2)) \rightarrow \text{list}(v_3) \] (2.2)

MapReduce allows for distributed processing of the map and reduction operations. Each map is an independent process and can be executed in parallel. All outputs of the map operation that share the same key are given to the reducer. All reduce tasks can be executed in parallel. The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation.

### 2.2.5 Agile Web Development using Ruby on Rails

Ruby on Rails, often simply Rails, is an open-source web application framework for designing and rapid prototyping applications based on the Ruby programming language using agile software development process. Ruby on Rails emphasizes the use of well-known software engineering patterns and principles, such as active record pattern, convention over configuration (CoC), don’t repeat yourself (DRY), and model-view-controller (MVC). The RoR framework uses *Convention over Configuration* for programming to limit the amount of code that needs to be written to
accomplish a certain task. Code needs to written only for functionality that is not using the conventions provided by the framework.

Ruby on Rails includes tools that make common development tasks easier “out of the box”, such as scaffolding that can automatically construct some of the models and views needed for a basic website [36]. Also included are WEBrick, ruby library for HTTP web services that integrated directly with Apache HTTP server, and Rake, a build system, distributed as a gem. Together with Ruby on Rails, these tools provide a basic development environment. Ruby on Rails is also noteworthy for its extensive use of the JavaScript libraries, such as jQuery and CoffeeScript for Ajax. Ruby uses RESTful web services. Ruby on Rails offers both HTML and JSON output formats. Ruby also provides a client library implementation for Hadoop WebHDFS. WebHDFS is the HTTP REST API that supports complete FileSystem interface for HDFS.

2.2.6 HighCharts: An Interactive JavaScript Charting Library

HighCharts is a charting library that allows creation of time-line charts in the JavaScript language, including sophisticated navigations options like a small navigator series, preset date ranges, date picker, scrolling and panning. Cloudwave uses HighCharts for signal rendering on the web. Some of the key features of HighCharts are compatibility with all modern browsers, free availability for non-commercial applications, open-source, support for JavaScript, and does not require any client-side plug-ins [45].

HighCharts offers numerous chart types for rendering, simple configuration syntax, support for dynamic update of charts with additional data points or series, range selector for dynamically configuring time ranges, and visual navigation to area of interest on the chart. HighCharts also features event marker flags, allows multiple axes, tooltip labels, configurable date time axis, data grouping, export
and print, zooming and panning, and ability to load external data from variety of sources.

2.2.7 PhysioToolkit: Physiologic signal processing and analysis APIs

PhysioNet offers free web access to large collections of recorded physiologic signals (PhysioBank) and related open-source software (PhysioToolkit). Each month, about 45,000 visitors worldwide use PhysioNet, retrieving about 4 terabytes of data [22]. PhysioToolkit is a large and growing library of software for physiologic signal processing and analysis, detection of physiologically significant events using both classical techniques and novel methods based on statistical physics and nonlinear dynamics, interactive display and characterization of signals, creation of new databases, simulation of physiologic and other signals, quantitative evaluation and comparison of analysis methods, and analysis of nonequilibrium and nonstationary processes. All PhysioToolkit software is available in source form under the GNU General Public License (GPL) [78].
Chapter 3

Cloudwave Architecture and Function

The core element of Cloudwave data management framework is the Web interface that enables researchers from different locations to store and share electrophysiological signal datasets. Cloudwave uses foundational resources, such as domain ontology, data representation for large-scale data, scalable and fault tolerant storage for providing real-time interactive signal query, analysis and visualization. This chapter presents the architecture of Cloudwave to address and its constituent modules that address the challenges of signal big data.

3.1 System Overview

Cloudwave framework is a client-server three-layered architecture as shown in Figure 3.1(A). The three-layers are:

1. Presentation. The presentation layer provides the application’s web interface (UI) and algorithms for parsing signal segments;

2. Application. The application layer implements the signal processing and analysis APIs. This layer is composed of components, such as JSON-based Cloudwave Signal Format, Cloudwave Classes for Data Management and
the domain ontology EpSO that facilitates signal partitioning, query and network transfer;

3. Data. The data layer provides secure access to distributed file systems on private/public clouds. This layer provides parallelization algorithm for processing EDF datasets to CSF.

Figure 3.1(B) shows the various software components that are used for development of the presentation, application and data layers. Cloudwave web interface is developed using Ruby on Rails agile web development framework and HighCharts. The WEBricks HTTP Ruby Gem provides APIs to communicate with Apache web server. The WebHDFS Client Ruby Gem provides filesystem utilities for HDFS. Apache Hadoop MapReduce, YARN, HDFS and WebHDFS modules are used by Cloudwave for processing, storage and access of CSF electrophysiology datasets.

The functional objectives for Cloudwave are:

1. enable real time interaction with massive scale signal data using web-based
signal visualization interface stored in scalable, high-performance distributed file systems;

2. an ontology-driven query interface that mitigates terminological heterogeneity in signal data annotation and improves quality of data retrieval;

3. efficient computation of clinically relevant seizure parameters, such as cardiac measurements, and oxygen desaturation and respiration levels;

4. for processing of large-scale electrophysiological datasets generated from multiple sources (EMUs) using parallel computing algorithms.

Figure 3.2 illustrates the high-level system overview of Cloudwave. Cloudwave is conceptually divided into two subsystems: Hadoop Electrophysiological Data Processing (HEDP) module, and Signal Visualization and Analysis module (SVA). HEDP handles signal processing and analysis using Hadoop MapReduce and Hadoop Distributed File Storage. SVA module houses the Web interface and built using Model View Controller (MVC) architecture with Ruby on Rails technology stack, HighCharts, AJAX and JSON. Cloudwave uses Physionet libraries for performing signal analysis, such as heart rate variability (HRV) on EKG signal.

### 3.2 Key Components and Workflow

Figure 3.3 illustrates the key components and workflow of Cloudwave framework. Electrophysiology data is generated from multiple EMUs and stored in proprietary data format in their local file systems. In Cloudwave, these data files are then converted to EDF, which is the standard for data-interchange for electrophysiology data (described in Chapter 2. The EDF files are then uploaded to HDFS using the Cloudwave FTP to HDFS MapReduce Copy APIs provided by HEDP module. Cloudwave uses EDFProcessor MapReduce parallel processing pipeline to process
Figure 3.2: Cloudwave System Overview
EDF files as they are uploaded, to be split into CSF files with metadata (experimental data, annotations, de-identified Patient Health Information (PHI) fields) and data segments for each signal. The web interface can query and visualize these CSF datasets accessed through WebHDFS REST API from HDFS.

3.3 Cloudwave Integration to MEDCIS

Cloudwave is integrated with MEDCIS to allow clinicians to query for patients using demography, medication, diagnosis and seizure events. Clinicians can review the signal data of patients in a cohort by navigating from MEDCIS VISual AGgregator Explorer (VISAGE) query interface to Cloudwave (Figure 3.4).
Figure 3.4: Cloudwave Integration to MEDCIS VISAGE Query Interface
Chapter 4

High Performance Data Processing Workflow using MapReduce

MapReduce is a popular programming framework introduced by Google to address computational and storage challenges for web-scale data [16]. Apache Hadoop provides an open-source implementation of the MapReduce framework, which can be used to store large volumes of data on HDFS and efficiently process the data by repeating the two steps of ‘Map’ and ‘Reduce’ on thousands of computing nodes [15]. Hadoop has built-in support for automated data distribution, recovery from component failures, balancing the computational load across different nodes, and parallel computation. In this chapter, we describe a novel approach to parallelize signal-processing workflows using MapReduce, to support real-time user interactions over massive-scale ontology annotated electrophysiological data in collaborative multi center research studies.

Typical EMU admissions for non-invasive electrophysiological study spans for a week generating multiple datasets of 6-hr recordings. This recording has 30-40 multimodal signals/channels, consisting of 20 EEG channels, 4 EKG channels, 1 channel for oxygen desaturation monitoring, 2 channels for respiratory signal,
and 1 channel for monitoring blood pressure. The data generated from the EMU is then converted to EDF for data sharing purposes. Each recording occupies 1-2 GB of disk space. However, as the number of channels, duration of the recording, sampling rate (the number of data points recorded per second) increases, the size of a dataset can range from few hundred GBs to 1-2 TB/dataset. Existing signal representation formats, such as EDF are not suitable for data-interchange for such large-scale datasets as described in Chapters 1 and 2. Hence there is a need for processing these massive datasets and storing them in a simpler, flexible data format that can be readily accessed from cloud-based web applications. This data format needs no support for efficient transfer of data segments to e browsers instead of complete signal which will significantly slow down the performance of the user interface. This chapter will describe the MapReduce classes for uploading EDF files to HDFS and processing EDF and Clinical Events to CSF.

### 4.1 Phase 1: Uploading EDFs to HDFS

EDF files generated by an EMU are uploaded to HDFS using the MapReduce HDFS Copy APIs provided as part of the HEDP module. Figure 4.1 describes the MapReduce algorithm for parallel copy of EDF files. The EDF Files are stored on distributed datanodes on the HDFS cluster.

### 4.2 Phase 2: Processing EDF to CSF using MapReduce

Cloudwave provides MapReduce classes that are specialized from Hadoop MapReduce library for processing large EDF files. In the following sections, the EDFProcessor MapReduce algorithm, workflow and classes are described.
4.2.1 EDFProcessor MapReduce Algorithm and Workflow

Figure 4.2 describes the MapReduce algorithm for processing EDF to CSF. The input to the MapReduce job is a directory of EDF Files that are generated during a single patient visit. Each EDF file is assigned to a map task for processing. The map task splits the EDF file and generates a CSF object for each signal with patient, study, signal metadata and data segments (will be described in Chapter 6). Number of CSF objects corresponds to the number of signals in the EDF. These CSF files are then written to a single directory by the Reduce task.

EDFProcessor MapReduce job allows for distributed processing of the map and reduction operations as shown in Figure 4.3. All EDF Files are processed in parallel and independently by each mapping operation. A set of reducers are used to collect the signals into a single directory corresponding to each EDF file.

The MapReduce workflow is illustrated for a specific example as shown in Figure 4.4. Patient visit 2012_12 has EDF files 2012_12_1.edf, 2012_12_2.edf, 2012_12_3.edf, 2012_12_4.edf and 2012_12_5.edf. These are processed to CSF as separate folders 2012_12_1/, 2012_12_2/, 2012_12_3/, 2012_12_4/ and 2012_12_5/. 2012_12_1 directory stores CSF files with metadata, annotations and data segments for each signal,
Figure 4.2: EDFProcessor MapReduce Algorithm

```plaintext
For each Patient p do
    Create MapReduce Job
    Read study s (EDF) and events e (.txt) from HDFS
    Create Map tasks (1 EDF + 1 event file) per map
    For each Map Task
        Generate PatientCSF object from EDF File Header of s
        Generate EventCSF object from events file e
        Generate StudyCSF object from EDF Signal Header of s
        For each signal sig in EDF Signal Header
            Read EDF data records for sig
            Partition Signal to Segments na based on segmentSize
            Convert Binary Signal to Numerical Analog Signal
            Generate SignalCSF Object for all na
        End
        Write (signal sig, CSFObjects)
    End
    For Reduce Task
        Create CSF_JSON File with all CSFObjects for each sig
        Write (sig, CSF_JSON)
    End
End
```

Figure 4.3: Cloudwave MapReduce Workflow
such as 2012_12_1_FP1.csf, 2012_12_1_SaO2.csf as shown in Figure 4.4.

4.2.2 EDFProcessor Classes for Processing EDF files

EDFProcessor is the Cloudwave package for processing EDF files to CSF for storage on distributed file systems. The functionality of these classes are enumerated below.

- **EDFFileInputFormat**: The EDFFileInputFormat class defines how the input EDF files are split and read by Hadoop. Given a directory of EDF files, the EDFFileInputFormat (extended from the abstract type FileInputFormat) read every EDF file as a key/value pair where the key is the filename and the value is the contents of the EDF file.

- **EDFRecordReader**: This class facilitates the actual loading of data from its
source (HDFS) using the input format and converts it into (key, value) pairs.

- **EDFMapper**: The Mapper performs the first phase of the MapReduce program. A map task is assigned for each EDF file. The Mapper processes the patient, study and signal metadata to CSF. EpSO provides the configuration parameter for duration of a signal segment in ‘seconds’. Based on this value, EDFMapper generates signal segments in numeric format and generates results as (filename, EDFWritable) pairs.

- **EDFWritable**: The EDFWritable object has two data fields namely signal identifier/label and signal segments in numerical value.

- **EDFReducer**: Each reducer instance receives a key that corresponds to the filename and an iterator over all the corresponding EDFWritable values to be written as individual signal files.

- **EDFFileOutputFormat**: The class specifies the output format of the Reducer phase, to generate CSF files for every (key, value) with the key as the signal filename and the value as signal segments along with metadata and annotations.

- **EDFRecordWriter**: The EDFRecordWriter creates a directory for each patient-study and writes the processed signal values into output files. For example, if an EDF file has 36 signals, then EDFRecordWriter will generate a directory with 36 files corresponding to 36 signals. Each file stores the data records of that specific signal.

Figure 4.5 illustrates the use of these new EDF-specific classes in the HEDP module to process the EDF. These classes will be released to the wider research community both as part of the PRISM project and to the Hadoop user community members for electrophysiological signal processing.
Figure 4.5: EDF Classes using Hadoop API
4.3 Results

Cloudwave was used to process electrophysiological data from 50 patients admitted to the UH-CMC EMU. The patient characteristics were tabulated that can be used by researchers in the PRISM project to query for patient cohorts (Table 4.1). Females outnumbered Males (61% vs. 39%) with a median age of 53 years within a range of 17-75 years. Majority of the patients suffered from epileptic seizures (91%) and only a small number of patients had non-epileptic seizures (9%). Almost all the patient recordings were made using surface electrodes on the scalp (98%) and only 2% of the recordings were generated from intracranial electrodes. Intracranial electrodes generate significantly larger volume of signal data as compared to scalp electrodes (200 vs. 30-40) and the use of Hadoop platform enables Cloudwave to efficiently manage intracranial data.

A comparative evaluation was performed to effectively measure the advantages of using Hadoop distributed computing platform for processing signal data in Cloudwave. The evaluation used five patient recordings collected from consented patients enrolled in the PRISM SUDEP study at UH-CMC EMU. The signal files were de-identified and manually verified to ensure removal of Protected Health Information (PHI) and stored in EDF data representation format. Table 4.2 shows the details of the dataset used in the evaluation with a total of 77GB of signal data. The evaluation involved processing of all five datasets on a standalone signal processing application running on a server machine with Quad-Core Intel Xeon 2.3 GHz processor, 3GB main memory, a 256KB L2 cache, and 8MB L3 cache. Cloudwave was installed on a single node cluster configuration with Intel Core i7 2.93 GHz processor, 16GB main memory, and 8MB cache. Cloudwave was also installed on a 30-node cluster configuration with all nodes having similar processor configuration.
Table 4.1: Characteristics of patient enrolled in PRISM SUDEP study

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Patients (% of 50)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>39</td>
</tr>
<tr>
<td>Female</td>
<td>61</td>
</tr>
<tr>
<td>Age (Range: 17-75, Median: 53)</td>
<td></td>
</tr>
<tr>
<td>0-20</td>
<td>8</td>
</tr>
<tr>
<td>21-40</td>
<td>52</td>
</tr>
<tr>
<td>41-60</td>
<td>34</td>
</tr>
<tr>
<td>61-80</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primary Diagnosis</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Seizure Semiology</td>
<td></td>
</tr>
<tr>
<td>Epileptic Seizure</td>
<td>91</td>
</tr>
<tr>
<td>Non-Epileptic Seizure</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Seizure Feature</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lateralizing Sign</td>
<td>64</td>
</tr>
<tr>
<td>No Lateralizing Sign</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Epileptogenic Zone</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal</td>
<td>83</td>
</tr>
<tr>
<td>Generalized</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Electrodes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalp</td>
<td>98</td>
</tr>
<tr>
<td>Intracranial</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Etiology</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Genetic Defect</td>
<td>9</td>
</tr>
<tr>
<td>Structural/Metabolic</td>
<td>13</td>
</tr>
<tr>
<td>Unknown</td>
<td>78</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Medication</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Anti-epileptic</td>
<td>89</td>
</tr>
<tr>
<td>Anti-depressant</td>
<td>7</td>
</tr>
<tr>
<td>Neuroleptic</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 4.2: Details of electrophysiological dataset used in comparative evaluation of Cloudwave

<table>
<thead>
<tr>
<th>DE identified Patient ID</th>
<th>Total Size in GB</th>
<th>Number of Studies</th>
<th>Number of Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012_6</td>
<td>14.93</td>
<td>25</td>
<td>51-74</td>
</tr>
<tr>
<td>2012_7</td>
<td>15.01</td>
<td>26</td>
<td>51-73</td>
</tr>
<tr>
<td>2012_8</td>
<td>13.67</td>
<td>33</td>
<td>55-63</td>
</tr>
<tr>
<td>2012_13</td>
<td>18.1</td>
<td>50</td>
<td>42-74</td>
</tr>
<tr>
<td>2012_25</td>
<td>15.46</td>
<td>37</td>
<td>72</td>
</tr>
</tbody>
</table>
4.3.1 Processing EDF to Binary Signal Files

In this evaluation, the EDF files were processed to individual signal files in binary format instead of numerical format as in CSF. Two performance tests were performed. The first test compares the time taken (in minutes) to process data on the standalone system and Cloudwave as the number of signals increase for 25 studies per patient. Figure 4.6 (a) shows that it takes 22-36 minutes for 10 signals and 1.5-3 hours (91-177 min) for 40 signals to be processed on the standalone system. In contrast, Figure 4.6 (c) shows that it takes only 4-6 min for 10 signals and 7-11 min for 40 signals on Cloudwave. All patient studies are six-hour recordings.

The second test was done to compare the execution time (in minutes) as the number of studies per patient is increased with each study consisting of 40 signals. Figure 4.6 (b) and (d) shows the results of this test. Figure 4.6 (b) shows that it takes 21-39 minutes for 5 studies and 1.5-3 hours (91-177 min) for 25 studies on the
Figure 4.7: The average time taken by Cloudwave and the standalone application to process various categories of signal data.

standalone system, but Cloudwave takes only 1 min for 5 studies and 7-11 min for 25 studies (Figure 4.6(d)).

Figure 4.7 (a) and (b) compares the average execution time on standalone system and Cloudwave for processing data from all 5 patients as the number of signals and studies are increased. The results clearly demonstrate that there is an order of magnitude difference between the average times taken by Cloudwave as compared to the standalone system for increasing number of studies having 40 signals/study.
4.3.2 Processing EDF to CSF

The performance of processing EDF to CSF was conducted for the same 5 patients with 5 studies, each study having 40 signals. Figure 4.8 shows the comparison of execution time in minutes to process data on a standalone system and Cloudwave. Figure 4.8(a) shows that it takes 31-49 minutes on the standalone system and 9-17 minutes on 30-node Hadoop cluster. Figure 4.8(b) shows the comparison of execution times in minutes for 2 patients, 2012_6 and 2012_8 on standalone system, 5-node, 10-node, 15-node, 20-node, 25-node and 30-node cluster. Execution time is 4 times faster on 30-node cluster (8 min) when compared to standalone system (32 min).

The results of the comparative performance evaluation of Cloudwave clearly demonstrate the significant advantages of using the Hadoop distributed computing platform for processing very large electrophysiological signal datasets.
Chapter 5

Cloud-based Web Interface for Signal Visualization

Scalable, intuitive, and real-time visualization are the primary objectives for developing a web application that communicates with high-performance distributed file systems, together with a new adaptive data interchange format designed for cloud based applications. This chapter presents the design and implementation of a Cloud based web application for multimodal signal visualization using a domain ontology for producing annotated waveforms. The chapter enumerates the key components of the web application, the salient features of the web server, and various protocols and technologies used for development. Performance results of a cloud application as compared to a desktop tool are also discussed.

Cloudwave uses Ruby on Rails agile web development framework for rapid prototyping and development. RoR framework provides the necessary tools required to rapidly launch a production grade web server for using application servers, such as WEBbrick, that integrates directly into Apache web server (discussed in Chapter 2. Cloudwave Signal Visualization and Analysis Module consists of the user interface components providing different visualization features
and the server-side components that handles the parsing of datasets, computations for UI components, such as filtering, communication protocol for transfer of files from the cloud, and provide secure access to underlying data stores for the users.

5.1 Cloudwave Web Interface Design Parameters

A single multimodal electrophysiological recording, conducted over 24-hr period with 100 or more channels results in 5-10 GB of data. To support visualization of many recordings of this size, the Cloudwave web interface has been designed to support the needs of researchers collaborating across multiple institutions with the following main objectives:

1. Easy Access. A secure and efficient access to the web interface is critical for consideration for collaboration. Researchers can use a browser as user interface to login to Cloudwave;

2. Flexible Data Model. A flexible model is required to represent application objects, such as patient details, recording information, event annotations and their relationships without having to migrate the database schema. These objects constitute the metadata of the signal raw data and is required to navigate to sections of the actual data. The flexibility in representing and parsing these objects are critical to the performance and intuitiveness of the application;

3. Easy Scalability. Cloudwave web interface supports accessing data from within a cluster or between clusters at different data centers;

4. Consistent High Performance. Cloudwave web interface is designed for massively concurrent data user and consistent high throughput; and
Clinical events mapped to EpSO classes

5. High Rate of Availability. Cloudwave web application should be ‘always running’ with very minimal downtime.

Cloudwave is integrated with MEDCIS VISAGE Patient Cohort query interface (described in Chapter 3). VISAGE uses EpSO for selecting query conditions such as, epileptogenic zone, etiology, demographics, semiology, and medications. Matching patients are listed as query results. Patient’s multimodal electrophysiological data can be visualized as a web link to the Cloudwave web interface as shown in Figure 5.1. Cloudwave interface has a top display area, for showing de-identified patient details, recording information such as data and time of recording, list of all the recordings for the patient during multiple clinical visits, and the visualization controls. The bottom display area shows a multimodal visualization of selected signals.

Figure 5.1: The Cloudwave User Interface showing the signal visualization, signal filtering, and montage features.
5.2 Web Server Communication with Hadoop HDFS

Web pages are served by Apache web server over HTTP. WebHDFS provides HTTP REST API that supports file system interface for HDFS. WebHDFS APIs consists of authentication, proxy users, file and directory operations on HDFS. Cloudwave provides a connection object called WebHDFSConnector that uses the WebHDFS Ruby gem for accessing HDFS from within the RoR framework. More details on the WebHDFS APIs, Apache web server and Ruby on Rails Framework are discussed in Chapter 2.

5.3 Main Features of the Web Interface

The key components of the web interface are:

1. Application of Montages to signals and support for custom montages;
2. Multimodal time-synchronized visualization of signal waveforms with pagination support;
3. Time and Frequency filtering on selected signals;
4. Annotated display of signal waveforms showing clinical seizure related events;
5. Standardized terminology for clinical events for data shared from multiple sources; and
6. Ability to list all electrophysiological recording from multiple visits for a patient.

These features are discussed in detail in the following sections.
5.3.1 Application of Montages to Signals

A montage is a composition of signal channels that are arranged in a logical series to provide an accurate localization of abnormal electrical activity to specific parts of the brain [69]. Visualization of signal data as montage helps neurologists to identify the brain regions that are responsible for start of seizures in patients and also help in diagnosis of specific category of epilepsy. The three major categories of montages are bipolar montage, referential montage, and Laplacian montage that are used in analysis of signal data in epilepsy [69]. Figure 5.2(A) shows the wiring of electrodes for a referential montage and Figure 5.2(B) for a bipolar montage.

There are six standard montages that are recommended for use in epilepsy center, which are supported as default montages in the Cloudwave user interface. These montages consist of multiple channels, which make it difficult to support interaction with signal data corresponding to these montages. These standard montages and the corresponding brain regions covered by them are described below:

1. **M1** montage with 19 channels covers the frontopolar, temporal, occipital, parietal, frontal, and central brain regions;

2. **M2** montage with 21 channels consists of signals from the frontopolar, auti-
ular, temporal, parietal, frontal, and occipital regions;

3. **M3** montage with 19 channels covers the frontopolar, temporal, parietal, frontal, and occipital brain regions;

4. **M4** montage with 21 channels consists of signals from frontotemporal, frontopolar, temporal, parietal, occipital, frontal, and auricular brain regions;

5. **M5** montage with 22 channels consists of signals from frontal, auricular, temporal, parietal, occipital, and frontopolar regions; and

6. **M6** montage consists of 22 channels covering the frontopolar, frontal, parietal, temporal, occipital, frontotemporal, and central brain regions.

First, a study is selected from the list of studies for the selected patient. This is shown in Figure 5.3(a). Second, a montage is selected for display as shown in Figure 5.3(b). The channels defined by that montage are populated within a ‘select menu’. Third, the user may select all or some of the signals/channels for display, as shown in Figure 5.3(c). After selection of appropriate constraints, the selected signals are displayed in the ‘charting area’. Users may also choose to create their own montages using the ‘add montage’ interface, as shown in Figure 5.3(d). The user has the ability to create any combination of signals for custom montages.
5.3.2 Multimodal Time-synchronized Signals displaying Clinical Seizure Events

The ‘charting area’ using an open source charting library called Highcharts [45], which allows creation of general timeline charts in the JavaScript language, including sophisticated navigations options like a small navigator series, preset date ranges, date picker, scrolling and panning (discussed in Chapter 2). A mutimodal chart for the selected signals is generated that are synchronized temporally in a single screen (Figure 5.4). In addition, users can choose to view events that are marked on the signals to better navigate through the data. The ontology terms from EpSO are used to reconcile differences in the terminology used for describing seizure events across the different participating centers. The ontology-driven approach in Cloudwave enables rendering of the correct signal data segment with standardized event markings in response to user query. For example, EMU datasets can be labeled with either ‘Intermittent Seizure’ or ‘Intermittent Slow Activity’ to represent the same event, which is reconciled to a standardized ‘IntermittentSlowActivity’ term modeled in EpSO. EpSO enables Cloudwave to map variations of a term used across different EMUs to a standard reference term to facilitate interoperability of signal datasets.

5.3.3 Application of Filters

Signal analysis is performed by the application of ‘sensitivity’ and ‘frequency’ filters as shown in part (a) of Figure 5.5. The sensitivity filter is calculated by multiplying the data by a value chosen from the drop down menu. The time constant filter is a low-cut filter that smooth out parts of the signals below a selected threshold frequency and the high frequency filter is a high-cut filter that smooth out arts of the signal above the given threshold frequency. Part (b) of Figure 5.5 illustrates
the selection of 5 microvolts sensitivity filter, part (c) and (d) of Figure 5.5 shows the signal before and after application of the filter.

5.3.4 Optimized Visualization

Client-side optimizations have increased the performance of signal visualization. For example, the signals are loaded asynchronously when possible and sent to the ‘client’ interface in a format that allows ‘caching’ to avoid repeated server access, which often slows down signal visualization. For signal rendering, best practices for JavaScript were followed to prevent memory leaks and eliminate unnecessary memory usage. The signals are rendered as they are selected, instead of waiting
for the user to make all selection and submit the request, which results in notable reduction in the ‘wait time’ associated with display of multi-graph chart.

5.4 Results

Cloudwave has been integrated with the MEDCIS for signal analysis and rendering of patient cohorts at the UH-CMC EMU. At present, Cloudwave database is populated with 1TB of signal data from 85 patients with seizures. A patient examination in the UH-CMC is typically done for five consecutive days after the patient is admitted and signal data are recorded in 6-hour segments, thereby generating 20 studies. Each study generates 1-2 GB of data per study. The evaluation dataset also consists of two patients with surgically implanted intra-cranial electrodes. Signal data from these patients with intra-cranial electrodes were recorded a period of one month that also generated about 1 TB of data.

We evaluated the performance of Cloudwave in terms of time taken to:

1. Render an increasing number of discrete signals per page;

2. Render signal data with increasing size corresponding to different recording time period;

3. Time to load each signal (in milliseconds); and

4. Time to render each signal by the HighCharts charting module (in milliseconds).

The evaluation tests were performed on a standard Web browser (Google Chrome) with both browser cache disabled (to simulate access for each query without caching) and enabled (to evaluate the impact of browser caching). The load time was measured as an average of 3 loads. The results of our evaluation are presented in the
Figure 5.6: Total time to render signals with increasing number of signals/page for increasing length of EEG recordings.

graphs below. Figure 5.6 shows that the total time taken to render an increasing number of signals with increasing length of recordings. The total number of signals per page was increased by a factor of 2, starting with 5 channels to a maximum of 40 channels. Similarly, the length of signal recording is doubled starting with 15 minutes to a maximum of 120 minutes. The total time to render the signal recordings is calculated as the sum of average query times for selected signals over 3 runs and the average rendering time on the web interface. It was observed that the total time for rendering 30 or fewer signals per page for 1-hour recording takes less than a minute for query and rendering on the web.

Figure 5.7 shows query and rendering time for a 60-minute EEG signal recording for increasing number of signals in increments of five, starting with 5 signals to a total of 50 channels in a single page. The time taken to retrieve all the selected signals increases linearly from 16 seconds (for 5 signals) to 2.5 minutes (for 40 signals). The rendering time is less than 15 seconds for 0-40 signals. It is important
to note that the rendering time significantly increases when the number of signals per page is greater than 40. Overall, the results of the Cloudwave performance evaluation demonstrate that Cloudwave is a practical tool and hence it is being deployed at the UH CMC EMU for use by the EMU staff [53].
Chapter 6

Ontology as a Conceptual Model for Data Partitioning and Querying

EpSO is a domain ontology developed as part of the PRISM project to address multiple data management challenges in epilepsy research, including integration of heterogeneous data, querying, and data validation [82]. EpSO is already used in a patient information capture system, which is deployed at the UH-CMC EMU to ensure consistency in collection of patient information [83]. EpSO has also been used for development of a clinical text processing system called EpiDEA to extract structured information from patient discharge summaries [14]. In addition to these traditional roles of a domain ontology, EpSO is also used as a conceptual model for big data management in Cloudwave. EpSO uses the well-known ‘four-dimensional classification of epileptic seizures and epilepsies’ [65] [62] to model concepts describing seizures (abnormal electrical activity in brain), location of the seizures in brain, cause of seizures, and other medical conditions. EpSO reuses concepts from many existing biomedical ontologies, such as the Foundational Model of Anatomy (FMA) [80] and RxNorm for medical drug classification [73].

EpSO uses OWL2 constructs, including existential quantifiers, to define classes,
properties, and restrictions on class attributes. Epilepsy syndromes are complex concepts with specific values assigned to their attributes and EpSO models this information by defining appropriate restrictions on multiple OWL2 object properties. For example, Carbamazepine is asserted as the preferred medication for a specific category of epilepsy called ADNFLE using restriction on the hasPreferredMedication object property (Figure 6.1 illustrates the restrictions on ADNFLE). Similar ontology constructs in EpSO allows software applications to automatically distinguish and classify different types of epilepsy syndromes, for example mesial frontal epilepsy is a sub category of frontal lobe epilepsy. EpSO models the six standard montages (described earlier in Chapter 5) together with electrodes that constitute the montages. In addition, the specific brain region associated with each electrode, which generates the signal data corresponding to a channel, is defined using OWL2 class restrictions on object property hasLocation.
In Cloudwave, EpSO classes are used to describe clinical events identified in signal data, which ensures commonality of clinical event descriptors across different EMUs. Signal data is manually analyzed to identify clinical events, such as abnormal electrical activity in EEG, and annotation tools are used to associate a text note with the signal data, which are automatically mapped to EpSO classes in Cloudwave. These ontology annotations on signal data are used to support efficient data query and retrieval in Cloudwave. User queries in the Cloudwave user interface are composed of EpSO classes, which allow Cloudwave to use EpSO as a query ontology (illustrated in Figure 6.1). Similar to traditional approaches to ontology-based query systems, Cloudwave uses EpSO to ‘expand’ the user query with synonym and subclass information (using standard OWL2 semantics for reasoning [70]). This expanded query expression allows Cloudwave to support signal retrieval beyond ‘keyword matching’ in signal data. In addition to its role as a query ontology in Cloudwave, the EpSO schema has been used to define a new JSON-based representation model for signal data that supports optimal data partitioning for storage and efficient network transfer. This domain ontology-based signal representation model is described in the next section.

### 6.1 Cloudwave Signal Format: An Extensible JSON Model for Electrophysiological Signal Data

The two essential requirements for a representation model for signal data in Cloudwave are efficient data partitioning and network transfer to remote Web clients implementing the user interface. JSON is a light-weight, platform-independent, and extensible representation format that can be adapted to model valid data segments conforming to domain ontology. JSON was developed for efficient data transfer in client-server applications by using the Javascript programming language [13] and
is defined by RFC 7159 as well as ECMA 404 standards. As compared to XML, JSON reduces the space requirements for datasets, but without constraining the expressivity or extensibility of the representation format. JSON uses nested objects of ‘attribute-value’ pairs with a set of standard syntactic elements that are easily parsed by JSON parsers, which are implemented in multiple languages [13] as described in Chapter 2. Hence, Cloudwave uses JSON that combines the flexibility with use of EpSO as reference schema.

The CSF is divided into two parts: (1) study metadata and annotations, and (2) data segments (Figure 6.2 illustrates the segments of the CSF). The study metadata segment of CSF is further sub-divided into two additional sections describing: (a) details of the experiment study, including the patient identifier, clinical events, recording time-stamps, the total duration of recording, and (b) the instrument details of the recording, including the unit of measurement (e.g. microvolts) and transducer type. The second part of CSF consists of the signal data values, which have been divided into segments of specific duration. The signal data are stored as an array of text values after conversion from the original EDF file binary format in Cloudwave (discussed in Chapter 2). The storage of data in CSF compliant text format reduces the computational workload on the Cloudwave user interface module to convert binary data into numeric format and improves the response time for signal visualization. Though, the storage of signal data in CSF file results in a moderate increase in total size of data as compared to the original EDF file binary format, there is a significant gain in user interface response time.

A key characteristic of CSF is the use of EpSO classes for describing clinical events across all channels in a given signal montage. CSF defines a composite ‘attribute-value’ element consisting of the timestamp associated with the clinical event, the name of the clinical event, and the associated EpSO classes (Figure 6.3 shows a segment of the CSF with ontology annotation and the corresponding
Cloudwave Signal Format (CSF)

Figure 6.2: The Cloudwave Signal Format consists of two segments storing the signal metadata and data values.
Epilepsy and Seizure Ontology

Cloudwave Signal Format (CSF)

Figure 6.3: The CSF ontology annotation segment illustrating the mapping to EpSO classes.

EpSO class). These CSF ontology annotations are used at multiple stages of signal management Cloudwave, including signal visualization, query execution, and data partitioning. In the next section, we describe the role of EpSO in Cloudwave data partitioning.

6.2 Cloudwave Data Partitioning

The default storage schema for signal data in EDF files partitions signal values into interleaved segments of all channels corresponding to a specific time duration of a recording [32] (described in Chapter 4). We compared two partitioning approaches for signal data in Cloudwave. The first approach extracted and aggregated data corresponding to a single signal channel, such as ECG. Although the Cloudwave storage module could use the single channel dataset, they could not be efficiently transferred over the network to the visualization module, as discussed in the following section showing comparative evaluation of the two formats. Hence, we implemented a second approach, which partitions the data of each signal channel into ‘epochs’ of 30 seconds duration.

The 30 seconds duration corresponds to the default ‘window size’ of many existing signal processing tools, such as the Nihon Kohden Neural Workbench [48], and is also implemented in the Cloudwave user interface. Hence, partitioning sig-
nal data into 30 seconds epochs, which are stored as CSF files, allows the Cloudwave user interface to visualize the signal data transferred from the server without additional processing. This approach significantly improves the response time of the user interface for visualization of complex signal montages consisting of multiple channels. We demonstrate the effectiveness of the Cloudwave data partitioning approach and CSF in the results section.

6.3 Comparing Signal Data Models

Table 6.1 enumerates the features of the EDF, CSF and MEF data formats. EDF is a conventional data format that bundles all the signals into 1 single file, as compared to MEF and CSF that stores signals in separate files. MEF and CSF are both hierarchical data formats, MEF uses XML standard, whereas CSF uses JSON, which is considered to have better parsing and representation for data elements. CSF uses EpSO as conceptual model to address clinical event heterogeneity unlike MEF or EDF. Also EpSO is used by CSF to partition data into configurable data segments based on duration of a segment required by the web application. MEF provides data compression, data encryption and cyclic redundancy checks for faster and secure network transfer.

6.4 Results

As part of the PRISM project, we have stored about 1 TB of electrophysiological signal data in Cloudwave storage module (implemented on HDFS), which is only a sub-set of the 11 TB of signal data collected at the University Hospitals of Cleveland epilepsy center. The remaining data will be processed and stored in Cloudwave over the next few months. To demonstrate the advantages of using EpSO
Table 6.1: Comparing Signal Data Models (CSF, EDF and MEF)

<table>
<thead>
<tr>
<th></th>
<th>EDF</th>
<th>CSF</th>
<th>MEF (IEEG-Portal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bundles all signals into 1 File</td>
<td>1 File for each signal (metadata + annotations + raw data)</td>
<td>2 Files for each signal (annotations are stored separately)</td>
<td></td>
</tr>
<tr>
<td>Does NOT address heterogeneity</td>
<td>Address clinical event heterogeneity (EpSO as conceptual model)</td>
<td>Does NOT address heterogeneity</td>
<td></td>
</tr>
<tr>
<td>Does not support data partitioning</td>
<td>Ability to partition into configurable segment sizes (e.g. 30s epoch)</td>
<td>Uses block index format to calculate segment to be retrieved</td>
<td></td>
</tr>
<tr>
<td>Not supported</td>
<td>Future Work</td>
<td>Provides data compression and 32-bit cyclic redundancy checks</td>
<td></td>
</tr>
<tr>
<td>Not supported</td>
<td>Future Work</td>
<td>Provides data encryption</td>
<td></td>
</tr>
</tbody>
</table>

as a domain ontology in Cloudwave, we conducted a comparative evaluation of using CSF and epoch-based data partitioning techniques with existing EDF files. The comparative evaluation was performed on: (1) a desktop computer with Intel Core i7 2.93 GHz processor, 16 GB main memory, and 8 MB cache, and (2) a single node cluster implementation of Hadoop on the same desktop computer configuration.

The objectives of the evaluations are to demonstrate:

1. The time taken by the Cloudwave storage module to retrieve and transfer data to the user interface module is low as compared to existing approach using a desktop file system;

2. The access time for multiple signal channels in montages is significantly lower for Cloudwave epochs as compared to channel-based partitioning scheme; and
3. The CSF format improves the responsiveness of the Cloudwave user interface module for signal montages as compared to existing EDF file binary data format.

6.4.1 Cloudwave Storage Module Compared to Desktop File System

Figure 6.4(a) shows that the time to retrieve and render signal data corresponding to a single ECG channel for increasing duration of signal recording time and Figure 6.4(b) shows the access time for 20 signal channels. The results clearly demonstrate that the CSF file served from Cloudwave storage module requires less time required to retrieve and render data in the user interface. The efficiency of the Cloudwave storage module is significantly higher as the size of the data (in terms of recording time) increases from 30 seconds to total of 6 hours as compared to desktop file systems.
6.4.2 Effectiveness of Cloudwave Data Partitioning Approach

Figure 6.5 shows that the Cloudwave approach to partition signal data into 30 seconds epochs leads to faster data retrieval and visualization as compared to simple channel-based data partitioning approach. The results demonstrate that the epoch-based partitioning approach is consistently faster as compared to channel-based partitioning for multiple signal channels corresponding to the six standard montages that are available as default choices for users in the Cloudwave user interface.

6.4.3 Performance of CSF for Network Transfer of Signal Data for Standard Montages

The final set of evaluation results in Figure 6.6 validate the advantages of the domain ontology-based CSF file for both signal data retrieval and signal rendering.
Figure 6.6: The access time for CSF files is clearly less than the binary format EDF files for 30 seconds epoch data segments across all the six standard montages.

as compared to existing binary format in EDF files for 30 seconds epoch segments. This is an important result that clearly demonstrates the role of EpSO domain ontology as a conceptual model for big data applications such as Cloudwave.
Chapter 7

Computational Processing of ECG Signal using MapReduce

ECG signal data is a vital source of clinical signs for seizure detection and for correlating effects of anti-epileptic drugs on the autonomic nervous system [94]. It is increasingly being studied for use in automated seizure-detection instruments [113] [56]. Heart rate changes in epilepsy patients are measured before seizure (pre-ictal), during seizure (ictal), between seizures (inter-ictal), and after seizure (post-ictal) to identify a variety of conditions, such as cardiac arrhythmia and conduction abnormalities [113].

Identifying the QRS complex in ECG recordings, which is associated with depolarization of the right and left heart ventricles, can be used to analyze increases in heart rate (also called tachycardia) or decreases in heart rate (also called bradycardia). About 75-80% of epilepsy patients with temporal lobe epilepsy have tachycardia, while bradycardia occurs more rarely in about 3% of the patients with frontal lobe epilepsy [113] [68]. The intervals between two consecutive heart beats, referred to as the RR interval time series, is used to derive HRV for analyzing changes in the two branches of the autonomic nervous system during seizures [6]. The in-
The interval between the Q and T features in ECG (QT interval) has been found to be increased during epileptiform EEG discharges and has been studied in the context of a poorly understood phenomenon called SUDEP [89]. Signal analysis algorithms for SUDEP study include:

1. Cardiac Arrhythmia. Heart Rate Variability (Variation of time interval between heartbeats (beat-to-beat interval));
2. Respiratory Arrhythmia. Respiration rate is above/below a threshold;
3. Postictal Generalized EEG Suppression. Time of ‘electro-cerebral shutdown’ where bipolar scalp EEG amplitude is less than 10 microvolts;
4. Oxygen Desaturation Levels below threshold; and
5. Quantification and characterization of seizure occurrences;

The Cloudwave framework provides parallel computing approach for computing cardiac measures, such as QRS detection, RR intervals and instantaneous heart rate for heart rate variability analysis on EKG signal.

7.1 Computation Algorithms for Cardiac Electrophysiology

The ECG data of EMU patients are usually recorded with multiple electrodes to provide both reference and redundancy. The ECG data together with electrophysiological data from other channels (e.g., EEG, blood oxygen measurements) are converted into the European Data Format (EDF+), which is a widely used standard for storage and exchange of electrophysiological data [32]. In the PRISM project, the four ECG channels are extracted from an EDF file and processed to compute the heart rate measurements. The two common cardiac measures used in epilepsy
clinical research are (1) RR intervals and (2) instantaneous heart rate (IHR) to detect tachycardia or bradycardia. These two measures are derived from the time interval between two consecutive heartbeats requiring the accurate detection of the R-wave in one QRS complex and the accurate detection of the R-wave in the next QRS complex [21]. Cloudwave uses the \textit{wqrs} open-source single-channel QRS detector algorithm and IHR algorithm developed by the PhysioNet project [21]. However, these algorithms were developed for sequential execution, and several critical challenges need to be addressed to integrate them into a parallel computational workflow (designing parallelization approaches for sequential algorithms has been an active area of computer science research for the past five decades [1]).

Cloudwave addresses these challenges in two phases: (1) formal algorithm analysis of the cardiac-measurement workflow to characterize its degree of parallelization based on the famous Amdahl’s law [1]; (2) defining a new parallel algorithm for the MapReduce model that can be implemented in the open-source Hadoop environment. Cloudwave uses a ‘coarse-grained parallelization’ approach for formal dependency analysis of the computational workflow [4] (Figure 7.1 illustrates the complete workflow for computing cardiac measures), where each node is considered as an atomic task (e.g., QRS detection). Cloudwave uses the three conditions of ‘flow dependency’, ‘anti-dependency’, and ‘output dependency’ [4] to identify computational tasks that can be executed in parallel. The dependency analysis reveals that the maximum length of the critical path [95] is three, spanning the tasks ‘EDF2MIT’, ‘WQRS’, and ‘IHR’, which allows ‘RDSAMP’ and ‘ANN2RR’ tasks to be executed in parallel. This dependency analysis is used to define an efficient parallel approach that conforms to the iterative two-step MapReduce framework.

In the second phase, Cloudwave introduces a new algorithm for signal processing in the MapReduce programming model (Figure 7.2 (A)) that defines (1)
Figure 7.1: Cardiac measurement workflow implemented in Cloudwave to identify QRS complexes and compute RR intervals and instantaneous heart rate (IHR) values from ECG signal data. BPM, beats/min; EDF, European Data Format; HRV, heart rate variability.
the suitable construct for signal data partitions to achieve effective parallelization (e.g., 10 min segments), (2) the set of computations that can be implemented during the Map phases, and (3) the aggregation steps that correspond to the Reduce phases. The MapReduce model operates on discrete entities of (key, value) pairs (other parallelization models such as Message Passing Interface require different data structures). The Cloudwave algorithm defines the sample identifier associated with each discrete signal measure as a ‘key’ and the signal measure as a ‘value’ for the Map phase. In the Reduce phase, the segment identifier is used as ‘key’ and the three sets of R-waves, the RR intervals, and IHR measures as ‘value’ (Figure 7.2 (B)). This algorithm enables the Cloudwave platform to efficiently compute cardiac measures, generate optimal-sized signal segments for visualization, and support real-time interactions for users with ontology-driven querying.

### 7.2 MapReduce for Cardiac Signal Analytics

There is no existing support to store, access, and process ECG data in Hadoop; hence we have developed a library of specialized classes for both computations in MapReduce framework and managing electrophysiological data in HDFS. These new classes constitute an open-source Hadoop signal-processing middleware layer that can be used by other software tools for processing large-scale electrophysiological data. The Cloudwave data-storage module uses HDFS, which is a high-performance distributed file system, to address the need to store and manage TBs of signal data over multiple machines in a cluster environment [87]. HDFS has built-in support to store data reliably even if some of the machines in the cluster fail and also effectively balance the distribution of data to ensure efficient retrieval.

Use of HDFS enables Hadoop MapReduce to efficiently deploy computational tasks near the location of the dataset and reduces the need to transfer large datasets
across a network. Two new Cloudwave classes called CWEDFRecordReader and CWEDFWritable were defined to facilitate reading and writing ECG data from the EDF files stored in HDFS. The Cloudwave MapReduce computation module supports the use of any cardiac measurement algorithms similar to the open-source PhysioNet algorithms, which can be easily integrated as ‘pluggable’ resources using the Cloudwave CWECGSignalWrapper class. Cloudwave implements the parallelization steps using four new classes:

1. **CWECGSignalWrapper** class, which implements three methods corresponding to algorithms used for R-wave detection and calculation of RR interval and IHR values;

2. **CWECGFileInputFormat** class, which uses the CWEDFRecordReader class to access the ECG signal data from the EDF files;

3. **CWECGFileOutputFormat** class, which uses the CWEDFWritable class to transfer the results of the computations as files to be stored in HDFS;

4. **CWECGProcessor** class, which implements the Map and Reduce phases using the (key, value) pairs discussed above under dependency analysis of computational algorithms for cardiac measurements.

These new classes together with user documentation for deploying Cloudwave on Hadoop installations will be made open source as part of the PRISM project. The Cloudwave computation module effectively uses the parallelized computations to perform near real-time signal-processing computations, which allows clinicians to access analysis results more rapidly than with traditional approaches (a comparative evaluation is described in Section 7.3). Cloudwave also enables biomedical signal-processing researchers to implement algorithms on a large scale, which was previously not supported by desktop computing approaches. Clinical
Figure 7.2: (A) Cloudwave MapReduce algorithm for cardiac measurements from the ECG data. (B) Implementation of the algorithm with specialized Cloudwave classes corresponding to the Map phases and Reduce phases. EDF, European Data Format; HDFS, Hadoop Distributed File System; IHR, instantaneous heart rate.
researchers can query and visualize the processed signal data using the Cloudwave ontology-driven web-based interface.

### 7.3 Results

The comparative evaluation was performed on (1) a desktop computer with an Intel Core i7 2.93 GHz processor (16 GB main memory and 8 MB cache), (2) a single-node cluster implementation of Hadoop on the same desktop computer configuration, and (3) a Hadoop implementation on a multi-node cluster with six nodes. In the multi-node cluster implementation, the master node uses a dual quad-core Intel Xeon 5150 2.66 GHz processor, while the other nodes use dual quad-core Intel Xeon 5450 3.0 GHz processors with 16 GB of memory, and the nodes are connected by a 10 Gigabit Ethernet (GigE). The three primary objectives of the comparative test are to evaluate:

1. the time taken to identify R-waves on a single 640 MB size EDF file with subsequent computation of RR intervals and IHR values;

2. the time taken to identify the R-waves, and to compute the RR intervals and IHR cardiac measure on the largest dataset supported by the desktop computer with 3.2 GB of signal data from five EDF files; and

3. the impact of optimization approaches that divide the signal data into 10 min segments for faster signal rendering on the Cloudwave visualization and query interface.

Figure 7.3(A) shows that the multi-node Cloudwave implementation reduces the time required for computation by a factor of 3.8 (0.32 vs 1.2 min) for data from one ECG channel, and it is one order of magnitude faster (4.8 vs 0.48 min) than the desktop computer for data from four ECG channels. This dramatic improvement
Figure 7.3: Comparative evaluation of computing three cardiac measures of QRS complex detection, RR intervals, and instantaneous heart rate (IHR) values using a desktop computer and two implementations of Cloudwave over (A) 640 MB data from one European Data Format (EDF) file, (B) 3.2 GB data from five EDF files and (C) 10 min data segments (optimized for use by Cloudwave interface), and (D) scalability of multi-node Cloudwave implementation for one to four ECG channels.
in performance is also seen as the size of data increases to 3.2 GB from five EDF files, where the multi-node Cloudwave implementation is 14 times faster for data from one ECG channel than single-node implementation (0.42 vs 6.08 min) and 20 times faster than the desktop computer for data from all four ECG channels (1.57 vs 32.45 min) (Figure 7.3(B)). We also note that the time required for computations on the multi-node Cloudwave implementation increases only by a factor of 3.7, although the number of channels increases by a factor of 4, which corresponds to the expected impact of parallelization. Figure 7.3(C) illustrates the effect of a Cloudwave optimization approach that divides the signal data into 10 min segments to support efficient visualization and query of ECG data by minimizing the impact of network latency. The multi-node Cloudwave implementation is 5.3 times faster (145 vs 27 s) and 18 times faster than the desktop computer for 36 segments of data. Since the desktop computer did not support computations on the signal dataset larger than 3.2 GB, we demonstrate the scalability of Cloudwave to support larger sized data on the multi-node cluster implementation in the next section.

7.3.1 Scalability of Cloudwave on multi-node cluster

Figure 7.3(D) illustrates the performance of Cloudwave implementation on the multi-node cluster in terms of time taken for computing cardiac measures as the size of signal data increases from 3.2 to 12 GB of data for one to four ECG channels. It is clear that Cloudwave easily scales to this dataset and takes only a maximum of 3.3 min to complete the computations. This Cloudwave implementation featured only six computing nodes with a total of 50 GB disk space, which could accommodate 12 GB of data because of the Hadoop replication factor. We are in the process of increasing the available disk space to 5 TB as we continue to load all PRISM data into Cloudwave, which can be easily supported because of the extensibility of Hadoop to hundreds or thousands of nodes [12].
Chapter 8

Electrophysiological Signal

De-identification and Normalization

Data interoperability and sharing within the scientific community should address the compliance of the data stored in EDF, de-identification of patient health information (PHI), and correctness of the signal metadata for visualization and analysis purposes [74]. At Case Western Reserve University (CWRU), several research projects on sleep medicine, such as PhysioMIMI [71], PRISM [35], and NSRR [38] have invested significant effort for developing EDF tools for curation and standardization purposes. This chapter introduces a Java tool suite for EDF editing and validation.

8.1 Main features of Editor and Validator

The EDF Editor is designed to facilitate the de-identification and normalization of signal data files saved in EDF. In order for data to be meaningfully shared between investigators and institutions two obstacles must be overcome: signals must be consistently and accurately annotated, and files must be purged of potentially
identifying patient information. The editor allows investigators to edit the header portion of the EDF for de-identification, and the use of templates allows multiple files to be edited simultaneously rather than having to edit them one by one.

The EDF header is comprised of two sections, Identity Attributes and Signal Attributes described in Chapter 2. Patient identifying information is often contained in the identity attributes (file name, local patient identification, local recording information, start date of recording), and these values need to be removed or modified in order to de-identify the file as shown in Figure 8.1. The EDF Editor allows for manual editing of these fields in individual files, or the application of a template to perform the same actions to modify multiple files at once. After the file attributes the EDF header contains information on each of the channels included in the data file.

Signal attributes are left incomplete by the export process of some vendor software, and while the signal data is still in the file, without information such as the
physical minimum and maximum or the dimension the data is not useful to researchers. Further, each channel is identified by a label, a string value supplied at collection, that describes what signal is being collected on that channel. There is no standardization of these labels within or across labs, so in order to be meaningfully interpreted some normalized identification of the collected signals is necessary as shown in Figure 8.1.

Figure 8.2 shows the main application window of the Editor and Validator tool. The key features of the editor are:

- View multiple EDF identity and signal headers
- Edit EDF headers
  - Manual Edit
  - Template Application
- Access to other EDF Application Tools
- Verifying EDF file validity

### 8.2 System Workflow and Key Components

The workflow for the EDF Editor is very flexible. In order to use the EDFEditor to its fullest potential, a new user may follow these steps:

- Starting a new task
- Working with Identity Attributes template
- Working with Signal Attributes template
- EDF Validation
- Tool Help
Figure 8.2: EDF Editor Workspace Overview. 1) Menubar, 2) Toolbar, 3) Task Navigator, 4) Work area, 5) Source File and Task Summary information, 6) Log and Incompliance area, 7) Status bar
8.2.1 Starting a new task

Upon launching the EDF Editor the user is presented with the Main Window. In order to continue the user must select EDF files. This can be accomplished by selecting the menu item File/New Task or by clicking the Select EDF files icon.

Files can be selected in two ways as shown in Figure 8.3:

- By Directory: Navigate to the appropriate directory and select the directory. All EDF files in that directory and all EDF files in its subdirectories (1 level deep) will be opened; and

- By File Selections: Navigate to the appropriate directory and selecting one or more files. The shift and ctrl keys can be used with the mouse in order to select multiple files.

When the desired files are selected the user clicks Open to continue. Once files are selected the input and output directories, as well as the number of files selected, are displayed on the bottom of the main window. The user is then prompted to choose whether or not to overwrite the source files. (a) Overwrite: No further action is needed. (b) Don’t overwrite: Select an output directory into which the modified files will be saved. In this version of the EDF Editor a copy of the original file is always made and the original is left unmodified. Browse and select the directory into which modified files will be saved.

8.2.2 Working with Identity Attributes template

View Identity Attributes

The Identity Attributes for all opened EDF files can be viewed by clicking the Identity attributes tab in the Data Workspace as shown in Figure 8.4. Identity attributes are displayed as a grid with the files as rows and the attributes as columns. The
Figure 8.3: Selecting EDF Files
Figure 8.4: EDF Identity Attributes

width of columns can be adjusted by clicking and dragging the column heading boundaries.

Edit Identity Attributes

1. Manual Edit. Values can be edited by double-clicking in the desired cell and typing or pasting a new value. Note that certain fields can be viewed but are not available for editing. These fields are displayed in black. Manual editing of certain fields could cause data integrity conflicts that could render the EDF file useless; and

2. Apply File Attribute Template. Values can also be modified for all files simultaneously by applying an Identity Template. These template files contain
values or keyword instructions that are applied to each file. A template can be applied by selecting the Template/Apply Identity Template menu item or by clicking the apply template taskbar icon. A currently open template file can be selected from the list, or the user can browse for an existing template file. The files to which the template will be applied can be configured by selecting and deselecting individual files. Because the application of a template edits every selected file, it is important that these templates be properly configured. This is shown in Figure 8.5.

Before changes are saved to the files they can be undone by selecting the Edit/Undo menu item or clicking the undo taskbar icon. This will restore the value before the last edit. To undo all changes since the last save, select Edit/Discard Changes. All identity attributes will be restored to the values at the time of the last save. All changes can be saved by selecting the File/Save menu item or clicking the save taskbar icon. This will write any changes made to the Identity Attributes or the Signal Attributes to the output EDF files. Note that if you are not overwriting existing files, the first time changes are saved the output files must be created. If the files are large and/or a large number of files was selected this process can take several minutes. During this process the entire EDF file (the header as well as the signal data) is being written to disk in the selected output directory. The task progress bar will indicate that the program is executing the save process. Subsequent changes, since they only involve modifying the header of each file and not the signal data, are very quick.

From the Edit File Attributes window, the current values in each header can be exported to a CSV file. This provides for creating a lookup file between the original files and the new files. To export file attribute data, select the File/Export menu item. The identity attributes for all open EDF files will be saved into the output directory as File_Attributes_[date]_[time].csv.
Figure 8.5: Applying Identity Attributes
Configuring Identity Attribute Templates

The purpose of Identity Attribute Template files is to make the same changes to multiple files simultaneously. This can be the insertion of the same value for each file, the removal of values for each file, or other actions specified by special keywords. Only certain elements in the file attributes section of the header can be modified by use of a template. In the current version this includes the Local Patient ID, Local Recording ID, and Start Date of Recording. The Identity Template editing interface appears as a new tab in the Data Workspace when a new template file is being created or an existing file is being edited as shown in Figure 8.6.

For each field in the template, the user can either specify a value or can select from one of several key words. If the first option is selected and text is supplied,
that same text will be written to that field in each file when the template is applied. The use of keywords gives the user more flexibility in specifying a template as shown in Table 8.1. These keywords can be used in combination with one another by including the keywords in the text box for each field. For example, if it was necessary to have the template specify that for each file the new Local Recording ID should be the current file name plus ‘.’ followed by the month and year of the study, that could be accomplished by including the following in the Local Recording ID field.

\[ \{filename\} \_\{mm\} \_\{yy\} \]  

(8.1)

Note: The Start Date field is a special case where not all values may be used. The start date must retain a xx.xx.xx structure where x are integers. File Attribute Template files are ASCII-based files structured after the pattern of the EDF Header itself, and saved with the extension ‘eia’ (EDF Identity Attributes).

An existing template file can be selected for editing by selecting the Template/Open Identity Template menu item and browsing for the appropriate file on the disk. A new template can be created by scratch by selecting the Template/New Identity Template menu item. A new template can also be created by importing the values from the header of an existing EDF file. This can be used as a starting point and then the attributes of the template modified as necessary before saving. File Attribute Template files can be saved by selecting the File/Save menu item or clicking the save taskbar icon. Files can be saved with a new file name by selecting the ‘File/Save as’ menu item.

8.2.3 Working with Signal Attributes template

The Signal Attributes for a single opened EDF file can be viewed by clicking the file name from the list of open EDF files in the Task Navigator, or by clicking the Signal
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{blank}</td>
<td>Indicates that the specified field is to be set to blank. This may be particularly helpful for clearing out patient identifiers in the Local Patient ID and Local Recording ID fields. This keyword should not be applied to the Start Date field.</td>
</tr>
<tr>
<td>{skip}</td>
<td>Indicates that the specified field is to be left as it currently is in the file.</td>
</tr>
<tr>
<td>{rand}</td>
<td>Inserts a random number into the field. This keyword is useful for the creation of new identification numbers separate from any patient identifying information.</td>
</tr>
<tr>
<td>{filename}</td>
<td>Inserts the current file name (without the .edf extension) into the field.</td>
</tr>
<tr>
<td>{pid}</td>
<td>Inserts the original Local Patient ID into the field.</td>
</tr>
<tr>
<td>{rid}</td>
<td>Inserts the original Local Recording ID into the field.</td>
</tr>
<tr>
<td>{yy}</td>
<td>Inserts the year portion of the current Recording Start Date</td>
</tr>
<tr>
<td>{mm}</td>
<td>Inserts the month portion of the current Recording Start Date</td>
</tr>
<tr>
<td>{dd}</td>
<td>Inserts the day portion of the current Recording Start Date</td>
</tr>
</tbody>
</table>

*Table 8.1: Use of keywords in template creation*
Attributes tab in the Data Workspace as shown in Figure 8.7. Signal attributes are displayed as a grid with the channels as rows and the attributes as columns. The width of columns can be adjusted by clicking and dragging the column heading boundaries.

Editing signal attributes can be done in two ways as shown in Figure 8.8:

1. Manual Edit. Values can be edited by double-clicking in the desired cell and typing or pasting a new value. Note that certain fields can be viewed but are not available for editing. These fields are displayed in black. Manual editing of certain fields could cause data integrity conflicts that could render the EDF file useless. The Transducer Type field will be used to hold a standardized signal type name, which is a combination of a procedure, a device, a location
(when applicable), and a reference location (when applicable). To facilitate creating these standard signal types, a Create Transducer Label expression builder has been built into the editor. The builder is launched by pressing and holding the ALT-key and clicking on the Transducer Type field in the Signal Attributes grid; and

2. Apply Signal Attribute Template. Values can also be modified for all files simultaneously by applying a Signal Template. These template files contain values or keyword instructions that are applied to each file. A template can be applied by selecting the Template/Apply Signal Template menu item or by clicking the apply template taskbar icon. A currently open template file can be selected from the list, or the user can browse for an existing template file. The files to which the template will be applied can be configured by selecting and deselecting individual files. Because the application of a template edits every selected file, it is important that these templates be properly configured.

Before changes are saved to the files they can be undone by selecting the Edit/Undo menu item or clicking the undo taskbar icon. This will restore the value before the last edit. To undo all changes since the last save, select Edit/Discard Changes. All signal attributes will be restored to the values at the time of the last save. All changes can be saved by selecting the File/Save menu item or clicking the save taskbar icon. This will write any changes made to the Identity Attributes or the Signal Attributes to the output EDF files. Note that if you are not overwriting existing files, the first time changes are saved the output files must be created. If the files are large and/or a large number of files was selected this process can take several minutes. During this process the entire EDF file (the header as well as the signal data) is being written to disk in the selected output directory. The task progress bar will indicate that the program is executing the save process. Subse-
Figure 8.8: Applying Signal Attributes
quent changes, since they only involve modifying the header of each file and not the signal data, are very quick.

**Configuring Signal Attribute Templates**

The purpose of Signal Template files is to provide a mapping between signal labels and standardized attributes which can then be used to modify the values for signals in individual EDF files. A Signal Template may contain many more channels than are found in any particular file, and in this respect it is more of a mapping or a lookup file. Because the order of channels can change from study to study, the label is used as the lookup for the Signal Template, and the order of signals in the template does not matter. When applied, for each channel in each file the EDF Editor will attempt to find a corresponding label in the selected Signal Template. If a match is found the other attributes (Transducer Type, Physical Dimension, Physical Minimum, Physical Maximum, Digital Minimum, Digital Maximum) will be inserted for that channel, overwriting the original values. If a particular channel label is not found in the template its values are retained.

The physical dimension and the physical and digital ranges are essential to the proper interpretation of the resulting EDF file, and must be completed with great care. EDF files generated by some vendors are populated with incorrect values for the physical range and dimensions. Users are encouraged to consult PSG technicians to determine the appropriate collection settings for each signal. The current version of the EDF Editor now contains an additional column for a corrected label. This allows users to rename channels to standard names by means of a template. Signal Template files are ASCII-based files saved with the extension .esa (EDF Signal Attributes). This is shown in Figure 8.9.

An existing template file can be selected for editing by selecting the Template/Open Signal Template menu item and browsing for the appropriate file on the disk. A
Figure 8.9: EDF Signal Attributes Template
new template can be created by scratch by selecting the Template/New Signal Template menu item. This presents the user with a blank grid of the Signal Attributes template fields with only one row. New rows can be inserted by selecting the Edit/Add row menu item or clicking the new row taskbar icon. The currently-selected row can be deleted by selecting the Edit/Remove row menu item or clicking the remove row taskbar icon. A new template can also be created by importing the values from the header of an existing EDF file. This can be used as a starting point and then the attributes of the template modified as necessary before saving. Signal Template files can be saved by selecting the File/Save menu item or clicking the save taskbar icon. Files can be saved with a new file name by selecting the ‘File/Save as’ menu item.

**EDF Validation**

Upon launching the EDF Editor the user is presented with the Main Window. In order to continue the user must select EDF file(s). This can be done by selecting the menu item File/New Task or by clicking the Select EDF files icon. EDF Validation Tool can be invoked by selecting the menu item Tools/Find EDF Header Errors icon as shown in Figure 8.10.

The error messages are shown in the ‘Log and Incompliance area’ of EDF-
Editor. Another copy of the same error messages will be recorded into the error log file. For every error message, the validator gives detailed information including (1) type of examination (EIA or ESA), (2) description of error, (3) path and name of erroneous EDF file, and (4) location (with row and column numbers) of the error in EDF table. User may point to and click on the arrow on every error message, and then corresponding EDF file is opened and erroneous locations is highlighted.

A brief explanation to the errors shown in Figure 8.11.

User specify what kind of errors and which EDF files should be fixed using the Error Fix Component. Having parameters set, user click the button ‘Apply Error-fixes’ to correct errors in the selected EDF files. A complete list of error-fix
messages will be recorded into the log file. ‘Swap Physical Max/min’ will swap ‘Physical maximum’ and ‘Physical minimum’ for negative amplifier gain scenario. In this case, the physical maximum is smaller than physical minimum as shown in Figure 8.12.

Tool Help

There are three ways to access the help files included with the EDFEditor.

1. Go to Help/How to Use;

2. Pressing F1 key on your keyboard; and

3. Clicking on ‘help icon image’ and clicking on different parts of the editor will give you the help file for that specific part of the editor.
Chapter 9

Conclusion and Future Work

Cloudwave is a generic, domain-agnostic signal management platform that can be used in a variety of medical disciplines that need to manage ‘Big Data,’ such as sleep medicine and neurological disorders. Electrophysiological signal data are increasingly characterized by both massive volume and high velocity, as they play a greater role in supporting patient care and clinical research. This dissertation has demonstrated the functions of Cloudwave in the context of epilepsy research for management of electrophysiological big data. The performance of processing such large datasets using Cloud based solutions for storage, web-based access, visualization and analysis were presented in Chapters 4, 6, 5, 7.

Chapter 4 has demonstrated Cloudwave as an emerging computing platform to meet the growing needs for multi-center collaborative studies in clinical research using large datasets, such as electrophysiological datasets, with interactive performance in processing and rendering requested data on the user interface. The results of the comparative performance evaluation of Cloudwave clearly demonstrate the significant advantages of using the Hadoop distributed computing platform for processing very large electrophysiological signal datasets.

Chapter 5 presented Cloudwave web-interface that accesses very large signal
data files from high performance distributed file systems. Implementation details of various features of the interface, such as clinical events annotated displaying of signals, configuration for selecting montages and signals for display, and filtering were presented. Results of comparative evaluation demonstrates that Cloudwave is a practical tool and hence can be deployed at epilepsy centers, for example at UH-CMC.

Chapter 6 described how EpSO was integrated into Cloudwave framework, not only to efficiently retrieve specific data segments based on interesting clinical events, but also to define a new signal representation model called CSF. A comparative evaluation of CSF with EDF binary format clearly demonstrates a significant reduction in data access time for increasing size of signal data for six standard montages implemented in the Cloudwave signal visualization module.

Chapter 7 demonstrated that computations took less time using Cloudwave when compared with the desktop approaches for measuring cardiac functions, such as QRS detection, RR and IHR. Cloudwave reduced the time by a factor of 3.8(0.32 vs. 1.2 minutes) for data from on ECG channel and one order of magnitude faster (4.8 vs. 0.48 min) than desktop computer from four ECG channels.

Chapter 8 demonstrated the use of EDF preprocessing tools for de-identification of PHI and normalization of signal metadata.

### 9.1 Future Work

Cloudwave data management framework will address the following needs for advanced query and analysis of large-scale signal datasets:

1. High Performance computing on Amazon EC2, S3 using Elastic MapReduce (EMR) for 10TB electrophysiology data with 100s of computing nodes;

2. Cloudwave Signal Toolkit. This toolkit will provide additional quantitative
signal analysis functionalities using MapReduce described in Chapter 7, such as quantification of respiratory arrhythmia, EEG suppression and percentage of seizure occurrences over multiple visits;

3. Efficient query execution strategy using distributed multilevel indexing on signal datasets;

4. Integration with MEDCIS Visage interface for query on signal datasets

Cloudwave will be deployed in other participating centers as part of PRISM project. Also Cloudwave classes and Cloudwave Signal Format for electrophysiology data management will be made publicly available for the research community.
Appendix A

Hadoop MapReduce classes for processing EDF to CSF

The Java source code will be released through GitHub.

A.1 Class: EDFProcessor

EDFProcessor is the MapReduce job class that submits the job. This class defines the mapper, reducer, input and output formats, HDFS reader and writer classes for processing EDF files.

Fields:

JobConfiguration

Constructors:

Job
Methods:

(a) **Main.** *Input:* EDF input directory, CSF output directory. *Output:* Job completion status.

A.2 **Class: EDFMapper**

EDFMapper runs the map task for the job. Each map is assigned an EDF file for execution by the input format. Extends Mapper.

**Fields:**

None

**Constructors:**

Default

**Methods:**

(a) **map.** *Input:* EDF filename, EDF file contents. *Output:* Context(signal key, signal segment CSF).

(b) **toIntArray.** *Input:* signal segment in binary format. *Output:* signal segment in numeric format.

(c) **toPhyArray.** *Input:* signal segment with digital values, physical minimum, physical maximum, digital minimum and digital maximum. *Output:* signal segment with physical values.

(d) **generatePatientCSF.** *Input:* EDF header. *Output:* Patient CSF JSON object.
(e) \texttt{generateStudyCSF}. \textit{Input}: EDF header. \textit{Output}: Study CSF JSON object.

(f) \texttt{generateSignalCSF}. \textit{Input}: EDF header. \textit{Output}: Signal CSF JSON object.

### A.3 Class: EDFReducer

EDFReducer combines the outputs generated by the map task. Extends Reducer.

**Fields:**

None

**Constructors:**

Default

**Methods:**

(a) \texttt{reduce}. \textit{Input}: signal key, signal segments. \textit{Output}: Context(signal key, signal CSF).

### A.4 Class: EDFFileInputFormat

This class defines the input format for the MapReduce job. Extends FileInputFormat.
Fields:

None

Constructors:

Default

Methods:

(a) `isSplittable`. *Input*: JobContext, Path. *Output*: false

(b) `createRecordReader`. *Input*: InputSplit, TaskAttemptContext. *Output*: EDFRecordReader instance.

### A.5 Class: EDFRecordReader

The EDFRecordReader class reads the entire EDF file for input to the EDFMapper. The key is denoted by the file name and the value is the EDF metadata and raw signal data. Extends RecordReader.

Fields:

FileSplit, Configuration, current key, current value, file processed status

Constructors:

Default
Methods:


(b) **nextKeyValue**. *Input*: none. *Output*: fileProcessed status.

(c) **getCurrentKey**. *Input*: none. *Output*: current key.

(d) **getCurrentValue**. *Input*: none. *Output*: current value.

A.6 **Class: EDFFileOutputFormat**

The EDFFileOutputFormat class defines the JSON-based CSF format for writing output files. Extends FileOutputFormat.

**Fields:**

None

**Constructors:**

Default

**Methods:**

(a) **getRecordWriter**. *Input*: TaskAttemptContext. *Output*: EDFRecordWriter instance.

A.7 **Class: EDFRecordWriter**

The EDFRecordWriter class writes the patient, study and signal JSON objects to the specified output location on HDFS for each signal in the EDF file.
Extends RecordWriter.

**Fields:**

None

**Constructors:**

EDFRecordWriter(TaskAttemptContext)

**Methods:**

(a) **write.** *Input:* CSF file path. *Output:* CSF file contents.

(b) **close.** *Input:* TaskAttemptContext. *Output:* none.
Bibliography


[14] L. Cui, A. Bozorgi, S. D. Lhatoo, G.-Q. Zhang, and S. S. Sahoo. Epi-DEA: extracting structured epilepsy and seizure information from pa-


[100] C. Wedekind, V. Hesselmann, and N. Klug. Comparison of MRI and electrophysiological studies for detecting brainstem lesions in trau-


son, D. Hunscher, and S. Arabandi. VISAGE: a query interface for
clinical research. *AMIA Summits on Translational Science Proceedings,*

[113] M. Zijlmans, D. Flanagan, and J. Gotman. Heart rate changes and ECG
abnormalities during epileptic seizures: Prevalence and definition of