HBase, MapReduce, and Integrated Data Visualization for Processing Clinical Signal Data

Andrew V. Nguyen, Rob Wynden, Yao Sun
University of California, San Francisco, Clinical and Translational Informatics
andrew-aaai-hbase@ucsfcti.org

Abstract

Processing high-density clinical signal data (data from biomedical sensors deployed in the clinical environment) is resource intensive and time consuming. We propose a novel approach to storing and processing clinical signal data based on the Apache HBase distributed column-store and the MapReduce programming paradigm with an integrated web-based data visualization layer. An integrated solution negates the need to marshal data into and out of the storage system while also easily parallelizing the computation, a problem that is becoming more and more important due to increasing numbers of sensors and resulting data. We estimate upwards of 50TB of clinical signal data for a 200-bed medical center within the next 5 years. Consequently, efficient processing of clinical signal data is a vital step towards multivariate analysis of the signal data in order to develop better ways of describing a patient's clinical status.

Background

Clinical Significance

Computational physiology is relatively new and there are increasing efforts towards using physiologic signal data to improve the quality of care being provided to patients (Saeed, Lieu, and Mark 2002; Stanley et al. 2000; Barnaby et al. 2002; Björkander et al. 2009; Buchman 1996; Garrard, Kontoyannis, and Piepoli 1993; Sorani et al. 2007; Staats, Austin, and Aboy 2008; Sun, Reisner, and Mark 2006; Agarwal et al. 1998; Baselli et al. 1987; Moca et al. 2009; de Godoy et al. 2009; Karamanoglu 1997; Lake et al. 2002). The literature is filled with a multitude of varying approaches: some are looking to create models(Staats, Austin, and Aboy 2008; Aleks et al. 2008) while others are attempting a more feature-based approach (Garrard, Kontoyannis, and Piepoli 1993).

Our current clinical problem of interest is the more accurate detection of cardiac arrhythmias using a multivariate, multi-feature approach. Most arrhythmia detection algorithms focus solely on the ECG waveforms, ignoring all other physiologic or clinical signal data. However, there are several examples in the literature that take a multivariate approach using a combination of signals (e.g. ECG with arterial pressure) to reduce the number of false alarms (Clifford et al. 2006; Aboukhalil et al. 2008).

To further investigate the use of a multivariate, multi-feature approach to cardiac arrhythmia detection, we have chosen to use a subset of the MIMICII dataset (Saeed, Lieu, and Mark 2002) that is released as part of the PhysioBank project (Stanley et al. 2000). This subset consists of both waveform data (recorded sampling frequency of 125 Hz) as well as numeric time-series data (sampled at 1 minute intervals) from nearly 500 patient-records, totaling approximately 45,000 hours of waveform data. Other clinical data, such as the laboratory, pharmacy, and nursing databases, are also available and time-correlated with the signal data. All of the data have been de-identified.

Our general approach consists of applying different classes of feature extraction algorithms to the signal data and then feeding these through machine learning algorithms to develop better algorithms for detecting cardiac arrhythmias. The specifics of our approach, however, are beyond the scope of this paper.

When considering the application of feature extraction algorithms over all of the waveform data, the issue of computational efficiency becomes a major concern. For example, the Sliding Window Central Tendency Measure takes approximately 20 minutes to run on a particular patient record (Nguyen, Hudson, and Cohen 2010; Cohen, Hudson, and Deedwania 1998). Multiplying this over 500 patient records results in almost 7 days of computation time. When implemented as a MapReduce job within the Hadoop environment, the Sliding Window Central Ten-

dency Measure had a run time of 3 minutes, saving almost 5 days of computation time if extrapolated over the entire dataset.

We propose the creation of a standard platform to simplify the computational approach to working with clinical signal data by combining the storage, processing, analysis, and visualization of large datasets into a single platform called a Signal Archiving and Computation System™ or SACS™. This paper describes one prototypical implementation of a SACS™.

MapReduce

Map and Reduce are not new concepts – they are common to many functional programming languages such as Lisp or Scheme. Google recently popularized the use of Map and Reduce as a simpler solution for parallelizing computation (Dean and Ghemawat 2004) for a certain subset of problems compared to other approaches such as MPI (The MPI Forum 1993) or PVM (Sunderam 1990).

One major benefit of the MapReduce approach is the ability to focus solely on the computation, and not the shuffling of data between processors. The programmer only needs to worry about the computation itself, and can assume that the data will be available as required. This allows those who have some programming experience to create and run jobs without extensive training in parallel computing.

The second major benefit of MapReduce is in data-locality. With the MapReduce paradigm, most of the computation is done on the slave node that contains a copy of the input data. This results in a minimal amount of data being sent over the network, increasing overall efficiency.

Generally, each "job" that needs to be run is split into two separate tasks – a map task and a reduce task. The map task is a user-defined function and handles the initial "mapping" of the input data, taking a <key1, value1> pair as input and outputting an intermediate <key2, value2> pair. Oftentimes, the map task maps multiple different values to the same key.

The reduce task is also user-defined and takes the intermediate <key2, value2> pairs and merges all of the values that correspond to the same key. One example of a reduce task is to take the running sum or mean of values with the same key.

Hadoop

Hadoop is an open-source implementation of the MapReduce parallel programming paradigm and is backed by both the open-source community as well as major companies such as Yahoo and Facebook.

Hadoop provides both a distributed file system (called HDFS) as well as the MapReduce parallel computation framework. Data are stored in the HDFS and made available to the various slave nodes for computation.

Hadoop is an Apache Foundation project and is written in Java though hooks are in place to facilitate the deployment of code written in other languages such as C or Python. Hadoop is a master-slave architecture where a single master node coordinates many slave machines which carry out the actual computation.

To enable data-local processing, each slave machine only computes on data of which it has a copy. This results in very little shuffling of data over the network, decreasing the network IO bandwidth needed.

Additional nodes can be added to the cluster to increase storage capacity or computational power as necessary. Hadoop has been used in clusters ranging from a handful of nodes to over 4000 nodes, demonstrating it's ability to scale as needed.

HBase and MongoDB

HBase is a distributed column-store that runs on top of the Hadoop system. As a result, it depends on (and takes advantage of) the distributed file system as well as the MapReduce framework. HBase provides a storage system where the full power of the MapReduce paradigm is available while also providing convenient, random-access to the data. There are other possible alternatives within the Hadoop project that may serve a similar purpose such as Pig and Hive though we have chosen to focus on HBase.

As a "NoSQL" database, HBase's approach to tables, rows, and columns is different than in traditional relational databases. Within HBase, each row is identified by a sortable row key. Each row can contain an arbitrary number of columns resulting in the sparse storage of tables. The columns are identified by a combination of "column family" and "label." The column family is important during schema design because data are stored in a per-column family basis. When a value is written to the database, it is stored with its row key, column family and label identifiers. This results in substantial storage overhead if the identifiers are large in size.

MongoDB is a document-store database and also has an integrated MapReduce computational framework. It is also a "NoSQL" database and is designed to be deployed in a clustered environment.

System Description

The integrated platform is built with four primary components (Figure 1): 1) Data storage using HBase, 2) metadata storage with MongoDB, 3) MapReduce using Hadoop, and 4) data visualization using Chronoscope with the Google Web Toolkit.

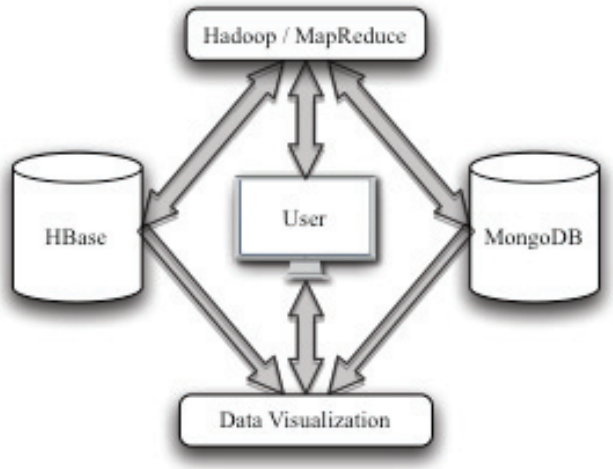

Figure 1 - System Overview

The signal data are stored in HBase while the signal metadata and other clinical data are stored in MongoDB. Data in both HBase and MongoDB are accessible from the Hadoop/MapReduce environment for processing as well as from the data visualization layer.

We currently have a prototype SACS™ deployed in a local UCSF datacenter which consists of 1 master node, 6 slave nodes, and several supporting servers. While some initial work was performed in the Amazon EC2 cloud environment, the majority of our work has taken place on this dedicated cluster.

Data Storage with HBase and MongoDB

The data are being stored within HBase as time-series data. We have elected to store the time-series data within HBase such that each row key is the timestamp of a signal's value at a particular point in time. This timestamp is recorded as an 8-byte value of the number of milliseconds from the epoch (January 1, 1970 00:00:00.000).

Within each row, each column contains the value of a particular signal for a particular patient corresponding to the row key timestamp. For example, each of the following would be stored in a separate column: arterial blood pressure values for patient A, arterial blood pressure values for patient B, and ECG Lead I values for patient A.

The columns can also contain the values resulting from different feature extraction algorithms. For example, one particular feature extraction algorithm extracts the beat-to-beat systolic pressures from an arterial blood pressure waveform. In this case, the column contains the systolic pressure value at a particular timestamp for a particular patient.

Inherent in the architecture of HBase, each value is stored along with its row key and column identifier. As a result, there is a noticeable increase in storage overhead. To mitigate this, we have chosen to store the identifiers in their binary form. Because the row key identifiers are just numeric values representing the number of milliseconds from the epoch, they are stored as standard 8-byte binary data. The column identifiers, on the other hand, are a combination of a patient identifier as well as a signal identifier. Given the anticipated size of the data that we are working with, we have limited the number of patients to a 2-byte value and the signal identifier to a 4-byte value. These values, however, are easily changed should the underlying dataset require it.

Because the patient and signal identifiers are not intuitive when stored as numeric values, we have setup a database to store the metadata. We are currently using an instance of MongoDB though any standard relational database would work. This database contains all of the patient demographic information, patient clinical data (e.g. data from other clinical databases) as well as the mappings to the column identifiers used in HBase.

MapReduce with Hadoop

Because HBase runs within the Hadoop environment, we can directly leverage the MapReduce framework for data stored within HBase. When writing a MapReduce job for processing the signal data, the user only needs to provide the code for the actual computation. While Hadoop handles the low level management of the data, we have also provided several Java classes to handle the selection of the appropriate row keys and column identifiers. The user only needs to write the actual computational code and select the input columns, dramatically simplifying the process.

Given the nature of the MapReduce approach and the Hadoop implementation, there is one potential issue that must be addressed when working with feature extraction algorithms that are not stateless. Because each slave machine only processes data that it contains, there is no default mechanism to communicate the status or results of computation occurring on another slave. So, for an algorithm such as the rolling mean, each slave machine is computing its own rolling mean of the subset of data it has a copy of, independent of any of the other slave machines.

One general solution is to force each map task to process an entire patient record. This, however, negates the data-locality benefit since a single slave machine will be requesting data stored on other slave machines.

For problems that only require some overlap of data (e.g. only the previous 5 data points are needed), there are hooks built-in to the Hadoop system that allow for custom "splitting" of the data. The splits can be made such that there is adequate overlap between one map task and the next. While this results in some additional shuffling of

data over the network, the majority of the computation is still data-local.

Data Visualization

In order to visualize both the raw signal data as well as the extracted features, we integrated a web-based visualization tool into the system. This provides the ability to easily visualize the data without having to extract it into a separate plotting tool.

The visualization layer is built using the Google Web Toolkit (GWT) along with Chronoscope (a charting tool) from Timepedia.

Users specify which signals they want to plot, both raw signals as well as derived signals from the feature extraction algorithms, along with the time period of interest. The visualization tool has direct access to the MongoDB instance in order to properly map the binary identifiers to their human-readable counterparts. While rudimentary, our data visualization system also has the ability to overlay annotations on top of the signal data, allowing us to display pertinent events from the patient's clinical history over the signal data.

Example

In previous work, we had used the Hadoop MapReduce system to more efficiently calculate the Sliding Window Central Tendency Measure (CTM). While this dramatically increased our computational efficiency, our workflow was still suboptimal due to the separate processing and visualization environments. As with most sliding window applications, some experimentation is needed in order to determine the optimal window size. Each time we changed the window size, we wanted to plot the Sliding Window CTM against other window sizes, the original arterial pressure signal or other physiologic signals.

We first ran the Sliding Window CTM calculation within the Matlab environment to take advantage of both its processing and visualization capabilities. However, the Sliding Window CTM was very slow when run via Matlab. To take advantage of the Hadoop MapReduce environment, we imported the data into Hadoop and processed them via MapReduce. In order to visualize the data, we first needed to export it from Hadoop and then import it into Matlab for plotting. Because we were experimenting with many different window sizes, both of these approaches were tedious and time-consuming.

Our prototype SACS™ addresses both of these issues by providing an environment that can efficiently process the data while making it immediately available for viewing. This allowed us to experiment with various parameters of the Sliding Window CTM in a fraction of the time otherwise. Funds permitting, we could add additional nodes to the cluster as necessary to further improve computation time.

Conclusion

The Signal Archiving and Computation System™ outlined in this paper has become an effective method for working with larger volumes of high-density signal data. By parallelizing the work via Hadoop and MapReduce, we are able to process the data in a fraction of the time it would take serially. Hadoop and MapReduce provide a simpler, yet powerful, means of parallelizing the signal processing, including the ability to add more server nodes as necessary to increase storage space or computational resources. This helps informatics researchers interact with the data more efficiently, allowing us to work on problems that were unfeasible a few years ago. The use of Hadoop also allows us to leverage developments such as Online MapReduce (Condie et al. 2009) allowing us to translate developed algorithms into real-time, online algorithms for clinical decision support.

By integrating visualization into the processing environment, researchers can conveniently view the signal data (both the raw data as well as the extracted features) alongside the patient's clinical history, making the workflow much more efficient and facilitating the discovery process.

While there are many other tools and platforms that could also be used, Hadoop, HBase, GWT, and Chronoscope have blended well together, allowing us to create an initial implementation of a SACS™ to facilitate informatics research in the area of physiologic signal data analysis.

References

Aboukhalil, Anton, Larry Nielsen, Mohammed Saeed, Roger G Mark, and Gari D Clifford. 2008. Reducing false alarm rates for critical arrhythmias using the arterial blood pressure waveform. *Journal of Biomedical Informatics* 41, no. 3 (June): 442-451.

Agarwal, Rajeev, Jean Gotman, Danny Flanagan, and Bernard Rosenblatt. 1998. Automatic EEG analysis during long-term monitoring in the ICU. *Electroencephalography and Clinical Neurophysiology* 107, no. 1 (July): 44-58.

Aleks, N., S. Russell, M. G Madden, D. Morabito, K. Staudenmayer, M. Cohen, and G. Manley. 2008. Probabilistic detection of short events, with application to critical care monitoring. In *Proceedings of NIPS 2008: 22nd Annual Conference on Neural Information Processing Systems*. Vancouver, Canada.

Barnaby, Douglas, Kevin Ferrick, Daniel T. Kaplan, Sachin Shah, Polly Bijur, and E. John Gallagher. 2002. Heart Rate Variability in Emergency Department Patients with Sepsis. *Academic Emergency Medicine* 9, no. 7: 661-670.

Baselli, G., S. Cerutti, S. Civardi, F. Lombardi, A. Malliani, M. Merri, M. Pagani, and G. Rizzo. 1987. Heart rate variability signal processing: A quantitative approach as an aid to diagnosis in cardiovascular pathologies. *International Journal of Bio-Medical Computing* 20, no. 1 (January): 51-70.

Björkander, Inge, Lennart Forslund, Mats Ericson, Nina Rehnqvist, Paul Hjemdahl, and Thomas Kahan. 2009. Long-term stability of heart rate variability in chronic stable angina pectoris, and the impact of an acute myocardial infarction. *Clinical Physiology and Functional Imaging* 29, no. 3 (May): 201-208.

Buchman, T. G. 1996. Physiologic stability and physiologic state. *The Journal of trauma* 41, no. 4: 599.

Clifford, G.D., A. Aboukhalil, J.X. Sun, W. Zong, B.A. Janz, G.B. Moody, and R.G. Mark. 2006. Using the blood pressure waveform to reduce critical false ECG alarms. In *Computers in Cardiology, 2006*, 829-832.

Cohen, ME, DL Hudson, and PC Deedwania. 1998. The use of continuous chaotic modeling in differentiation of categories of heart disease. In *7th Information processing and management of uncertainty in knowledge-based systems Conference*, 548-554. EDK, Paris, FRANCE.

Condie, Tyson, Neil Conway, Peter Alvaro, Joseph M Hellerstein, Khaled Elmeleegy, and Russell Sears. 2009. *MapReduce Online*. EECS Department, University of California, Berkeley, October.

Dean, Jeffrey, and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*. San Francisco, CA, December.

Garrard, Christopher S., Dimitrios A. Kontoyannis, and Massimo Piepoli. 1993. Spectral analysis of heart rate variability in the sepsis syndrome. *Clinical Autonomic Research* 3, no. 1 (February 1): 5-13.

de Godoy, Moacir Fernandes, Isabela Thomaz Takakura, Paulo Rogério Correa, Mauricio de Nassau Machado, Rafael Carlos Miranda, and Antonio Carlos Brandi. 2009. Preoperative nonlinear behavior in heart rate variability predicts morbidity and mortality after coronary artery bypass graft surgery. *Medical Science Monitor: International Medical Journal of Experimental and Clinical Research* 15, no. 3 (March): CR117-122.

Karamanoglu, Mustafa. 1997. A System for Analysis of Arterial Blood Pressure Waveforms in Humans. *Computers and Biomedical Research* 30, no. 3 (June): 244-255.

Lake, Douglas E., Joshua S. Richman, M. Pamela Griffin, and J. Randall Moorman. 2002. Sample entropy analysis of neonatal heart rate variability. *Am J Physiol Regul Integr Comp Physiol* 283, no. 3 (September 1): R789-797.

Moca, Vasile V., Bertram Scheller, Raul C. Muresan, Michael Daunderer, and Gordon Pipa. 2009. EEG under anesthesia--Feature extraction with TESPAR. *Computer Methods and Programs in Biomedicine* 95, no. 3 (September): 191-202.

Nguyen, A. V., Hudson, D. L., and Cohen, M. E.. 2010. Hadoop and MapReduce for the storage and processing of physiologic data. In *SEDE 2010: 19th International Conference on Software Engineering and Data Engineering*. San Francisco.

Saeed, M., C. and Raber Lieu, and R. G Mark. 2002. MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring. *Computers in Cardiology* 29 (September): 641-644.

Sorani, Marco, J. Hemphill, Diane Morabito, Guy Rosenthal, and Geoffrey Manley. 2007. New Approaches to Physiological Informatics in Neurocritical Care. *Neurocritical Care* 7, no. 1: 45-52.

Staats, C, D Austin, and M Aboy. 2008. A statistical model and simulator for cardiovascular pressure signals. *Proceedings of the Institution of Mechanical Engineers. Part H, Journal of Engineering in Medicine* 222, no. 6 (August): 991-998.

Stanley, H. E, A. L Goldberger, L. A. N Amaral, L. Glass, J. M Hausdorff, P. Ch Ivanov, R. G Mark, J. E Mietus, G. B Moody, and C. -K Peng. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101, no. 23 (June 13): e215–e220.

Sun, J.X., A.T. Reisner, and R.G. Mark. 2006. A signal abnormality index for arterial blood pressure waveforms. In *Computers in Cardiology, 2006*, 13-16.

Sunderam, V. S. 1990. PVM: A framework for parallel distributed computing. *Concurrency: Practice and Experience* 2, no. 4 (12): 315-339.

The MPI Forum. 1993. MPI: A Message Passing Interface.