Scaling the PhysioNet WFDB Toolbox for MATLAB and Octave

Tristan Naumann[1], Ikaro Silva[2]

[1]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA

[2] MIT Laboratory for Computational Physiology, Cambridge, MA, USA

Abstract

The PhysioNet WaveFormDataBase (WFDB) Toolbox for MATLAB and Octave is a collection of functions for reading, writing, and processing physiologic signals and time series used by PhysioBank databases. Using the WFDB Toolbox, researchers have access to over 50 PhysioBank databases consisting of over 3TB of physiologic signals. These signals include ECG, EEG, EMG, fetal ECG, PLETH (PPG), ABP, respiration, and others.

The WFDB Toolbox provides support for local concurrency; however, it does not currently support distributed computing environments. Consequently, researchers are unable to utilize the additional resources afforded by popular distributed frameworks. The present work improves the scalability of the WFDB Toolbox by adding support for distributed environments. An example is shown of surrogate data significance testing. The example uses StarCluster to launch, within minutes, Hadoop Streaming on a newly created Amazon EC2 cluster with minimum configuration. The results demonstrate up to 30 fold performance increases can be achieved compared to single node processing.

1. Introduction

The recent abundance of cloud offerings has made it easier and more affordable than ever for researchers to obtain the computational resources necessary for grappling with "big data." However, this opportunity also presents a significant challenge: many existing tools do not support the distributed computing environments provided by cloud solutions. The WFDB Toolbox for MATLAB and Octave is one such tool [1]. While commonly used to analyze the signals and time series available in the PhysioBank databases, its usage does not currently scale beyond a single machine.

This ongoing work aims to provide simple instruction and examples for researchers looking to leverage the WFDB Toolbox in distributed environments. Specifically, this paper discusses some of the tools that can be used and a corresponding codebase available on PhysioNet provides more detailed guidance with respect to: (1) deploying an appropriate cluster, (2) working with it, and (3) running code on it.

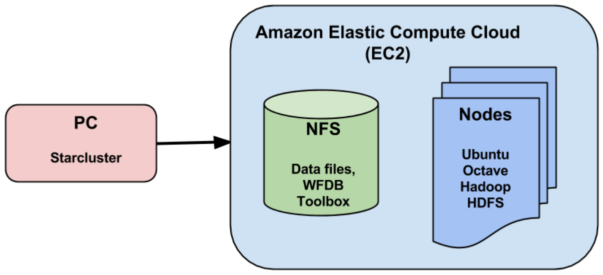

Figure 1. Proposed software architecture for processing PhysioNet data with Amazon EC2. StarCluster automatically configures the EC2 instance(s), requiring minimal additional configuration from the user. Default standard templates for different cluster size (number of nodes and instance types) are provided in the corresponding codebase.

2. Methods

A schematic diagram of our proposed software stack is shown in Figure 1. Amazon's Elastic Compute Cloud (Amazon EC2) provides re-sizable compute capacity in the cloud and related services provide the other necessary resources (e.g. storage) for working with big data. While only one of the many options available to researchers, Amazon EC2 is simple to use and has a large software ecosystem. As such, the present work uses this platform, but should likely generalize to others.

On top of this platform, Apache Hadoop [2] is used to facilitate distributed computation and StarCluster [3] is used for the purposes of deploying and managing an appropriate cluster.

2.1. Hadoop

Apache Hadoop is an open-source framework for storage and processing large data sets. Namely, it supports computation and storage abstractions derived from Google's MapReduce and Google File System [4–6]. While it was originally designed to be run on clusters of consumer hardware, it is often run in cloud environments as well.

Being a mature software project, Hadoop is both powerful and extremely configurable. In the context of distributing WFDB Toolbox computation, however, it is primarily useful to leverage only two of its components: (1) Hadoop MapReduce to "map" WFDB Toolbox commands across PhysioBank files stored in NFS, and (2) Hadoop Distributed File System (HDFS) for storing the results of the computation performed as a result of each "map." In this manner, Hadoop is used to solely to coordinate the parallel execution of commands across the cluster and store the resulting output while the input lies on storage accessible to all nodes in the cluster.

2.2. StarCluster

Deploying and managing a cluster can be a difficult task. Even when facilitated by the tools provided by Amazon EC2, researchers need to concern themselves with the minutiae of provisioning resources, configuring them appropriately, and making them aware of one another.

StarCluster is a utility for creating and managing computing clusters hosted on Amazon EC2 with sensible defaults. Therefore, it is used to facilitate deploying and configuring environments suitable for the WFDB Toolbox.

3. Processing PhysioNet Data in EC2

This paper's corresponding codebase contains step-by-step instructions, WFDB-specific cluster configuration templates, and code used to run the following examples.

In the examples, we use PhysioNet's Massachusetts General Hospital/Marquette Foundation Waveform Database (MGHDB). The MGHDB consists of records from 250 patients with a wide range of pathophysiologic states. These records vary in length starting from 12 to up to 86 minutes. Most cases are about an hour long.

3.1. Cluster Architectures

Clusters of different sizes, according to Table 1, were used to run the examples in this article. These configurations use three Amazon EC2 instance types, namely:

- *t1.micro*: 1 vCPU, 0.613 GiB RAM per node. Ideal for experimenting while leveraging Amazon's "Free Tier" usage.
- *m1.small*: 1 vCPU, 1.7 GiB RAM per node. A good general-purpose instance type with a balance of compute and memory.
- *c1.xlarge*: 8 vCPU, 7 GiB RAM per node. An instance type with high performing processors ideal for computationally heavy loads.

The nodes on each of the clusters were configured through the StarCluster to automaticaly boot with the following software suite installed: Ubuntu 12.04, GNU Octave Version 3.2.4, Hadoop 0.20.2, and Java version 1.6. The WFDB Toolbox version 10.5.23 and the PhysioNet database MGHDB were installed through a single shell script available in the code repository of this project. Both the WFDB software package and the PhyioNet data were stored on a shared NFS mount, accessible to all the nodes on a given cluster. The NFS mount point consisted of a 20 GB Amazon Elastic Block Store (EBS) volume. This mechanism provided a persistent and scalable storage volume. For each cluster configuration, a single node (i.e. "sequential mode") was also run with each of the examples in order to provide a reference benchmark.

Table 1. Cluster specifications. Several instance types and sizes were used ranging from a single t1.micro instance, up to eight of of the c1.xlarge instances.

Cluster	Instance Type	Size
wfdbcluster	t1.micro	1
wfdbcluster-small	m1.samll	2
wfdbcluster-medium	m1.small	4
wfdbcluster-large	m1.small	8
wfdbcluster-small-fast	c1.xlarge	2
wfdbcluster-medium-fast	c1.xlarge	4
wfdbcluster-large-fast	c1.xlarge	8

3.2. Example1: Spectral indexing

As a first example, a simple spectral indexing task inspired from music information retrieval systems was applied to all signals from the MGHDB [7]. The spectral indexing task consisted of a single map that found the 3 largest peaks on the spectrum of a given signal. Peaks were considered separate if they were more than 50 Hz apart. The main purpose of this example was to quantify the amount of overhead associated with a distributed computing environment in EC2 versus running the process on a single serial instance for tasks with minimum computation.

3.3. Example2: Surrogate Series Test for Non-linearity

A more realistic and computationally intensive example involves the use of Mont Carlo simulations for bootstrapping statistical tests of non-linearity on a time-series. More specifically, for each of the 730 ECG signals in the MGDHB, RR series were extracted using WFBD Software Package and the functions *WQRS* and *ANN2RR* [1, 8]. We then generated 20 amplitude adjusted surrogate RR series from each original RR series according to the Octave script

Figure 2. Implementation of the surrogate series test for non-linearity.

```
function y=surrogate(x)
%Step 1: Amplitude transform original
%data to Gaussian distribution
N=length(x);
y=randn(N,1);
y=ampTransform(x,y,N);

%Step 2: Phase randomize series in 1
y=phaseShuffle(y,N);

%Step 3: Amplitude transform #2 to original
y=ampTransform(y,x,N);


function target=ampTransform(source,target,N)
%Steps:
%1. Sort the source
%2. Sort target based on source
%3. Swap source by target
%4. Sort target by increasing time index source
source=[[1:N]' source];
source=sortrows(source,2);
target=[source(:,1) sort(target)];
target=sortrows(target,1);
target=target(:,2);


function y=phaseShuffle(x,N)
%Shuffle spectrum
X=fft(x);
Y=X;
mid=floor(N/2)+ mod(N,2);
phi=2*pi*rand(mid-1,1); %Generate random phase
Y(2:mid)=abs(X(2:mid)).*cos(phi)
         + j*abs(X(2:mid)).*sin(phi);
if(~mod(N,2))
    %Even series, ignore Nyquist
    Y(mid+2:end)=conj(flipud(Y(2:mid)));
    Y(mid+1)=X(mid+1);
else
    %Odd series, no Nyquist
    Y(mid+1:end)=conj(flipud(Y(2:mid)));
end
y=real(ifft(Y));
```

shown below. Statistics computed on such surrogate series versus original series allows us to test the null hypothesis that the original time series was generated by a process with linear dynamics with possibly non-linear, monotonically increasing, measurement function [9, 10]. The statistic used to test for non-linearity was the short term slope of the multi-scale entropy curve using WFDB's MSE function [11].

4. Results

Figures 3 and 4 show the computation time of the example tasks on clusters of different size on Amazon's EC2. For the spectral indexing task, the overhead of starting Hadoop on a distributed mode on a small cluster size was detrimental to performance (increasing the computation time by as much as 5 fold). The spectral indexing task exhibited improved computational time only when faster nodes were used.

The results for the second example show a clear advantage of performing surrogate data testing on a distributed fashion 4. This is not surprising, since this task is very computationally intensive and embarrassingly parallel. Improvement in speed of close to 30 fold was observed a cluster consisting of eight *c1.xlarge* nodes.

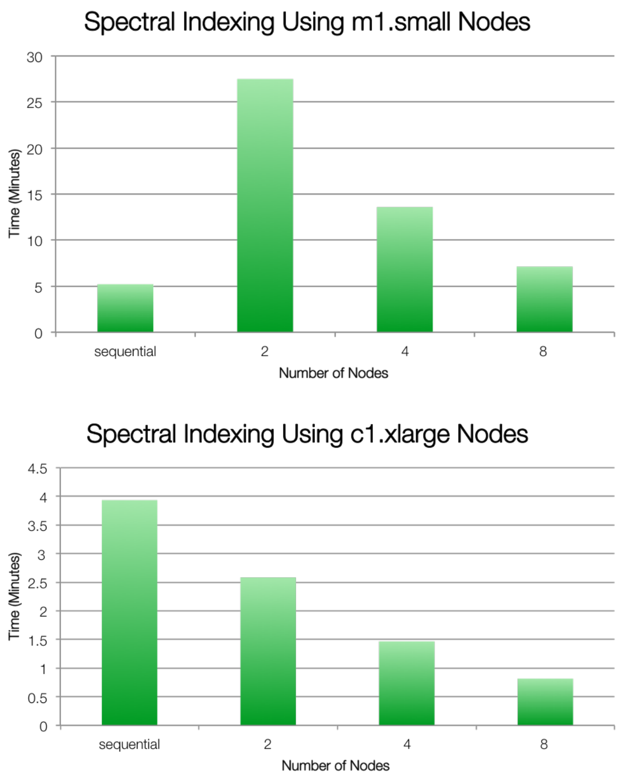

Figure 3. Results of the spectral indexing example in EC2 with clusters of different sizes.

5. Discussion

An unprecedented ampleness of distributed computing environments is now available to researchers, among these: Amazon EC2, Microsoft Windows Azure, Google Compute Engine, and Open Stack. The focus of this paper was to show-case a powerful software-stack that allows researchers to quickly process PhysioNet data in a distributed fashion with minimum configuration overhead. The software stack includes industry standard tools like Hadoop, and the WFDB Software package. The example codes associated with this work will be posted on PhysioNet, along with a tutorial, for the interested researchers.

Our results do not show any benefit of distributed com-

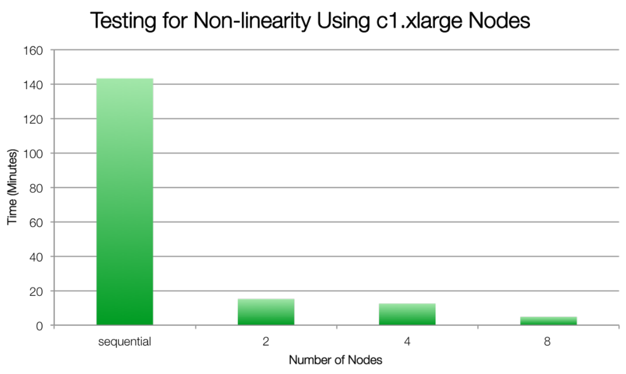

Figure 4. Results of the surrogate testing in EC2 with clusters of different sizes.

puting on small nodes for tasks that are relatively simple and fast to compute on individual records (Example 1). In general, the amount of overhead associated with the management of the distributed environment diminishes any computational gain from the extra nodes. Part of the computational costs may be due to the fact that we used Hadoop exclusively as queuing tool: we did not maximize it's efficiency by processing the data directly in HDFS nor leveraging the "reduce" step. This is a current limitation of our software stack, and requires either a redesign of the WFDB software to operate against single files (rather than "records") or development of a package using Hadoop's Java API to control the low level HDFS data splits of WFDB records.

On the other hand, a large (close to 30 fold) increase in performance was obtained for tasks that are computationally intensive and massively parallel (as in the case of Example 2). Thus tasks that involve intensive computation on individual record files are likely to benefit from the current infrastructure. In the end, we hope that the code and the examples that we make available on PhysioNet will help researchers to quickly decide for themselves if a distributed environment is efficient enough for their particular computational needs.

Acknowledgments

This research was funded in part by the Intel Science and Technology Center for Big Data and the National Library of Medicine Biomedical Informatics Research Training grant (NIH/NLM 2T15 LM007092-22) which supports Tristan Naumann as well as the National Institute of General Medical Sciences grant NIH/NIGMS R01-GM104987 which supports Ikaro Silva. The authors would like to thank Benjamin Moody, Mengling 'Mornin' Feng, and Mohammad Ghassemi for helpful discussions as well as Professors Peter Szolovits and Roger G. Mark, whose labs enable research on improving health care through the application of technology.

References

[1] Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. Circulation 2000;101(23):e215–e220.

[2] Apache Software Foundation. Apache Hadoop, 2014. URL `http://hadoop.apache.org/`.

[3] J. Riley. StarCluster, 2014. URL `http://star.mit.edu/cluster/`.

[4] Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. Communications of the ACM 2008; 51(1):107–113.

[5] Ghemawat S, Gobioff H, Leung ST. The google file system. In ACM SIGOPS Operating Systems Review, volume 37. ACM, 2003; 29–43.

[6] White T. Hadoop: The definitive guide. " O'Reilly Media, Inc.", 2012.

[7] Wang A, et al. An industrial strength audio search algorithm. In ISMIR. 2003; 7–13.

[8] Zong W, Moody G, Jiang D. A robust open-source algorithm to detect onset and duration of qrs complexes. In Computers in Cardiology, 2003. IEEE, 2003; 737–740.

[9] Kantz H, Schreiber T. Nonlinear time series analysis, volume 7. Cambridge university press, 2004.

[10] Kaplan D, Glass L. Understanding nonlinear dynamics, volume 19. Springer, 1995.

[11] Costa M, Goldberger AL, Peng CK. Multiscale entropy analysis of biological signals. Physical Review E 2005; 71(2):021906.

Address for correspondence:

Ikaro Silva
45 Carleton St.
MIT LCP Cambridge, MA 02142
ikaro@mit.edu