Phylogenetic Code in the Cloud – Can it Meet the Expectations?

Adam Kraut[1], Sébastien Moretti[2,3], Marc Robinson-Rechavi[2], Heinz Stockinger[3], and Dean Flanders[4]

1) BioTeam Inc., Middleton, MA, USA
2) University of Lausanne, Department of Ecology and Evolution, Swiss Institute of Bioinformatics, Lausanne, Switzerland
3) Swiss Institute of Bioinformatics, Vital-IT Group, Lausanne, Switzerland
4) Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland

Abstract. *Cloud computing has recently become very popular, and several bioinformatics applications exist already in that domain. The aim of this article is to analyse a current cloud system with respect to usability, benchmark its performance and compare its user friendliness with a conventional cluster job submission system. Given the current hype on the theme, user expectations are rather high, but current results show that neither the price/performance ratio nor the usage model is very satisfactory for large-scale embarrassingly parallel applications. However, for small to medium scale applications that require CPU time at certain peak times the cloud is a suitable alternative.*

Keywords: Cloud computing, bioinformatics, phylogeny, cluster computing

1. Introduction

In the last few years, cloud computing has gained enormous attention in both commercial and academic environments. Based on Amazon's original concept of an "elastic cloud" [1], resource provisioning based on virtualisation has become a very popular solution to solve CPU intensive applications. This is particularly true if certain peak performance demands cannot be met in-house and should be addressed via out-sourcing. Today, several different cloud computing solutions exist, but Amazon's Elastic Compute Cloud (EC2) seems to be the most popular one.

Given that certain applications in the life sciences are very CPU intensive, it is quite natural to consider a cloud solution as a potential "processing engine". Several applications have already been tested in the cloud (e.g. [3], [6], [7]). In our case, we are interested to apply the cloud concept to an evolutionary biology application. In

particular, we address the question of a typical bioinformatician who has the choice to either run the application on a HPC cluster or on a cloud system such as Amazon's EC2. The following questions need to be answered:

- How easy is it to port an existing application to EC2?
- Given prior experience with a cluster job submission system, can a similar interface be used?
- What is the performance of the cloud with respect to a compute cluster?
- What is the actual price?

In the following article we address the questions above using the example of codeml [10], a CPU-intensive phylogenetic application which checks for positive selection in genomic data. This application is embarrassingly parallel and can therefore be used in both a Grid as well as a cloud environment. We assume that the envisaged bioinformatician has experience with a cluster job submission system such as Platform LSF. Performance benchmarks are done on both EC2 and an HPC cluster dedicated to life sciences (http://www.vital-it.ch/). Our results show that current cloud technologies are not yet mature enough, nor do they provide a rich set of features to "easily" take an application that is running on a cluster and port it to a cloud system. There is still considerable work to do. Most importantly, batch-system-like job submission is not yet well addressed by systems such as EC2 that rely on machine images created by end-users.

The remainder of the article is organised as follows: we first provide background on the biological application and its computational challenge. Section 3 gives details about what was necessary to adapt the EC2 environment to give the end user the impression that she uses a cluster environment rather than raw disk images. In the experimental results in Section 4 we provide benchmarks for the cloud and cluster environments using a specific benchmark job and the codeml application. In particular, we compare the cloud environment with an HPC cluster environment with particular focus on run-time performance, usability and price. Finally, we give an outlook on expected costs to run a full-scale phylogenetic application on a regular basis on EC2.

2. Background on Selectome and codeml

The PAML software package [10] is the most widely used software for studying selective pressure on DNA and proteins. It notably includes the branch-site model in the program codeml, which allows a precise detection of Darwinian selection, which might have transiently affected a small part of a gene a very long time ago. This precision has an important computational cost, which is highlighted in updating the database Selectome [8]. Selectome aims to be a comprehensive collection of detected Darwinian selection on animal genes.

2.1. An embarrassingly parallel application

Similar to other bioinformatics and phylogenetic applications, a codeml analysis can be optimised by running several executables in parallel, i.e. it falls into the category

of embarrassingly parallel applications [9]. In fact, in order to compute the data for the Selectome service, a script-based job-submission and execution exists already for LSF on the Vital-IT production cluster. A similar approach can also be taken on a cloud, given that the input data is distributed to several compute nodes which then process the data in parallel.

2.2. The computational challenge

The main computational challenge is imposed by the large dataset that needs to be processed in a rather short amount of time in order to provide a new release of the Selectome database. Selectome is based on phylogenetic families pre-computed by Ensembl [4]. Ideally, a new Selectome release is produced with each new Ensembl release, i.e. every 2 months. In practice, a Selectome release on Vital-IT takes 8 months at present.

Typically, the analysis on a phylogenetic family is composed of several computations: two codeml jobs are required for each internal node of the phylogenetic tree (two hypotheses are tested per node). For current vertebrate phylogenetic families, a Selectome release needs **1'400'000** codeml jobs; a single-threaded codeml job taking on average **20 minutes** on Vital-IT (see Section 4 for CPU details).

At each Ensembl release, the dataset size increases. And the more genomes, the more internal nodes there are, with more computations; and with deeper trees implying longer codeml computations. In addition to this, the Ensembl system is being extended to other groups for which selection information would be relevant, such as bacteria, plants or fungi [5].

3. Preparing a job submission environment on EC2

An Amazon Machine Image (AMI) has been prepared with a self-assembling Sun Grid Engine (SGE) distribution for EC2. As each node is provisioned it queries the Amazon API to configure its role in the cluster. Since EC2 instances have no knowledge of their public hostname at launch, a bootstrap script downloads and runs a configuration recipe to install and configure the necessary software. A typical cluster consists of a head node and many compute nodes. The head node handles network file system services and job distribution while the compute nodes are reserved for the units of work. The job submission system is depicted in Figure 1.

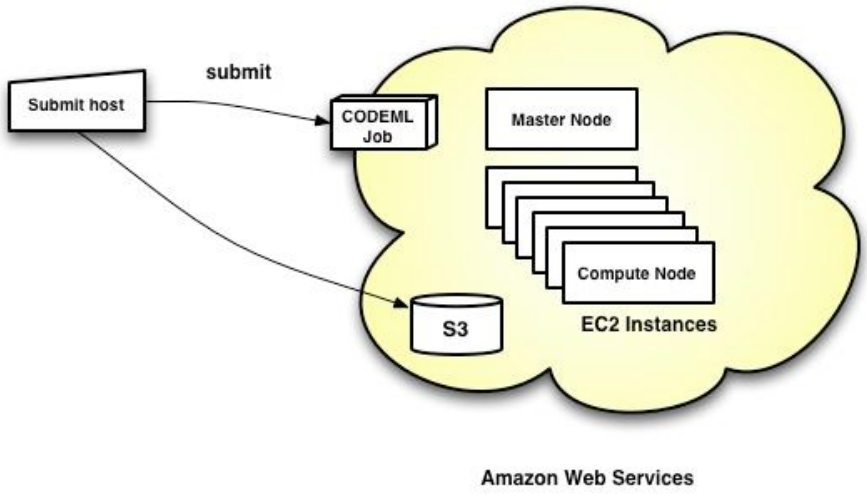

Figure 1. Cloud submission architecture on top of EC2 and SGE.

The Sun Grid Engine distribution for EC2 also provides a shared file system across all of the nodes for a streamlined user environment. This architecture is common in physical computing environments, providing a short learning curve for experienced cluster users. In a few minutes a fully operational cluster is configured, and an end user can interact with the system in a similar way to a "local" cluster, i.e. one can use standard SGE client commands such as qsub and qstat to submit and monitor jobs. A typical interface is given in Figure 2.

Given that the cluster is operational, the user has several options for moving applications as well as the necessary input files to the cloud. For this purpose, standard scp or sftp clients can be used. The user can upload inputs to an S3 bucket and download them to the cluster. Elastic Block Storage (EBS) is available for creating storage volumes containing software or datasets. EBS volumes can be attached to the head node and easily shared with the rest of the cluster. It is also important to note that a shared file system minimises further data movement during the execution of end user applications.

Figure 2. Logging into an EC2 cluster via secure shell

4. Experimental results

The job submission system that was prepared on top of EC2 allows for a direct comparison of job submission using either EC2, or a local resource management system such as LSF. Our analysis has the following main objectives:

- Cloud performance analysis: how long does it take to run a certain experiment in the cloud?
- Performance comparison to local cluster: compare the performance to a production cluster.
- Price: how much does it cost to run a full biological experiment?

A benchmark dataset[1] has been prepared that contains 52 individual and short- to mid-time datasets that need to be processed with codeml. As a performance reference, the entire benchmark has been executed on a 64-bit processor (Intel Xeon 5160, 3 GHz, 2 MB RAM) and lasted 5 hours 56 minutes and 56 seconds. All 52 jobs were launched and executed in serial. On average, an individual job lasted 411 seconds (ranging from 11 to 833 seconds).

1 http://bioinfo.unil.ch/selectome/download/TEST_CASE/codeml_testCase_data.tar.gz

4.1. Cloud benchmark

The EC2 benchmark was performed on 8 EC2 Instances of type 'c1.xlarge'. These instances have Intel Xeon E5410 CPU's with a reported clock speed of 2.33 GHz. This instance type was chosen since it offers the best price to performance ratio. Amazon's high-memory 'm1' class of instances has newer Intel processors but the cost for the large memory instances is too high to justify for these calculations. The cluster is homogeneous, although it is possible to use a smaller instance type for the head node and bigger instances for compute. In this test the head node was also allowed to execute jobs. The EC2 cluster is running Sun Grid Engine resource manager as described above with Gigabit Ethernet interconnects.

Since the EC2 cluster offered 64 available slots, all 52 jobs in the test dataset were started in parallel. The individual runtime for jobs ranged from 10 to 929 seconds (15.48 minutes) with an average of 519 seconds. The total running time for the test dataset was 930 seconds or approximately equal to the length of the longest individual job. The complete cost of running the benchmark was USD 5.44 (8 times USD 0.68 in the zone US – N. Virginia[2]). In this specific case it would be better to run fewer instances so that the overall running time is closer to the nearest pay-hour. Even though the benchmark lasted roughly 15 minutes, Amazon EC2 is billed by the hour. A reasonable estimation of the optimal cluster size could be calculated by (Average Job Time * Number of Jobs) / 60. The scheduling overhead imposed by SGE can be neglected since there are no other jobs in the job queue except the ones we submit here.

4.2. Cluster benchmark

The cluster benchmark was done on Vital-IT's HPC cluster that currently consists of more than 1000 individual processing cores which are connected via a fast Infiniband network connect. The cluster is not fully homogeneous but all nodes have Intel Xeon-based processors (ranging between 2.8 and 3.4 GHz), and they all share the same file system space via a parallel file system. Job submission is done via Platform LSF.

The benchmark dataset was divided into 52 individual jobs that were submitted for execution via LSF. Similar to the single-processor benchmark, the individual jobs lasted between 10 and 833 seconds (13.98 minutes) with an average of 385 seconds. Note that there is a slight difference with respect to the single-core experiment at the beginning of the section due to the heterogeneity of the cluster (CPU speeds range from 2.8 to 3.4 GHz). The overall runtime of the 52 jobs was consequently about 14 minutes. If job scheduling times are added, the total user perceived run-time was 17 minutes. This scheduling time certainly depends on the current load of the cluster. Table 1 summarises the performance results of the cluster and the cloud.

2 http://aws.amazon.com/ec2/#pricing (Feb. 2010)

	Cluster (time in seconds)	Cloud (time in seconds)
Execution time of shortest job	10	10
Execution time of longest job	833	929
Average processing time	385	519
Total run time	833	929

Table 1. Cluster versus cloud – run-time of codeml

4.3. Discussion

The experimental results show that the proposed solution based on Sun Grid Engine on top of EC2 is rather easy to use for end users that have already experience with conventional job submission systems such as LSF or similar (note that LSF and SGE behave in the same way for our application and can easily be compared). Therefore, the learning curve for cloud usage can be kept minimal due to our contribution. As regards the price, we have shown that an embarrassingly parallel application that would last about 6 hours can be done in about 15.5 minutes at a price of less than USD 6. Given that Amazon always charges full hours, the actual problem size could have been four times bigger than the one we used, keeping the same price level.

An interesting question might also be: how much would it cost if the same performance requirements would need to be met for an entire year, i.e. 64 processing cores for 365 days. According to the current price range in EC2, one 8-core CPU is USD 1'820, i.e. the full 64 cores would cost USD 14'560 in both USA and Europe. If we further consider data transfer costs, the numbers are slightly higher (e.g. USD 256 for 1 TB/month outbound. Inbound data transfer will be free starting in June 2010.)

Note that our proposed set-up does not scale smoothly beyond a few hundred nodes due to intrinsic scalability and performance limitation of NFS. However, since the problem is embarrassingly parallel, one can create several smaller SGE clusters (in the order of 64 to 128 hosts) and adapt the job submission to such an environment.

5. Conclusion

We have successfully ported a CPU intensive, embarrassingly parallel life science application to a cloud environment and analysed it for both usability and performance. It should also be noted that using the EC2 environment is not straight forward as one needs to create and deploy machine images – a considerable "burden" for certain users. Another important observation was that a popular cloud environment such as EC2 does not provide a high-level job submission system with a conventional and easy-to-use interface. It was not a show-stopper in our case since BioTeam had the expertise to create such an environment. However, the currently proposed solution has intrinsic scalability limitations (NFS does not scale (well) to hundreds or thousands of nodes with simultaneous reads and writes) – a possible solution to this problem is to use

several SGE/EC2 clusters independently. An alternative solution might be Hadoop's MapReduce implementation but this requires adapting to a new programming paradigm. Finally, LSF seems to provide a new solution [2] on top of EC2 that looks more promising but needs thorough testing.

We have demonstrated that our proposal provides an easy-to-use solution to an embarrassingly parallel problem in life sciences. Overall, a phylogenetic problem that would last about 24 hours can be calculated on 8 hosts each offering 8 processing cores (i.e. 64 processing cores in total) for less than USD 6. That might be a solution to deal with peak performance needs of small life science groups that do not have immediate access to powerful computational clusters.

In contrast, a complete Selectome release is faced with a trade-off between fully parallel and fully serial computations. A fully serial approach looks unrealistic due to time and money required. A fully parallel approach looks promising but requires allocating too many CPUs, and money, at the same time for both cluster and cloud solutions. A hybrid solution would be to choose the time required for a release and to allocate enough CPUs to be on time. Following Ensembl releases, 1'400'000 codeml jobs should to be done in 2 months (60 days). For the whole dataset, the mean time for a job is 20 minutes. Thus, about 325 cores could produce a Selectome release in 2 months. If we consider failures and other technical problems, 344 (43 EC2 compute units) cores would be a right basis, and computations should cost USD 42'105.6. On the cluster side, if one considers buying 90 quad-core processor, commodity-off-the-shelf machines, one is in the same price range, neglecting networking, cooling and maintenance (incl. costs of a system administrator). These figures show again that peak demands could potentially be served cost-effectively via EC2, but regular processing requirements such as four to six Selectome releases per year for a longer period are more cost-efficiently done in-house on a dedicated cluster.

Acknowledgements

This work was in part funded via the SystemsX.ch initiative in Switzerland. The cluster computations were performed at the Vital-IT (http://www.vital-it.ch) Center for high-performance computing of the Swiss Institute of Bioinformatics.

References

[1] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/, Jan. 2010.

[2] M. Feldman. Platform Launches Private and Hybrid Cloud Management Products for HPC. HPCwire, Nov. 2009. http://www.hpcwire.com/features/Platform-Launches-Private-and-Hybrid-Cloud-Management-Products-for-HPC-69295592.html

[3] B. Halligan, J. Geiger, A. Vallejos, A. Greene, S. Twigger. Low Cost, Scalable Proteomics Data Analysis Using Amazon's Cloud Computing Services and Open Source Search Algorithms. *Journal of Proteome Research*, 8(6):3148–3153, 2009.

[4] T. Hubbard, et al. Ensembl 2009. *Nucleic Acids Res*., 37(Database issue):D690-7, 2009.

[5] P. Kersey, et al. Ensembl Genomes: extending Ensembl across the taxonomic space. *Nucleic Acids Res*., 38(Database issue):D563-9, 2010.

[6] A. Kraut. Antibody Docking on the Amazon Cloud. BioIT – World, May 19, 2009. http://www.bio-itworld.com/issues/2009/may-jun/antibody-docking-EC2.html

[7] B. Langmead, M. Schatz, J. Lin, M. Pop, S. Salzberg. Searching for SNPs with cloud computing. *Genome Biology*, 10**:**R134, 2009.

[8] E. Proux, R. Studer, S. Moretti, M. Robinson-Rechavi. Selectome: a database of positive selection. *Nucleic Acids Res*., 37(Database issue):D404-D407, 2009.

[9] H. Stockinger, M. Pagni, L. Cerutti, L. Falquet. Grid Approach to Embarrassingly Parallel CPU-Intensive Bioinformatics Problems. *2nd IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*, IEEE Computer Society Press, Amsterdam, The Netherlands, Dec. 4-6, 2006.

[10] Z. Yang. PAML 4: Phylogenetic Analysis by Maximum Likelihood. *Molecular Biology and Evolution*, 24:1586-1591, 2007.