Bringing Elastic MapReduce to Scientific Clouds

Pierre Riteau, Kate Keahey, Christine Morin

Bringing Elastic MapReduce to Scientific Clouds

Pierre Riteau
Université de Rennes 1, IRISA
INRIA, Centre Rennes - Bretagne Atlantique
Rennes, France
Pierre.Riteau@irisa.fr

Kate Keahey
Computation Institute, University of Chicago
Argonne National Laboratory
Chicago, IL, USA
keahey@mcs.anl.gov

Christine Morin
INRIA, Centre Rennes - Bretagne Atlantique
Rennes, France
Christine.Morin@inria.fr

ABSTRACT
The MapReduce programming model, proposed by Google, offers a simple and efficient way to perform distributed computation over large data sets. The Apache Hadoop framework is a free and open-source implementation of MapReduce. To simplify the usage of Hadoop, Amazon Web Services provides Elastic MapReduce, a web service that enables users to submit MapReduce jobs. Elastic MapReduce takes care of resource provisioning, Hadoop configuration and performance tuning, data staging, fault tolerance, etc. This service drastically reduces the entry barrier to perform MapReduce computations in the cloud. However, Elastic MapReduce is limited to using Amazon EC2 resources, and requires an extra fee. In this paper, we present our work towards creating an implementation of Elastic MapReduce which is able to use resources from other clouds than Amazon EC2, such as scientific clouds. This work will also serve as a foundation for more advanced experiments, such as performing MapReduce computations over multiple distributed clouds.

Keywords
MapReduce, Cloud Computing, Elastic MapReduce, Hadoop

1. INTRODUCTION
The MapReduce programming model [5] proposed by Google offers a simple way to perform distributed computation over large data sets. Users provide a map and a reduce function. The map function takes a set of input key/value pairs, and produces intermediate key/value pairs. The reduce function merges intermediate key/value pairs together to produce the result of the computation. This programming model became popular because it is simple yet expressive enough to perform a large variety of computing tasks, from data mining to scientific computations. This programming model is backed by a large framework that takes care of scheduling tasks to workers, sharing data through a distributed file system, handling faults, etc.

The Apache Hadoop [2] project develops a free and open-source implementation of the MapReduce framework. This framework is heavily used by companies such as Yahoo! and Facebook to perform thousands of computations per day over petabytes of data [9]. However, managing a Hadoop cluster can require some expertise, especially when scaling to a large number of machines. Moreover, users wanting to perform MapReduce computations in cloud computing environments need to instantiate and manage virtual resources, which further complicates the process.

To lower the entry barrier for performing MapReduce computations in the cloud, Amazon Web Services provides Elastic MapReduce [1]. Elastic MapReduce is a web service to which users submit MapReduce jobs. The service takes care of provisioning resources, configuring and tuning Hadoop, staging data, monitoring job execution, instantiating new virtual machines in case of failure, etc.

However, this service has a number of limitations. First, it is restricted to Amazon EC2 resources. Users are not able to use Elastic MapReduce with resources from other public clouds or from private clouds, which might be less expensive. This is especially true for scientists who can have access to scientific clouds [7]. Moreover, Elastic MapReduce is provided for an hourly fee, in addition to the cost of EC2 resources. This fee ranges from 17% to 21% of the EC2 resource cost. Second, users are not able to use all types of instances provided by Amazon. For instance, it is not possible to use spot instances, even though this could lead to a substantial cost reduction [4]. It is also impossible to use a different virtual machine image than the one provided by Amazon, which is based on Debian 5.0.

Providing an implementation of Elastic MapReduce without these limitations would make it much easier for scientists to take advantage of resources from scientific clouds. Moreover, it can serve as a foundation for MapReduce research, such as evaluating MapReduce computations over multiple clouds. In this paper, we present our ongoing work towards creating this implementation.

2. AMAZON ELASTIC MAPREDUCE
Amazon Elastic MapReduce is part of the Amazon Web Services infrastructure. After signing up (and providing a credit card number), users can submit MapReduce jobs through the AWS management console (a web interface), through a command line tool, or by directly calling the web service API

(libraries to access the Elastic MapReduce API are available for programming languages such as Java and Python).

Users interact with Elastic MapReduce by submitting job flows. Before executing a job flow, Elastic MapReduce provisions a Hadoop cluster using Amazon EC2 resources. Then, the service executes bootstrap actions, which are scripts specified by users. These actions allow users to provide some level of customization of the Hadoop appliance. A job flow contains a sequence of steps, which are executed in order. Internally, each step corresponds to a Hadoop job. After all steps are executed, the cluster is shut down. Users can ask for the cluster to be kept alive, which can be useful for debugging job flows, or submitting multiple job flows without waiting for cluster provisioning and paying extra usage (instance cost is rounded up to the hour).

Typically, input data is fetched from Amazon S3. Intermediate data is stored in HDFS, the Hadoop Distributed File System. Output data is saved in Amazon S3, since the HDFS file system is destroyed when the Hadoop cluster is terminated.

3. DESIGN AND IMPLEMENTATION

Our Elastic MapReduce prototype, which is work in progress, currently implements a subset of the Amazon Elastic MapReduce service. Users are able to submit job flows and query job flow statuses. The corresponding API calls are RunJobFlow and DescribeJobFlows. Job flows written using SQL-like languages (Hive and Pig) are not supported yet.

When a new job flow request is received and after its parameters are validated, the service contacts a cloud on behalf of the user to provision a Hadoop cluster. A Hadoop cluster is composed of a master node and multiple slave nodes. The master node runs the NameNode service (the meta data server of the HDFS file system) and the JobTracker service (which receives and schedules jobs to slave nodes). Slave nodes run the DataNode service (which provides data storage to HDFS) and the TaskTracker service (which executes map and reduce computations). By leveraging the EC2 API to provision Hadoop clusters, our implementation is able to get resources from most clouds managed by open-source toolkits (Nimbus, OpenNebula, Eucalyptus).

Once an Hadoop cluster is provisioned and configured, each step in the job flow is executed as a Hadoop job. Steps specify input data and output locations, usually as S3 URLs. Hadoop supports getting and storing data directly from Amazon S3. This means it is possible to provide our Hadoop clusters with input data from Amazon S3, like in Amazon Elastic MapReduce. However, since traffic from Amazon S3 to the outside world is charged to users, this would incur a high cost. Also, bandwidth constraints between Amazon S3 and the cloud running the Hadoop cluster would reduce performance. We assume that clouds used with our Elastic MapReduce implementation will typically have a storage service available. For instance, both Nimbus [3] and Eucalyptus [8] provide S3 implementations (respectively called Cumulus and Walrus). Currently, our system fetches input data from a local repository using Boto, a Python library supporting the S3 API, and stores it in HDFS. Similarly, output data is stored into the local repository after a step is finished. We are currently investigating how we can configure or modify Hadoop to use an alternate S3 repository.

For validating our prototype, we execute job flows on resources provisioned from the Nimbus [3] cloud running of the University of Chicago FutureGrid cluster.

4. RELATED WORK

Gunarathne et al. [6] proposed a MapReduce implementation built on top of Microsoft Azure cloud services. Their work advocates for using MapReduce systems on other clouds than Amazon EC2. Chohan et al. [4] propose using spot instances to execute MapReduce jobs. Spot instances are Amazon EC2 instances which have a variable cost, depending on infrastructure load, and can be killed at any time. We plan to study spot instance usage in our implementation.

5. CONCLUSION AND FUTURE WORKS

In this paper, we presented our ongoing work towards creating an Elastic MapReduce implementation that allows users to take advantage of resources other than Amazon EC2 instances, such as resources from scientific clouds. Our system takes care of provisioning Hadoop clusters and submitting jobs, allowing users to focus on writing their MapReduce application rather than managing cloud resources. In the near future, we will evaluate our implementation compared to Amazon Elastic MapReduce, for example by analyzing cluster instantiation time, performance, and cost. We also plan to use our system to experiment with MapReduce jobs over multiple distributed clouds.

6. REFERENCES

[1] Amazon Elastic MapReduce. http://aws.amazon.com/elasticmapreduce/.

[2] Apache Hadoop. http://hadoop.apache.org/.

[3] Nimbus. http://www.nimbusproject.org/.

[4] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz. See spot run: using spot instances for MapReduce workflows. *HotCloud'10: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010.

[5] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[6] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox. MapReduce in the Clouds for Science. In *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 2010.

[7] K. Keahey and T. Freeman. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. In *Cloud Computing and Its Applications 2008 (CCA-08)*, 2008.

[8] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System. 2009.

[9] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sarma, R. Murthy, and H. Liu. Data warehousing and analytics infrastructure at Facebook. *SIGMOD '10: Proceedings of the 2010 International Conference on Management of Data*, 2010.