On the Varieties of Clouds for Data Intensive Computing

Robert L. Grossman
University of Illinois at Chicago
and Open Data Group

Yunhong Gu
University of Illinois at Chicago

Abstract

By a cloud *we mean an infrastructure that provides resources or services over a network, often the Internet, usually at the scale and with the reliability of a data center. We distinguish between clouds that provide on-demand computing instances (such as Amazon's EC2 service) and clouds that provide on-demand computing capacity (such as provided by Hadoop). We give a quick overview of clouds and then describe some open source clouds that provide on-demand computing capacity. We conclude with some research questions.*

1 Introduction

1.1 Types of Clouds

There is not yet a standard definition for cloud computing, but a good working definition is to say that *clouds* provide on demand resources or services over a network, often the Internet, usually at the scale and with the reliability of a data center.

There are quite a few different types of clouds and again there is no standard way of characterizing the different types of clouds. One way to distinguish different types of clouds is to categorize the architecture model, computing model, management model and payment model. We discuss each of these below. See Table 1.

1.2 Architectural Model

We begin with the architecture model. There are at least two different, but related, architectures for clouds: the first architecture is designed to provide computing *instances* on demand, while the second architecture is designed to provide computing *capacity* on demand.

Amazon's EC2 services [1] provides computing instances on demand and is an example of the first architectural model. A small EC2 computing instance costs $0.10 per hour and provides the approximate computing power of 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor, with 1.7 GB memory, 160 GB of available disk space and moderate I/O performance [1].

Google's MapReduce provides computing capacity on demand and is an example of the second architectural model for clouds. MapReduce was introduced by Google in the paper [8]. This paper describes a sorting application that was run on a cluster containing approximately 1800 machines. Each machine had two 2 GHz Intel Xeon processors, 4 GB memory, and two 160 GB IDE disks. The TeraSort benchmark [10] was coded

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Model	Variants
Architecture Model	clouds that provide on-demand computing instances; clouds that provide on-demand computing capacity
Programming Model	Using queues to pass message; MapReduce over storage clouds; UDFs over storage clouds; message passing
Management Model	private vs shared; internal vs hosted
Payment Model	pay as you go; subscribe for a specified period of time; buy

Table 1: Some different types of clouds.

using MapReduce, a parallel programming model which is described in more detail below. The goal of the TeraSort benchmark is to sort 10^{10} 100-byte records, which is about 1 TB of data. The application required about 891 seconds to complete [8] on this cluster.

The Eucalyptus system [19] is an open source cloud that provides on demand computing instances and shares the same APIs as Amazon's EC2 cloud. The Hadoop system is an open source cloud that implements a version of MapReduce [16].

Notice that both types of clouds consist of loosely coupled commodity computers, but that the first architecture is designed to scale out by providing additional computing instances, while the second architecture is designed to support data or compute intensive applications by scaling computing capacity. By scaling computing capacity, we mean the ability to aggregate a large number of loosely coupled computers so that the aggregate infrastructure can manage very large datasets, sustain very large aggregate input/output, and perform a very large aggregate number of computing tasks.

1.3 Programming Model

Clouds that provide on-demand computing instances can support any computing model compatible with loosely coupled clusters. For example, instances in an Amazon EC2 can communicate using web services [3], using queues [2], or using message passing. It is important to note though that the performance using message passing on loosely coupled systems is much slower than message passing or tightly coupled clusters.

Clouds that provide on-demand computing capacity can also support any computing model compatible with loosely coupled clusters. Programming using web services and message passing can be complicated though and beginning with [8], a programming model called MapReduce has become the dominant programming model used in clouds that provide on-demand computing capacity. MapReduce assume that many common programming applications can be coded as processes that manipulate large datasets consisting of <key, value> pairs. Map is a process that maps each <key, value> pair in the dataset into a new pair of <key′, value′>. Reduce is a process that merges values with the same key. Although this is a seemingly simple model, it has been used to support a large number of data intensive applications, especially applications that must manipulate web related data. MapReduce is described in more detail in Section 2.1 below.

Stream-based parallel programming models in which a User Defined Function (UDF) is applied to all the data managed by the cloud have also proved to be quite useful [14].

1.4 Payment Model

Amazon popularized a cloud that provides on-demand computing instances with a "pay as you go" economic model. By simply setting up a Amazon Web Services account that links to a credit card, one can set up a computing instance, with attached storage and network connectivity and pay about 10 cents an hour for just those hours that you actually use the resources.

Of course, you can also buy, set up, and run your own cloud. Alternately, you can make arrangements with a third party to pay for the exclusive use of cloud resources for a specified period of time.

1.5 Management Model

The hardware for clouds can be by provided internally by an organization (internal clouds) or externally by a third party (hosted clouds). A cloud may be restricted to a single organization or group (private clouds) or shared by multiple groups or organizations (shared clouds). All combinations of these management options arise.

1.6 What's New?

Local and remote loosely coupled clusters have been available for quite some time and there is a large amount of middleware available for such clusters. Because of this, it is important to ask what is new with clouds.

The first thing that is new is the scale. Google and Yahoo have reported computing on clouds that contain 1000, 2000 and up to 10,000 loosely coupled computers. With Hadoop, datasets that are tens to hundreds of terabytes can be managed easily, something that requires significant effort with a database.

The second thing that is new is the simplicity that clouds provide. For example, with just a credit card and a browser connected to the Internet, you can use Amazon's EC2, S3, and SQS to bring up 100 computing instances, perform a computation, and return the results without any capital investment, without hiring a system administrator, and without installing and mastering any complex middleware. Useful machine images containing precisely the pre-installed software required can be invoked by simply referencing an Amazon Machine Image identifier, such as ami-3c47a355.

As another example, with MapReduce, a new software engineer can be analyzing a 10 TB dataset of web data on 100 nodes with less than a day of instruction by using simple, small MapReduce programs.

It is interesting to note that this style of cloud computing came from industry's need for a simple to use, yet powerful platform for high performance computing, not from academic research in high performance computing.

2 Clouds That Provide On-Demand Computing Capacity

2.1 Google's Storage, Compute and Table Cloud Services

The basic architecture for clouds that provide on-demand computing capacity was articulated in a series of Google technical reports. See Figure 1. A cloud storage service called the Google File System (GFS) was described in [9]. GFS was designed to scale to clusters containing thousands of nodes and was optimized for appending and for reading data.

For computing with data managed by GFS, a parallel programming framework for loosely coupled systems called MapReduce was described in [8]. A good way to describe MapReduce is through an example: Assume that node i in a cloud stores web pages $p_{i,1}$, $p_{i,2}$, $p_{i,3}$, ..., $p_{i,n}$. Assume also that web page p_i contains words w_1, w_2, w_{i_j}, ... A basic structure important in information retrieval is an inverted index, which is a data structure consisting of a word followed by a list of web pages

$$(w_1; p_{1,1}, p_{2,1}, p_{3,2}, \ldots)$$

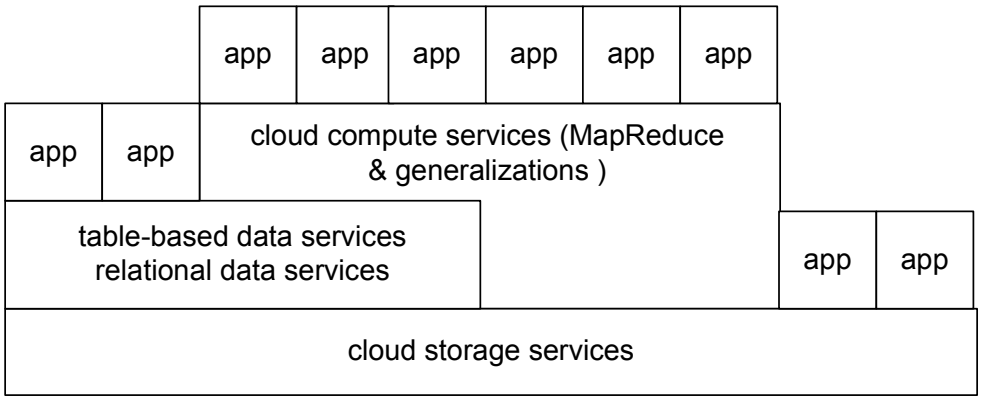

Figure 1: Clouds that provide on-demand computing capacity often layer services as shown in the diagram.

$$(w_2; p_{1,1}, p_{1,2}, p_{3,1}, \ldots)$$
$$(w_3; p_{1,3}, p_{2,2}, p_{3,3}, \ldots)$$

with the properties:

1. The inverted index is sorted by the word w_j;
2. If a word w_j occurs in a web page p_i, then the web page p_i is on the list associated with the word w_j.

A mapping function processes each web page independently, on its local storage node, providing data parallelism. The mapping function emits multiple <key, value> pairs (<keyword, page_id> in this example) as the outputs. This is called the Map Phase.

A partition function $m(w)$, which given a word w, assigns a machine labeled with $m(w)$, is then used to send the outputs to multiple common locations for further processing. This second step is usually called the Shuffle Phase.

In the third step, the processor $m(w_i)$ sorts all the <key, value> pairs according to the key. (Note that there may be multiple keys sent to the same node, i.e., $m(w_i) = m(w_j)$.) Pairs with same key (keyword in this example) are then merged together to generate a portion of the inverted index $<w_i$: $p_{x,y}$, $\ldots)>$. This is called the Reduce Phase.

To use MapReduce, a programmer simply defines the (input) Record Reader (for parsing), Map, Partition, Sort (or Comparison), and Reduce functions and the infrastructure takes care of the rest.

Since many applications need access to rows and columns of data (not just bytes of data provided by the GFS), a GFS-application called BigTable [5] that provides data services that scale to thousands of nodes was developed. BigTable is optimized for appending data and for reading data. Instead of the ACID requirements of traditional databases, BigTable choose an eventual consistency model.

2.2 Open Source Clouds That Provide On-Demand Computing Capacity

The Google's GFS, MapReduce and BigTable are proprietary and not generally available. Hadoop [16] is an Apache open source cloud that provides on-demand computing capacity and that generally follows the design described in the technical reports [9] and [8]. There is also an open source application called HBase that runs over Hadoop and generally follows the BigTable design described in [8].

Sector is another open source system that provides on-demand computing capacity [18]. Sector was not developed following the design described in the Google technical reports, but instead was designed to manage and distribute large scientific datasets, especially over wide area high performance networks. One of the first

Design Decision	Google's GFS, MapReduce, BigTable	Hadoop	Sector
data management	block-based file system	block-based file system	data partitioned into files; native file system used
communication	TCP	TCP	UDP-Based Data Transport (UDT) and SSL
programming model	MapReduce	MapReduce	User defined functions, MapReduce
replication strategy	at the time of writing	at the time of writing	periodically
security	not mentioned	yes	yes (HIPAA capable)
language	C++	Java	C++

Table 2: Some of the similarities and differences between Google's GFS and MapReduce, Hadoop and Sector.

Sector applications was the distribution of the 10+ TB Sloan Digital Sky Survey [15]. Sector is based upon a network protocol called UDT that is designed to be fair and friendly to other flows (including TCP flows), but to use all the otherwise available bandwidth in wide area high performance network [13].

The Hadoop Distributed File System (HDFS), like Google's GFS, implements a block-based distributed file system, which is a fairly complex undertaking. HDFS splits files to into large data blocks (usually 64MB each) and replicates each block on several nodes (the default is to use three replicas). In contrast, Sector assumes that the user has split a dataset into several files, with the size and number of files depending upon the number of nodes available, the size of the dataset, and the anticipated access patterns. Although this imposes a small burden on the user, the result is a much simpler design can be used for the underlying system.

On top of the Sector Distributed File System is a parallel programming framework that can invoke user defined functions (UDFs) over the data managed by Sector. Three specific, but very important UDFs, are the Map, Shuffle and Reduce UDFs described above, which are available in Sector.

Table 2 contains a summary of some of these similarities and differences.

2.3 Experimental Studies

In this section, we describe some experimental studies comparing the performance of Sector and Hadoop. The experiments were performed on the Open Cloud Testbed, a testbed managed by the Open Cloud Consortium [17]. The Open Cloud Testbed consists of four geographically distributed racks located in Chicago (two locations), San Diego and Baltimore and connected by 10 Gb/s networks. Each contains 30 Dell 1435 computers with 4GB memory, 1TB disk, 2.0GHz dual-core AMD Opteron 2212, with 1 Gb/s network interface cards. Since the tests were done, the current equipment in the Open Cloud Testbed has been upgraded and and additional sites have been added.

Table 3 contains some experimental studies comparing Sector and Hadoop using the Terasort benchmark [10]. The tests placed 10GB of data on each node. The tests were run on a single rack, two racks connected by a Metropolitan Area Network in Chicago, three racks connected by a Wide Area Network, and four racks connected by a Wide Area Network. In all cases, the networks provided 10 Gb/s of bandwidth. Notice that although there is a penality incurred for the computing across geographhically distributed racks, it is not prohibitive. It is about 20% when using Sector and about 64% when using Hadoop, when wide area high performance networks are available.

Table 4 contains some experimental studies that were done using CreditStone [4], which is a benchmark that can be used for testing clouds that provide on-demand computing capacity. CreditStone provides code that generates synthetic events that are roughly modeled on credit card transactions and flags some of the transactions.

	Number of nodes	Sector	Hadoop
WAN-2 (UIC, SL, UCSD, JHU)	118	3702 sec	1526 sec
WAN-1 (UIC, SL, UCSD)	88	3069 sec	1430 sec
MAN (UIC, SL)	58	2617 sec	1301 sec
LAN (UIC)	29	2252 sec	1265 sec

Table 3: The table shows the time required to complete the Terasort benchmark. The tests were run on the Open Cloud Testbed. The time required to generate the data is excluded. The test used 10 GB of data per node. The four racks on the testbed were connected by a 10 Gb/s network.

# Locations	Sector	Hadoop	# Events
1 location, 30 nodes, LAN	36 min	126 min	15 billion
4 locations, 117 nodes, WAN	71 min	189 min	58.5 billion

Table 4: Some experimental studies using the CreditStone benchmark comparing Hadoop and Sector run on the Open Cloud Testbed. Hadoop was configured to use one replica for these experiments.

The benchmark requires that certain ratios of unflagged to flagged transactions be computed, a computation that is quite straightforward to do using MapReduce, UDFs, or similar programming models.

3 Research Questions

In this section, we discuss several research questions.

1. In Section 2, we discussed two parallel programming models for clouds that provide on-demand computing capacity (MapReduce and invoking UDFs on dataset segments managed by a storage cloud), both of which are more limited than parallel programming using message passing but which most programmers find easier to use. A research question is to investigate other parallel programming models for these types of clouds that cover a different class of applications but are also quite easy to use.

2. Most clouds today are designed to do the computation within one data center. A interesting research question is to develop appropriate network protocols, architectures and middleware for wide area clouds that span multiple data centers.

3. Another research question is to investigate how different clouds can interoperate; that is, how two different clouds, perhaps managed by two different organizations, can share information.

4. A practical question is to develop standards and standards based architectures for cloud services for clouds that provide on-demand computing capacity so, for example, alternate storage, compute, or table services could be used in a cloud application.

References

[1] Amazon. Amazon Elastic Compute Cloud (amazon ec2). ams.amazon.com/ec2, 2008.

[2] Amazon. Amazon Seb Services Queue Service. Retrieved from http://aws.amazon.com/sqs, 2008.

[3] Amazon. Amazon Web Services Developer Connection. Retrieved from http://aws.amazon.com, 2008.

[4] Collin Bennett, Robert L Grossman, Jonathan Seidman, and Steve Vejcik. Creditstone: A benchmark for clouds that provide on-demand capacity. to appear, 2008.

[5] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. In *OSDI'06: Seventh Symposium on Operating System Design and Implementation*, 2006.

[6] Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y. Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. In *NIPS*, volume 19, 2007.

[7] Data Mining Group. Predictive Model Markup Language (pmml), version 3.2. http://www.dmg.org, 2008.

[8] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.

[9] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, New York, NY, USA, 2003. ACM.

[10] Jim Gray. Sort benchmark home page. http://research.microsoft.com/barc/SortBenchmark/, 2008.

[11] Robert L Grossman and Yunhong Gu. Data mining using high performance clouds: Experimental studies using sector and sphere. In *Proceedings of The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*. ACM, 2008.

[12] Robert L Grossman, Mark Hornick, and Gregor Mayer. Data mining standards initiatives. *Communications of the ACM*, 45(8):59–61, 2002.

[13] Yunhong Gu and Robert L Grossman. UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks*, 51(7):1777—1799, 2007.

[14] Yunhong Gu and Robert L Grossman. Sector and sphere: Towards simplified storage and processing of large scale distributed data. *Philosophical Transactions of the Royal Society A, also arxiv:0809.1181*, 2009.

[15] Yunhong Gu, Robert L Grossman, Alex Szalay, and Ani Thakar. Distributing the sloan digital sky survey using udt and sector. In *Proceedings of e-Science 2006*, 2006.

[16] Hadoop. Welcome to Hadoop! hadoop.apache.org/core/, 2008.

[17] Open Cloud Consortium. http://www.opencloudconsortium.org, 2009.

[18] Sector. http://sector.sourceforge.net, 2008.

[19] Rich Wolski, Chris Grzegorczyk, and Dan Nurmi et. al. Eucalyptus. retrieved from http://eucalyptus.cs.ucsb.edu/, 2008.