Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 53 (2012) 12059-12064

ISSN: 2229-712X

A Comparison of Amazon Elastic Mapreduce and Azure Mapreduce

Bahman Rashidi[1], Esmail Asyabi[1] and Talie Jafari[2]
[1]Iran University of Science and Technology (IUST).
[2]Amirkabir University of Technology.

ARTICLE INFO

Article history:
Received: 18 October 2012;
Received in revised form:
7 December 2012;
Accepted: 14 December 2012;

Keywords
Cloud computing.
Mapreduce.
Cloud mapreduce.
Azure mapreduce.
Amazon elastic mapreduce.

ABSTRACT

In last two decades continues increase of comput-ational power and recent advance in the web technology cause to provide large amounts of data. That needs large scale data processing mechanism to handle this volume of data. MapReduce is a programming model for large scale distributed data processing in an efficient and transparent way. Due to its excellent fault tolerance features, scalability and the ease of use. Currently, there are several options for using MapReduce in cloud environments, such as using MapReduce as a service, setting up one's own MapReduce cluster on cloud instances, or using specialized cloud MapReduce runtimes that take advantage of cloud infrastructure services. Cloud computing has recently emerged as a new paradigm that provide computing infrastructure and large scale data processing mechanism in the network. The cloud is on demand, scalable and high availability so implement of MapReduce on the top of cloud services cause faster, scalable and high available MapReduce framework for large scale data processing. In this paper we explain how to implement MapReduce in the cloud and also have a comparison between implementations of MapReduce on AzureCloud, Amazon Cloud and Hadoop at the end.

Introduction

The amount of data in our world has been exploding, and analysing large data sets called big data, will become a key basis of many researches. Data is being collected and stored at unprecedented rates. The challenge is not only to store and manage the vast volume of data ("big data"), but also to analyse and extract meaningful value from it. There are several approaches to collecting, storing, processing, and analysing big data. MapReduce is one of existing mechanisms for big data processing.

MapReduce is a distributed programming framework designed to ease the development of scalable data-intensive applications for large clusters of commodity machines. The MapReduce distributed data analysis framework model introduced by Google provides an easy-to-use programming model that features fault tolerance, automatic parallelization, scalability and data locality-based optimizations. Due to their excellent fault tolerance features, MapReduce frameworks are well-suited for the execution of large distributed jobs in brittle environments such as commodity clusters and cloud infrastructures [5][12].

Hadoop MapReduce provides a mechanism for programmers to leverage the distributed systems for processing data sets. MapReduce can be divided into two distinct phases:

• Map Phase: Divides the workload into smaller sub workloads and assigns tasks to Mapper, which processes each unit block of data. The output of Mapper is a sorted list of (key, value) pairs. This list is passed (also called shuffling) to the next phase.

• Reduce: analyses and merges the input to produce the final output. The final output is written to the HDFS in the cluster.

Cloud computing is a new paradigm for the provision of computing infrastructure. This paradigm shifts the location of this infrastructure to the network to reduce the costs associated with the management of hardware and software resources. Hence, businesses and users become able to access application services from anywhere in the world [11].

Characteristics of cloud services like On-demand self-service, Broad network access, Resource pooling, Rapid elasticity and Measured Service cause MapReduce take advantage of cloud infrastructure services and probably cloud is a good platform for implementation of MapReduce [3] [11].

In this paper we bring out a complete comparison of the two different implementations of MapReduce programming model that implemented on top of cloud computing. The rest of the paper is organized as follows. The cloud computing and cloud service models are briefly explained. Also the MapReduce and his architecture are briefly explained and the characteristics of MapReduce implementation in the cloud environment. At last discusses and compares two models of cloud MapReduce. Concluding remarks are presented.

Cloud Computing

The concept of cloud computing addresses the next evolutionary step distributed computing. The goal of this computing model is to make a better use of distributed resources, put them together in order to achieve higher throughput and be able to tackle large scale computation problems. Cloud computing is not a completely new concept for the development and operation of web application. It allows for the most cost-effective development of scalable web portals on highly available and fail-safe infrastructure [1].

Cloud computing deals with virtualization, scalability, interoperability, quality of service and the delivery models of the cloud, namely private, public and hybrid.

A more structured definition is given by Buyya et al [2]: who define a Cloud as a "type of parallel and distributed system consisting of a collection of interconnected and

Tele:
E-mail addresses: b_rashidi@comp.iust.ac.ir

virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreement".

The US National Institute of Standards and Technology (NIST) offer a comprehensive and general definitions for cloud computing, aspects and Characteristics. It summarizes cloud computing as [3]:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

NIST also defines the characteristics of cloud computing, these characteristics are defined as follows:

- On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- Resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.
- Rapid elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

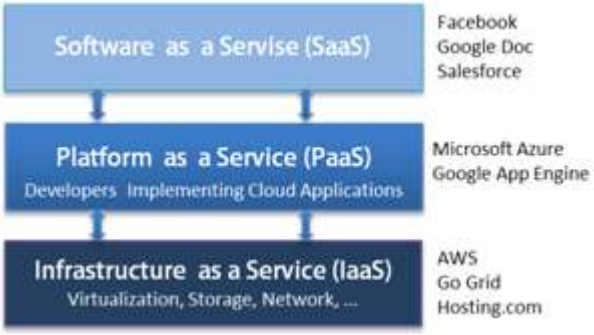

Fig 1. Models of Cloud Services

- Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

The NIST definition also the wide variety of services exposed by the Cloud computing can be classified and organized into three classes (Figure 1) which are referred to as a services models:

Software as a Service (SaaS): The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

Platform as a Service (PaaS): The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

Infrastructure as a Service (IaaS): The capability provided to the consumer to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

- NIST defined primarily four cloud deployments according to who the owner of the cloud data centre is, which are discussed below:
- Private cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.
- Community cloud: The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.
- Public cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
- Hybrid cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

MapReduce Programming Model

- MapReduce is a programming model introduced by Google for large scale data processing. These processing is parallel essentially, so we can give the large-scale data analysis to any operators with enough machines [5]. MapReduces core task is to divide the data into different logic blocks, programs written with the distributed properties model, can process on distributed clusters in parallel [6].

- The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. MapReduce model is inspired by the map and reduce functions commonly used in functional programming. The Map function takes an input pair, and then generates an intermediate key/value pairs set. MapReduce library put all values with the same intermediate key together, then pass them to the Reduce function.
- The Reduce function accepts an intermediate key and related value, it combined the value to form a set of relatively small value set, and usually this collection is smaller than the input. Typically just 0 or 1 output value is produced per Reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator [5]. The MapReduce data process model is shown in Figure 2.
- The Map/Reduce framework consists of a single master and one worker in each node of machines. The master is responsible for scheduling the jobs component tasks on the workers, monitoring them and re-executing the failed tasks and also the master keeps several data structures. For each map task and reduce task, it stores the state (idle, in-progress, or completed), and the identity of the worker machine (for non-idle tasks).
- After the assignment of map tasks to the registered workers, the job of the worker is to process the assigned job and return the intermediate result to master. Master collects all the results and consolidate the intermediate result, then assign the reduce task to the workers. Number of reduce workers depend on the size of the intermediate result. After collecting the reduce result from the workers, master returns the consolidated data to the user program [4][5].
- MapReduce has slowly gained popularity with its swift and efficient data-intensive processing abilities, so the MapReduce model has become a widely acclaimed processing model for big data and also for complex computations on these data [7][8].

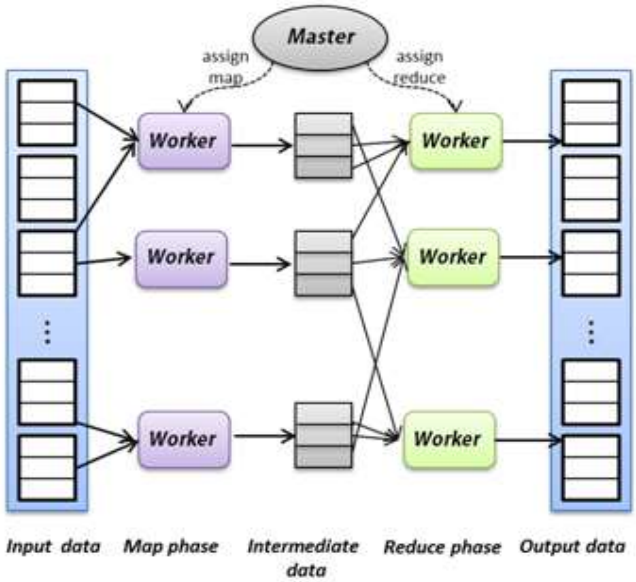

Fig 2. MapReduce Process Architecture

MapReduce has many implementations, Hadoop is an open source implementation of MapReduce. It is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. According to the MapReduce programming model, It is designed to scale up from a single server to thousands of machines, with a very high degree of fault tolerance [9].

Hadoop MapReduce uses the HDFS distributed parallel file system for data storage, which stores the data across the local disks of the computing nodes while presenting a single file system view through the HDFS API [9].

MapReduce In The Cloud

In cloud environments data processing has become an important research problem. As a cloud is a proper distributed system platform, parallel programming model like MapReduce is widely used for developing scalable and fault tolerant applications deployable on cloud [10].

In this new architecture of MapReduce implementation we can use of MapReduce programming model advantages in a cloud computing environment for process large scale data. With implementation of the MapReduce model in cloud, MapReduce resource requirements prepared from cloud services or resources (e.g., network, storage, computing, services).

The main purpose of implementation MapReduce in the cloud is that a cloud provider offers the MapReduce programming model as a service, So consumers Including regular Internet users or small organizations that do not have adequate equipment for the processing of their data can upload his data on storage of cloud datacenter and cloud service process these data with the use of two user-defined map and reduce function and return back the result of processing to the consumer. The user defines map and reduce functions and these functions are sent to the cloud.

Characteristics

The characteristics of the implementation MapReduce on a cloud platform as a service for processing large scale data sets are defined as follows [11]:

- Availability: Service must be always accessible even on the occasions where there is a network failure or a whole datacenter has gone offline.
- Scalability: Cloud service must be able to support very large databases with very high request rates at very low latency. And with adding tasks and operations to cloud, cloud resource management prepares resource for tasks without much effort beyond that of adding more hardware. So we do not have a drop in performance.
- Multitenancy: Cloud service must be able to support many applications (tenants) on the same hardware and software infrastructure. However, the performance of this tenant must be isolated from each other. Adding a new tenant should require little or no effort beyond that of ensuring that enough system capacity has been provisioned for the new load.
- Fault Tolerance: For transactional workloads, a fault tolerant cloud data management system needs to be able to recover from a failure without losing any data or updates from recently committed transactions.

Why MapReduce in the cloud?

While clouds offer raw computing power combined with cloud infrastructure services offering storage and other services, there is a need for distributed computing frameworks to harness the power of clouds both easily and effectively.

Implementation of MapReduce cloud takes advantage of the scalability, high availability and the distributed nature of cloud infrastructure services, guaranteed by cloud service provider, to deliver a fault tolerant, dynamically scalable runtime with a familiar programming model for the users [12].

The most important advantages of cloud MapReduce are listed as follows [13].

Scalability: we have adopted a fully distributed architecture for implementation of MapReduce in the cloud. And we do not have a single master node as a bottleneck. Besides these

advantages, on demand and scalable cloud services make MapReduce implementation on top of these services, is scalable and reliable.

Simplicity: when we implement the MapReduce as a service in the cloud computing every part of the MapReduce operations (e.g., data storing, data processing, data management) is service based. So in MapReduce operations do not exist this need to know the storage service or other services how to work.

Faster acting: Parallelize processing and data transferring: Cloud MapReduce starts uploading reduce results as soon as they are produced in the map phase even before a map task finishes. This parallelizes the network transfer with the CPU intensive processing.

No disk paging: Since the number of key-value pairs in a reduce task is unbounded, Because Hadoop implementation of MapReduce there is no enough memory available then may have to spill partial sorting results to disk multiple times in order to fit within the main memory. But in cloud we have a lot of memory, then we do not need paging.

No staging: Hadoop always stores the intermediate results on disks and then copies over the results to the hard disks on the destination node when instructed by the master. As a result, the data not only transits through the network once, but it also transits twice through the local disk. In comparison, MapReduce in the cloud can do everything in memory; therefore, the data only transits once through the network.

Incast problem: Hadoop and other MapReduce implementations start to shuffle data from mappers to reducers at the end of the map stage. The simultaneous transfers of a large amount of data could overflow the switch buffer, resulting in packet losses, which in turn causes TCP to back off. Current TCP implementations require a 200ms wait time before they retry, which significantly lower the overall throughput. This problem is referred to as the incast problem, and it has been observed in data centers [14]. In contrast, MapReduce in the cloud starts to transfer data as soon as it is generated. Because traffic is smoothed out, it is unlikely to trigger the incast problem.

Implementations of Mapreduce in the cloud environments

Cloud services are on demand, scalable and high available services. In addition distributed nature of such services cause to avoid single point of failure [17][11]. So implementing of MapReduce on the top of cloud services allows MapReduce take advantage of cloud services.

There are two implementations of cloud MapReduce in cloud computing environments, AzureMapReduce [12] and Amazon Elastic MapReduce [16]. We explain these implementations and also have a comparison between them at the end of the paper.

Azure MapReduce

Azure MapReduce is a PaaS cloud that offers a platform as a service to users. As mentioned for implementing MapReduce on the top of cloud services, we need capabilities such as storage, and compute instance and so on. Azure MapReduce provides all of these capabilities. Beside this we need Scheduling mechanism to deploy MapReduce in the cloud. Azure has a queue mechanism that is used for scheduling of MapReduce operations.

The Azure storage queue is an eventual consistent, reliable, scalable and distributed web scale message queue service, ideal for small, short-lived, transient messages. This messaging framework can be used as a message-passing mechanism to communicate between distributed components of any application running in the cloud [12].

The Azure BLOB service provides storage service. We can utilize this service to perform MapReduce operation on top on Microsoft Azure cloud. This service is a scalable distributed storage service that can be easily used to store and retrieve data from it. Azure MapReduce uses Azure BLOB and Azure queue with virtual compute instances to perform MapReduce operations. The architecture of MapReduce implementation on the top of Microsoft Azure cloud is shown in Figure 3 [12].

As mentioned many of MapReduce implementations such as Hadoop use single node to control operations. This node called master node, handles task assignment, fault tolerance and mentoring for map and reduce computations. Hadoop implementation supposes master node failure is rare. Cloud environments are more brittle than the traditional computing clusters are. Thus, cloud applications should be developed to anticipate and withstand these failures. Because of this, it is not possible for AzureMapReduce to make the same assumptions of reliability about a master node as in the above-mentioned runtimes. Due to these reasons, AzureMapReduce is designed around a decentralized control model without a master node, thus avoiding the possible single point of failure [12].

This implementation uses Azure queues for map and reduces task scheduling, Azure tables for metadata and monitoring data storage, Azure BLOB storage for input, output and intermediate data storage and the Window Azure Compute worker roles to perform the computations. The map and reduce tasks of the AzureMapReduce runtime are dynamically scheduled using a global queue.

Amazon elastic MapReduce (EMR)

Amazon Elastic MapReduce is an Amazon implementation of MapReduce framework that uses Amazon infrastructure services for MapReduce operations. This implementation use EC2 APIs to spawn up new virtual machines (also called instances) to process new MapReduce jobs [15][16]. To store input and possibly output data in use S3 services. By leveraging the distributed nature of S3, it can achieve higher data throughput since data come from multiple servers and communications with the servers potentially all traverse different network paths. This implementation use SQS, which is a critical component that allows designing MapReduce in a simple way. A queue serves two purposes. First, it is a synchronization point where workers (a process running on an instance) can coordinate job assignments. Second, a queue serves as a decoupling mechanism to coordinate data flow between different stages. Lastly, it uses SimpleDB, which serves as the central job coordination point in this fully distributed implementation. It keeps all workers' status here [13]. The architecture of the implementation of EMR is shown in Figure 4.

This implementation uses SQS queue to handle communication in MapReduce implementation. Size of messages in queues should be less than 8KB. For example each entry in input queue holds the pointer to a chunk of MapReduce input data. To handle MapReduce operations, one output queue, one MapReduce queue and multiple reduce queue should be used.

A set of map workers, each runs as a separate EC2 instance, poll the input queue for work, then Map Worker invokes the user-defined map function to process a chunk of input data that assign to it.

Table I. Summary of results

	Apache Hadoop	Azure MapReduce	Amazon EMR
Programming Model	MapReduce	MapReduce - will extend to Iterative MapReduce	MapReduce - will extend to Iterative MapReduce
Data Handling	HDFS (Hadoop Distributed File System)	Azure Blob Storage (max size of objects unlimited)	Amazon S3 (max size of object limited to 5GB)
Scheduling	Data Locality; Rack aware, Dynamic task scheduling through global queue	Dynamic task scheduling through global queue	Dynamic task scheduling through SQS service
Failure Handling	Re-execution of failed tasks; Duplicate execution of slow tasks	Re-execution of failed tasks; Duplicate execution of slow tasks	Re-execution of failed tasks; Duplicate execution of slow tasks; through (SQS's visibility timeout, Simple DB)
Environment	Linux Clusters, Amazon Elastic MapReduce on EC2	Window Azure Compute, Windows Azure Local Development Fabric	Linux Clusters, Amazon Elastic MapReduce on EC2
Intermediate Data Transfer	File, Http	Files, TCP	File, TCP
Dynamic Scalability	No	Yes	Yes
Control Model	Central	Decentral	Central(Simple DB)

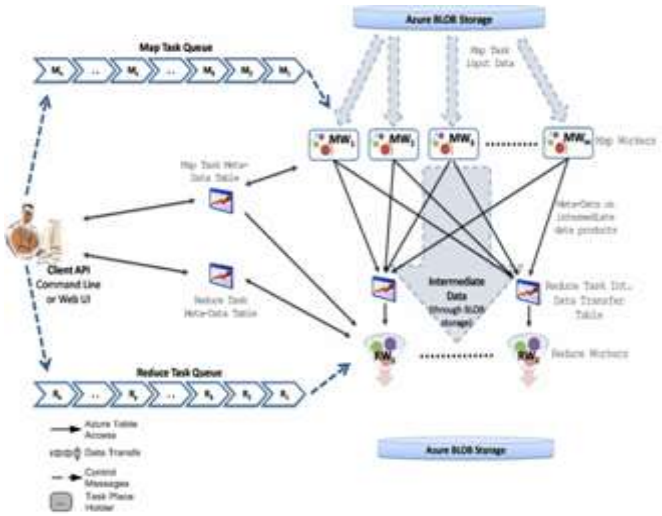

Fig 3. Azure MapReduce Architecture

After all map workers finish their jobs, Reduce workers start their jobs by polling the MapReduce queue. Similar to map worker, reduce worker invoke user-defined function to process a chunk of data that exist in reduce queues and MapReduce queue point to it.

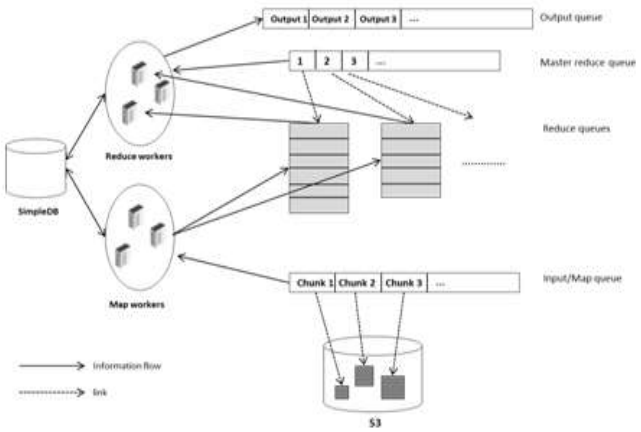

Fig 4. Amazon Elastic MapReduce Architecture

Summary

Generally speaking, according to the contents of the table. I and previously mentioned characteristics of each implementation have different features from other implementations. For example Azure MapReduce framework has a decentralized controlling for control task in workers and in these implementation we do not have bottleneck for controlling. So if we have a failing in the master node we can control the tasks in the workers. Which fulfils the much needed requirement of a distributed programming framework for Azure users. Azure MapReduce is built using Azure cloud infrastructure services that take advantage of the quality of service guarantees provided by the cloud service providers. In contrast, Amazon Elastic MapReduce(EMR) have a central controlling unit for scheduling and controlling of tasks. So in these implementations of MapReduce in the cloud we have bottleneck in controlling unit and if an error occurs in the master we've been faced with serious problems in MapReduce operation and in these aspect Azure MapReduce is superior to Amazon EMR. In data storing and data management each of implementations such as Azure or Amazon EMR for providing storage environment for their operations, for example Amazon employ S3 service and Azure use Azure blob. Limitation in data storing and data management is the main difference in these two data management approach in a cloud environment. Because S3 is a service that offered for public and general users then in these implementations we have limitation more than Azure blob storage for example in S3 the maximum size of each file is 5GB and this maximum size in Azure blob is unlimited.

In cost estimation for this two implementation we can conclude that employment of Amazon EMR for processing our large scale data need to pay cost more than Azure implementation. Because services in Amazon Web Service are more expensive than Azure MapReduce.

Although they have different aspects but these implementations have the same viewpoints in general. For example the architecture of them constructed from four same part storage, computational resource, scheduling queue and a database for preserving task statuses for controlling processes. In rate of parallelism of operations each of two implementation we have the same rate of parallelism.

Conclusion

In analysing and comparison these various MapReduce implementations in the cloud environment we find that there are salient differences between them, for example control flow, task scheduling, and also in data management approaches. We also

present the differences between implementation of MapReduce in the cloud and the traditional implementations (e.g. Hadoop), For example, implementation of MapReduce in the cloud is faster and simpler than implementation in non-cloud environments. Also cloud MapReduce is more scalable.

References

[1] R. Bahaskar Prasad, Eunmi Choi and Iun Lumb, "A Taxonomy and Survey of Cloud Computing Systems,"

[2] R. Buyya, C.S. Yeo and S. Venugopal, Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, Keynote Paper, in Proc. 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008), IEEE CS Press, Sept. 25–27, 2008, Dalian, China.

[3] P. Mell and T. Grance. Definition of cloud computing. Technical report, National Institute of Standard and Technology (NIST), July 2009.

[4] N. Loutas, E. Kamateri, "Cloud Computing Interoperability: The State of Play," 3rd IEEE International Conference on Cloud Computing Technology and Science, Athens, Greece, November 29 - December 1, 2011.

[5] J. Dean, and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107-113, 2008.

[6] R. L¨ammel, "Google's MapReduce programming model — Revisited," *Journal of Science of Computer Programming*, Elsevier, vol. 70, no. 1, pp. 1-30, 2008.

[7] M. Fadhli, T. Abdul Gani et al., "Comparison on Efficiency of Computational Efforts between Cluster Computation (MapReduce) and Single Host Computation," International Conference on Cloud Computing and Social Networking (ICCCSN), April 26-27, 2012, Bandung, Indonesia.

[8] Z. Fadika, E. Dede et al., " MARLA: MapReduce for Heterogeneous Clusters," 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 13-16, 2012, Ottawa, Canada.

[9] Apache Hadoop, Retrieved Aug 20, 2010, from ASF: http://hadoop.apache.org/core/.

[10] B. Thirumala Rao, L. S. S. Reddy, " Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments," *International Journal of Computer Applications*, vol. 34, no. 9, pp.28-32, 2011

[11] S. Sakr, A. Liu and D. Batista et al., " A Survey of Large Scale Data Management Approaches in Cloud Environments," IEEE Communications Surveys & Tutorials, vol. 13, no. 3, pp. 311-336, 2011

[12] T. Gunarathne, T. Lon Wu, J. Qiu et al., " MapReduce in the Clouds for Science," 2nd IEEE International Conference on Cloud Computing Technology and Science, Nov. 29- Dec. 1, 2011, Athens, Greece.

[13] H. Liu, D. Orban, " Cloud MapReduce: a MapReduce Implementation on top of a Cloud Operating System," 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 23-26, 2011, Newport Beach, CA, USA.

[14] R. Griffith, Y. Chen, J. Liu et al., " Understanding TCP Incast Throughput Collapse in Datacenter Networks," In Proc. ACM SIGCOMM WREN Workshop, Aug. 21, Barcelona, Spain, 2009

[15] Amazon Web Services LLC, "Amazon Elastic Compute Cloud (Amazon EC2)," http://aws.amazon.com/ec2/, Last Visit: Jul 2012.

[16] Amazon Web Services LLC, "Amazon Elastic MapReduce (Amazon EMR)," http://aws.amazon.com/elasticMapReduce/, Last Visit: Jul 2012.

[17] L. Zhou, Z. Wang and Xiangfeng, "A Survey of HPC Development," IEEE International Conference on Computer Science and Electronics Engineering (ICCSEE), vol. 2, pp. 103-106, March 23-25, 2012.