CHAPTER 14

THE MapReduce PROGRAMMING MODEL AND IMPLEMENTATIONS

HAI JIN, SHADI IBRAHIM, LI QI, HAIJUN CAO, SONG WU and XUANHUA SHI

14.1 INTRODUCTION

Recently the computing world has been undergoing a significant transforma tion from the traditional noncentralized distributed system architecture typified by distributed data and computation on different geographic areas to a centralized cloud computing architecture, where the computations an data are operated somewhere in the "cloud"—that is, data centers owned an maintained by third party.

The interest in cloud computing has been motivated by many factors [1] such a the low cost of system hardware, the increase in computing power and storag capacity (e.g., the modern data center consists of hundred of thousand of cores an petascale storage), and the massive growth in data size generated by digital medi (images/audio/video), Web authoring, scientific instruments, physical simula tions, and so on. To this end, still the main challenge in the cloud is how to effectivel store, query, analyze, and utilize these immense datasets. The traditional data intensive system (data to computing paradigm) is not efficient for cloud computin due to the bottleneck of the Internet when transferring large amounts of data to distant CPU [2]. New paradigms should be adopted, where computing and dat resources are co-located, thus minimizing the communication cost and benefitin from the large improvements in IO speeds using local disks, as shown in Figure 14.1 Alex Szalay and Jim Gray stated in a commentary on 2020 computing [3]:

> In the future, working with large data sets will typically mean sending computations to data rather than copying the data to your work station.

Cloud Computing: Principles and Paradigms, Edited by Rajkumar Buyya, James Broberg and Andrzej Goscinski

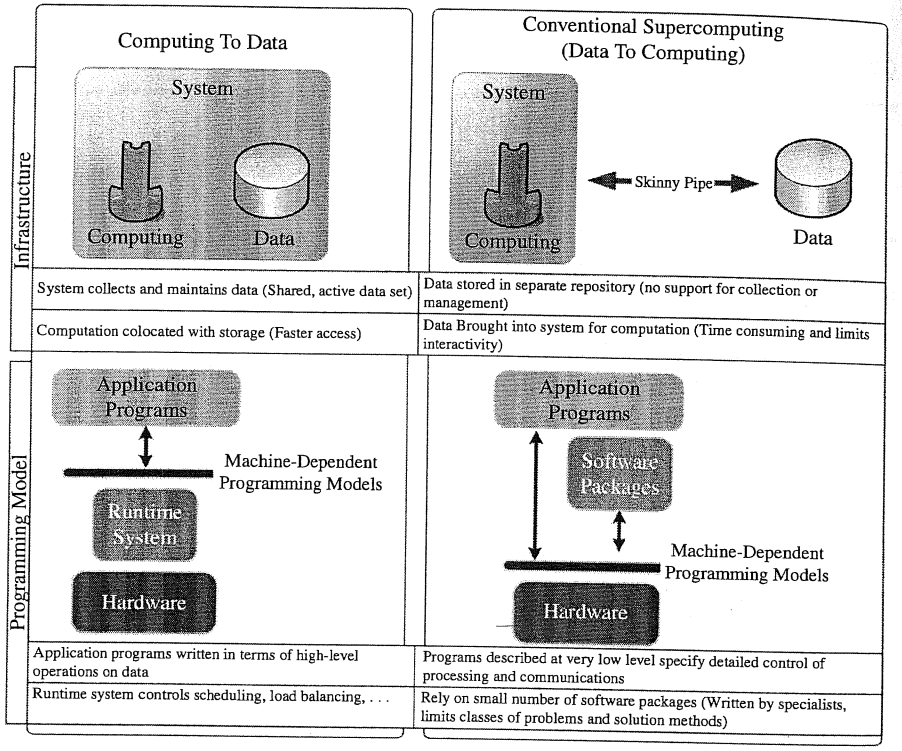

FIGURE 14.1. Traditional Data-to-Computing Paradigm versus Computing-to-Data Paradigm [10].

Google has successfully implemented and practiced the new data-intensive paradigm in their Google MapReduce System (e.g., Google uses its MapReduce framework to process 20 petabytes of data per day [4]). The MapReduce system runs on top of the Google File System (GFS) [5], within which data are loaded, partitioned into chunks, and each chunk is replicated. Data processing is co-located with data storage: When a file needs to be processed, the job scheduler consults a storage metadata service to get the host node for each chunk and then schedules a "map" process on that node, so that data locality is exploited efficiently.

At the time of writing, due to its remarkable features including simplicity, fault tolerance, and scalability, MapReduce is by far the most powerful realization of data-intensive cloud computing programming. It is often advocated as an easier-to-use, efficient and reliable replacement for the traditional data-intensive programming model for cloud computing. More significantly, MapReduce has been proposed to form the basis of the data-center software stack [6].

MapReduce has been widely applied in various fields including data- and compute-intensive applications, machine learning, graphic programming, multi-core programming, and so on. Moreover, many implementations have been developed in different languages for different purposes.

Its popular open-source implementation, Hadoop [7], was developed primarily by Yahoo!, where it processes hundreds of terabytes of data on at least 10,000 cores [8], and is now used by other companies, including Facebook, Amazon, Last.fm, and the *New York Times* [9]. Research groups from the enterprise and academia are starting to study the MapReduce model for better fit for the cloud, and they explore the possibilities of adapting it for more applications.

14.2 MapReduce PROGRAMMING MODEL

MapReduce is a software framework for solving many large-scale computing problems. The MapReduce abstraction is inspired by the Map and Reduce functions, which are commonly used in functional languages such as Lisp [4]. The MapReduce system allows users to easily express their computation as *map* and *reduce* functions (more details can be found in Dean and Ghemawat [4]):

- The map function, written by the user, processes a key/value pair to generate a set of intermediate key/value pairs:

```
map (key1, value1) → list (key2, value2)
```

- The reduce function, also written by the user, merges all intermediate values associated with the same intermediate key:

```
reduce (key2, list (value2)) → list (value2)
```

14.2.1 The Wordcount Example

As a simple illustration of the Map and Reduce functions, Figure 14.2 shows the pseudo-code and the algorithm and illustrates the process steps using the widely used "Wordcount" example. The Wordcount application counts the number of occurrences of each word in a large collection of documents.

The steps of the process are briefly described as follows: The input is read (typically from a distributed file system) and broken up into key/value pairs (e.g., the Map function emits a word and its associated count of occurrence, which is just "1"). The pairs are partitioned into groups for processing, and they are sorted according to their key as they arrive for reduction. Finally, the key/value pairs are reduced, once for each unique key in the sorted list, to

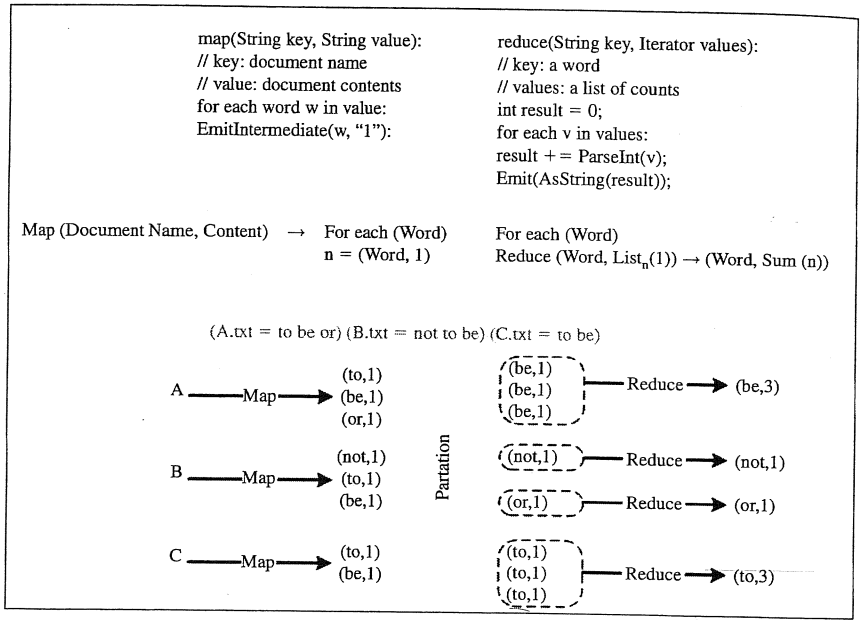

FIGURE 14.2. The Wordcount example.

roduce a combined result (e.g., the Reduce function sums all the counts nitted for a particular word).

4.2.2 Main Features

this section we list the main features of MapReduce for data-intensive plication:

- Data-Aware. When the MapReduce-Master node is scheduling the Map tasks for a newly submitted job, it takes in consideration the data location information retrieved from the GFS-Master node.
- Simplicity. As the MapReduce runtime is responsible for parallelization and concurrency control, this allows programmers to easily design parallel and distributed applications.
- Manageability. In traditional data-intensive applications, where data are stored separately from the computation unit, we need two levels of management: (i) to manage the input data and then move these data and prepare them to be executed; (ii) to manage the output data. In contrast, in the Google MapReduce model, data and computation are allocated, taking advantage of the GFS, and thus it is easier to manage the input and output data.

- Scalability. Increasing the number of nodes (data nodes) in the system will increase the performance of the jobs with potentially only minor losses.
- fault Tolerance and Reliability. The data in the GFS are distributed on clusters with thousands of nodes. Thus any nodes with hardware failures can be handled by simply removing them and installing a new node in their place. Moreover, MapReduce, taking advantage of the replication in GFS, can achieve high reliability by (1) rerunning all the tasks (completed or in progress) when a host node is going off-line, (2) rerunning failed tasks on another node, and (3) launching backup tasks when these tasks are slowing down and causing a bottleneck to the entire job.

14.2.3 Execution Overview

As shown in Figure 14.3, when the user program calls the MapReduce function, the following sequence of actions occurs. More details can be found in Dean and Ghemawat [4]:

The MapReduce library in the user program first splits the input files into M pieces of typically 16 to 64 megabytes (MB) per piece. It then starts many copies

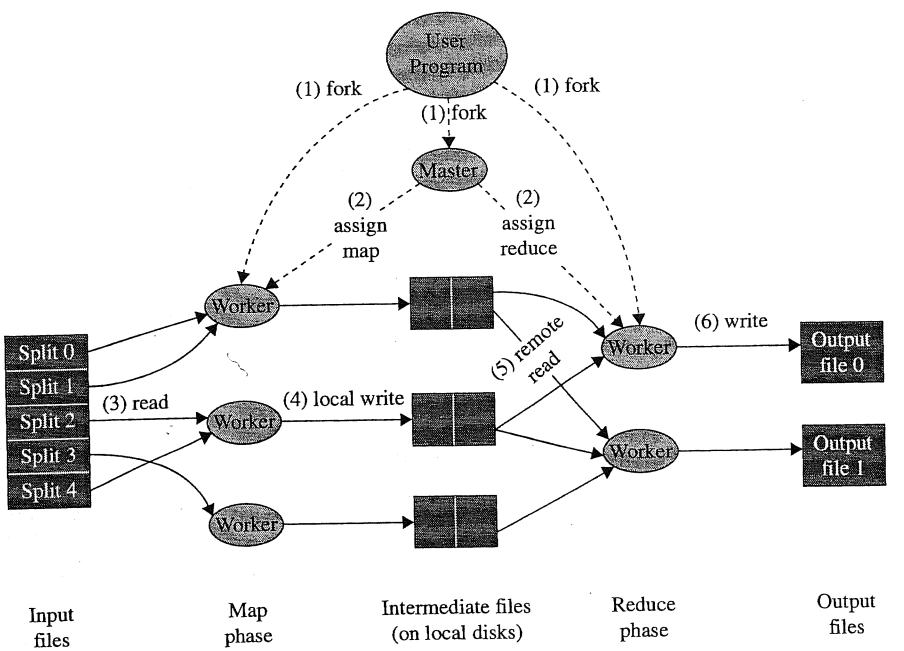

FIGURE 14.3. MapReduce execution overview [4].

of the program on a cluster. One is the "master" and the rest are "workers." The master is responsible for scheduling (assigns the map and reduce tasks to the worker) and monitoring (monitors the task progress and the worker health).

When map tasks arise, the master assigns the task to an idle worker, taking into account the data locality. A worker reads the content of the corresponding input split and emits a key/value pairs to the user-defined Map function. The intermediate key/value pairs produced by the Map function are first buffered in memory and then periodically written to a local disk, partitioned into R sets by the partitioning function.

The master passes the location of these stored pairs to the reduce worker, which reads the buffered data from the map worker using remote procedure calls (RPC). It then sorts the intermediate keys so that all occurrences of the same key are grouped together. For each key, the worker passes the corresponding intermediate value for its entire occurrence to the Reduce function. Finally, the output is available in R output files (one per reduce task).

14.2.4 Spotlight on Google MapReduce Implementation

Google's MapReduce implementation targets large clusters of Linux PCs connected through Ethernet switches [11]. Tasks are forked using remote procedure calls. Buffering and communication occurs by reading and writing files on the GFS. The runtime library is written in C++ with interfaces in Python and Java [12]. MapReduce jobs are spread across its massive computing clusters. For example, the average MapReduce job in September 2007 ran across approximately 400 machines, and the system delivered approximately 11,000 machine years in a single month as shown in Table 14.1 [4].

TABLE 14.1. MapReduce Statistics for Different Months [4]

	Aug. '04	Mar. '06	Sep. '07
Number of jobs (1000s)	29	171	2,217
Avg. completion time (sec)	634	874	395
Machine years used	217	2,002	11,081
Map input data (TB)	3,288	52,254	403,152
Map output data (TB)	758	6,743	34,774
Reduce output data (TB)	193	2,970	14,018
Avg. machines per job	157	268	394
Unique implementations			
Map	395	1,958	4,083
Reduce	269	1,208	2,418

14.3 MAJOR MapReduce IMPLEMENTATIONS FOR THE CLOUD

In the following sections, we will introduce some of the major MapReduce implementations around the world as shown in Table 14.2, and we will provide a comparison of these different implementations, considering their functionality, platform, the associated storage system, programming environment, and so on, as shown in Table 14.3.

14.3.1 Hadoop

Hadoop [7] is a top-level Apache project, being built and used by a community of contributors from all over the world [13]. It was advocated by industry's premier Web players—Google, Yahoo!, Microsoft, and Facebook—as the engine to power the cloud [14]. The Hadoop project is stated as a collection of various subprojects for reliable, scalable distributed computing [7]. It is defined as follows [7]:

TABLE 14.2. MapReduce Cloud Implementations

Owner	Imp Name and Website	Start Time	Last Release	Distribution Model
Google	Google MapReduce http://labs.google.com/papers/mapreduce.html	2004	—	Internal use by Google
Apache	Hadoop http://hadoop.apache.org/	2004	Hadoop0.20.0 April 22, 2009	Open source
GridGain	GridGain http://www.gridgain.com/	2005	GridGain 2.1.1 February 26, 2009	Open source
Nokia	Disco http://discoproject.org/	2008	Disco 0.2.3 September 9, 2009	Open source
Geni.com	SkyNet http://skynet.rubyforge.org	2007	Skynet0.9.3 May 31, 2008	Open source
Manjrasoft	MapReduce.net (Optional service of Aneka) http://www.manjrasoft.com/products.html	2008	Aneka 1.0 March 27, 2009	Commercial

TABLE 14.3. Comparison of MapReduce Implementations

	Google MapReduce	Hadoop	Disco	MapReduce.NET	Skynet	GridGain
Focus	Data-intensive	Data-intensive	Data-intensive	Data- and compute-intensive	Data-intensive	Compute-intensive and data-intensive
Architecture Platform	Master–Slave Linux	Master–Slave Cross-platform	Master–slave Linux, Mac OS X	Master–Slave .Net Windows	P2P OS-independent	Master–slave Windows, Linux, Mac OS X
Storage System	GFS	HDFS, CloudStore, S3	GlusterFS	WinDFS, CIFS, and NTFS	Message queuing: Tuplespace and MySQL	Data grid
Implementation Technology	C++	JAVA	Erlang	C#	Ruby	Java
Programming Environment	Java and Python	JAVA, shell utilities using Hadoop streaming, C++ Using Hadoop pipes	Python	C#	Ruby	Java
Deployment	Deployed on Google clusters	Private and public cloud (EC2)	Private and public cloud (EC2)	Using Aneka, can be deployed on private and public Cloud	Web application (Rails)	Private and public cloud
Some Users and Applications	Google	Baidu [46], NetSeer [47], A9.com [48], Facebook [49] ...	Nokia Research center [21]	Vel Tech University [50]	Geni.com [17]	MedVoxel [51], Pointloyalty [52], Traficon [53], ...

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. Hadoop includes these subprojects:

- Hadoop Common: The common utilities that support the other Hadoop subprojects.
- Avro: A data serialization system that provides dynamic integration with scripting languages.
- Chukwa: A data collection system for managing large distributed systems.
- HBase: A scalable, distributed database that supports structured data storage for large tables.
- HDFS: A distributed file system that provides high throughput access to application data.
- Hive: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- MapReduce: A software framework for distributed processing of large data sets on compute clusters.
- Pig: A high-level data-flow language and execution framework for parallel computation.
- ZooKeeper: A high-performance coordination service for distributed applications.

HadoopMapReduce Overview. The Hadoop common [7], formerly Hadoop core, includes file System, RPC, and serialization libraries and provides the basic services for building a cloud computing environment with commodity hardware. The two fundamental subprojects are the MapReduce framework and the Hadoop Distributed File System (HDFS).

The Hadoop Distributed File System is a distributed file system designed to run on clusters of commodity machines. It is highly fault-tolerant and is appropriate for data-intensive applications as it provides high speed access the application data.

The Hadoop MapReduce framework is highly reliant on its shared file system (i.e., it comes with plug-ins for HDFS, CloudStore [15], and Amazon Simple Storage Service S3 [16]).

The Map/Reduce framework has master/slave architecture. The master, called JobTracker, is responsible for (a) querying the NameNode for the block locations, (b) scheduling the tasks on the slave which is hosting the task's blocks, and (c) monitoring the successes and failures of the tasks. The slaves, called TaskTracker, execute the tasks as directed by the master.

Hadoop Communities. Yahoo! has been the largest contributor to the Hadoop project [13]. Yahoo! uses Hadoop extensively in its Web search and advertising businesses [13]. For example, in 2009, Yahoo! launched, according to them, the world's largest Hadoop production application, called

Yahoo! Search Webmap. The Yahoo! Search Webmap runs on a more than 10,000 core Linux cluster and produces data that are now used in every Yahoo! Web search query [8].

Besides Yahoo!, many other vendors have introduced and developed their own solutions for the enterprise cloud; these include IBM Blue Cloud [17], Cloudera [18], Opensolaris Hadoop Live CD [19] by Sun Microsystems, and Amazon Elastic MapReduce [20], as shown in Table 14.4. Besides the

TABLE 14.4. Some Major Enterprise Solutions Based on Hadoop

Or Name	Solution and Website	Brief Description
Yahoo!	Yahoo! Distribution of Hadoop, http://developer.yahoo.com/hadoop/distribution/	The Yahoo! distribution is based entirely on code found in the Apache Hadoop project. It includes code patches that Yahoo! has added to improve the stability and performance of their clusters. In all cases, these patches have already been contributed back to Apache.
Cloudera	Cloudera Hadoop Distribution, http://www.cloudera.com/	Cloudera provides enterprise-level support to users of Apache Hadoop. The Cloudera Hadoop Distribution is an easy-to-install package of Hadoop software. It includes everything you need to configure and deploy Hadoop using standard Linux system administration tools. In addition, Cloudera provides a training program aimed at producers and users of large volumes of data.
Amazon	Amazon Elastic MapReduce, http://aws.amazon.com/elasticmapreduce/	*"Web service that enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data. It utilizes a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2)[17] and Amazon Simple Storage Service (Amazon S3)."*
Sun Microsystems	Hadoop Live CD, http://opensolaris.org/os/project/livehadoop/	This project's initial CD development tool aims to provide users who are new to Hadoop with a fully functional Hadoop cluster that is easy to start up and use.
IBM	Blue Cloud, http://www-03.ibm.com/press/us/en/pressrelease/22613.wss	Targets clients who want to explore the extreme scale of cloud computing infrastructures quickly and easily. *"Blue Cloud will include Xen and PowerVM virtualized Linux operating system images and Hadoop parallel workload scheduling. It is supported by IBM Tivoli software that manages servers to ensure optimal performance based on demand."*

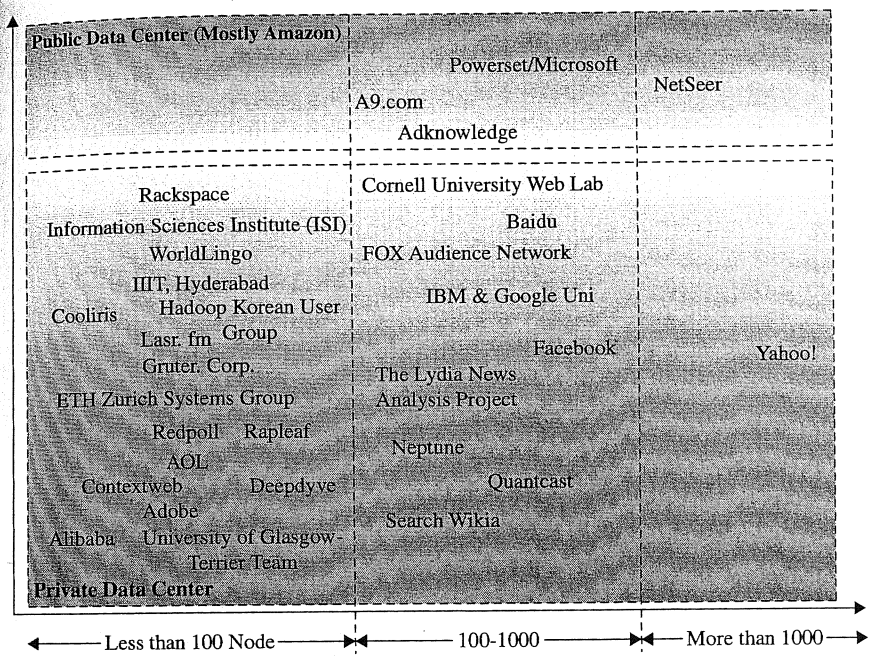

FIGURE 14.4. Organizations using Hadoop to run distributed applications, along with their cluster scale.

aforementioned vendors, many other organizations are using Hadoop solutions to run large distributed computations as shown in Figure 14.4 [9].

14.3.2 Disco

Disco is an open-source MapReduce implementation developed by Nokia [21]. The Disco core is written in Erlang, while users of Disco typically write jobs in Python. Disco was started at Nokia Research Center as a lightweight framework for rapid scripting of distributed data processing tasks. Furthermore, Disco has been successfully used, for instance, in parsing and reformatting data, data clustering, probabilistic modeling, data mining, full-text indexing, and log analysis with hundreds of gigabytes of real-world data.

Disco is based on the master-slave architecture as shown is Figure 14.5. When the Disco master receives jobs from clients, it adds them to the job queue, and runs them in the cluster when CPUs become available. On each node there is a Worker supervisor that is responsible for spawning and monitoring all the running Python worker processes within that node. The Python worker runs the assigned tasks and then sends the addresses of the resulting files to the master through their supervisor.

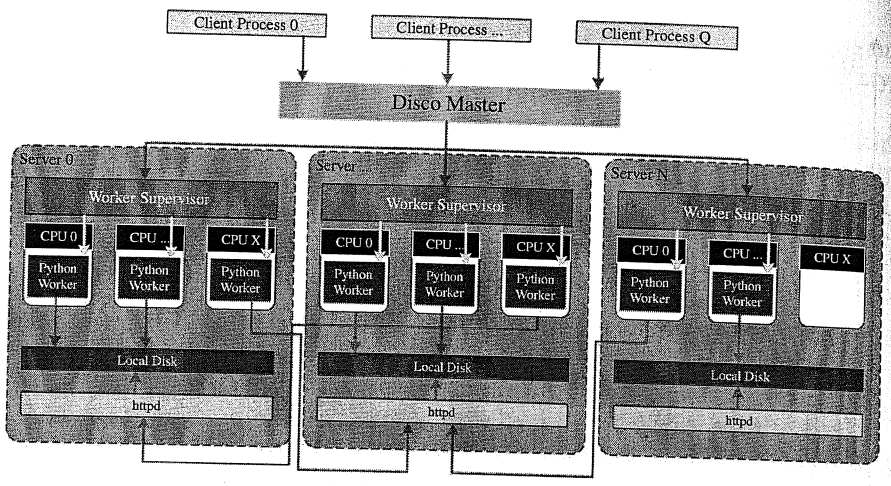

FIGURE 14.5. Architecture of Disco [21].

An "httpd" daemon (Web server) runs on each node which enables a remote Python worker to access files from the local disk of that particular node.

14.3.3 Mapreduce.NET

Mapreduce.NET [22] is a realization of MapReduce for the.NET platform. It aims to provide support for a wider variety of data-intensive and compute-intensive applications (e.g., MRPGA is an extension of MapReduce for GA applications based on MapReduce.NET [23]).

MapReduce.NET is designed for the Windows platform, with emphasis on reusing as many existing Windows components as possible. As shown in Figure 14.6, the MapReduce.Net runtime library is assisted by several components services from Aneka [24, 25] and runs on WinDFS.

Aneka is a.NET-based platform for enterprise and public cloud computing. It supports the development and deployment of.NET-based cloud applications in public cloud environments, such as Amazon EC2.

Besides Aneka, MapReduce.NET is using WinDFS, a distributed storage service over the.NET platform. WinDFS manages the stored data by providing an object-based interface with a flat name space. Moreover, MapReduce.NET can also work with the Common Internet File System (CIFS) or NTFS.

14.3.4 Skynet

Skynet [17, 26] is a Ruby implementation of MapReduce, created by Geni. Skynet is "an adaptive, self-upgrading, fault-tolerant, and fully distributed system with no single point of failure" [17]. At the heart of Skynet is plug-in based message queue architecture, with the message queuing allowing workers to

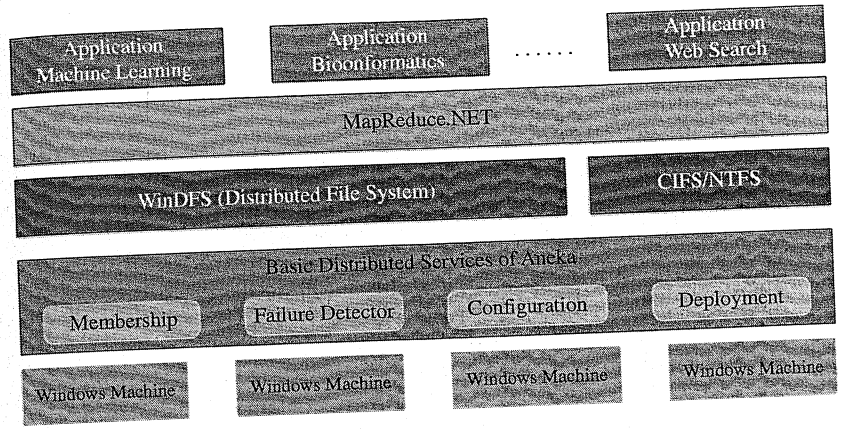

FIGURE 14.6. Architecture of Mapreduce.NET [22].

watch out for each other. If a worker fails, another worker will notice and pick up that task. Currently, there are two message queue implementations available: one built on Rinda [27] that uses Tuplespace [28] and one built on MySQL.

Skynet works by putting "tasks" on a message queue that are picked up by skynet workers. Skynet workers execute the tasks after loading the code at startup; Skynet tells the worker where all the needed code is. The workers put their results back on the message queue.

14.3.5 GridGain

GridGain [29] is an open cloud platform, developed in Java, for Java. GridGain enables users to develop and run applications on private or public clouds. The MapReduce paradigm is at core of what GridGain does. It defines the process of splitting an initial task into multiple subtasks, executing these subtasks in parallel and aggregating (reducing) results back to one final result. New features have been added in the GridGain MapReduce implementation such as: distributed task session, checkpoints for long running tasks, early and late load balancing, and affinity co-location with data grids.

14.4 MapReduce IMPACTS AND RESEARCH DIRECTIONS

Since J. Dean and S. Ghemawat proposed the MapReduce model [4], it has received much attention from both industry and academia. Many projects are exploring ways to support MapReduce on various types of distributed architecture and for a wider range of applications as shown in Figure 14.7.

For instance, QT Concurrent [30] is a C++ library for multi-threaded application; it provides a MapReduce implementation for multi-core computers. Stanford's Phoenix [31] is a MapReduce implementation that targets

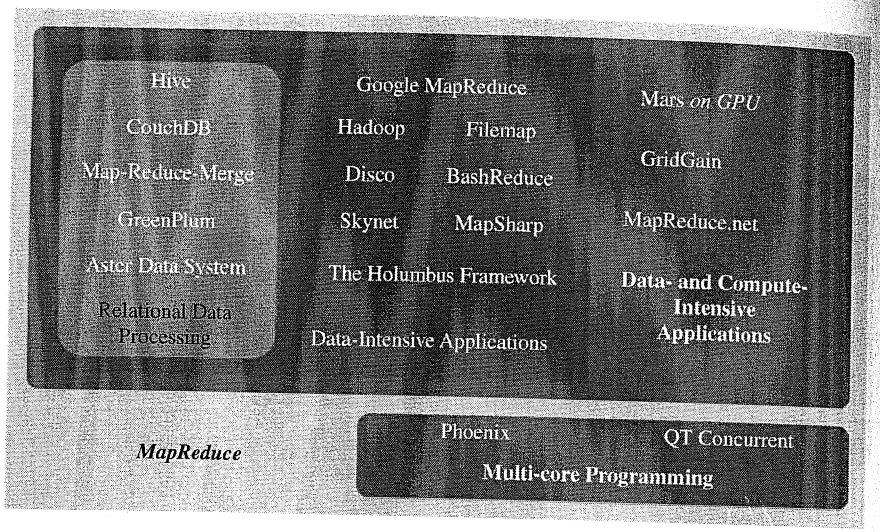

FIGURE 14.7. MapReduce different implementations.

shared memory architecture, while Kruijf and Sankaralingam implemented MapReduce for the Cell B.E. architecture [32]. Mars [33] is a MapReduce framework on graphic processors (GPUs). The Mars framework aims to provide a generic framework for developers to implement data- and computation-intensive tasks correctly, efficiently, and easily on the GPU.

Hadoop [7], Disco [21], Skynet [26], and GridGain [29] are open-source implementations of MapReduce for large-scale data processing. Map-Reduce-Merge [34] is an extension on MapReduce. It adds to MapReduce a merge phase to easily process data relationships among heterogeneous datasets. Microsoft Dryad [35] is a distributed execution engine for coarse-grain data parallel applications. In Dryad, computation tasks are expressed as directed acyclic graph (DAG).

Other efforts [36, 37] focus on enabling MapReduce to support a wider range of applications. S. Chen and S. W. Schlosser from Intel are working on making MapReduce suitable for performing earthquake simulation, image processing and general machine learning computations [36]. MRPSO [38] utilizes Hadoop to parallelize a compute-intensive application, called Particle Swarm Optimization. Research groups from Cornell, Carnegie Mellon, University of Maryland, and PARC are also starting to use Hadoop for both Web data and non-data-mining applications, like seismic simulation and natural language processing [39].

At present, many research institutions are working to optimize the performance of MapReduce for the cloud. We can classify these works in two directions:

The first one is driven by the simplicity of the MapReduce scheduler. In Zaharia et al. [40] the authors introduced a new scheduling algorithm called the

Longest Approximate Time to End (LATE) to improve the performance of Hadoop in a heterogeneous environment by running "speculative" tasks—that is, looking for tasks that are running slowly and might possibly fail—and replicating them on another node just in case they don't perform. In LATE, the slow tasks are prioritized based on how much they hurt job response time, and the number of speculative tasks is capped to prevent thrashing.

The second one is driven by the increasing maturity of virtualization technology—for example, the successful adoption and use of virtual machines (VMs) in various distributed systems such as grid [41] and HPC applications [42, 43]. To this end, some efforts have been proposed to efficiently run MapReduce on VM-based cluster, as in Cloudlet [44] and Tashi [45].

14.5 CONCLUSION

To summarize, we have presented the MapReduce programming model as an important programming model for next-generation distributed systems, namely cloud computing. We have introduced the MapReduce metaphor and identified some of major MapReduce features. We have introduced some of the major MapReduce implementations for cloud computing, especially data- and compute-intensive cloud computing owned by different organizations.

We have presented the different impacts of the MapReduce model in the computer science discipline, along with different efforts around the world. It can be observed that while there has been a lot of effort in the development of different implementations of MapReduce, there is still more to be achieved in terms of MapReduce optimizations and implementing this simple model in different areas.

14.5.1 Acknowledgments

This work is supported by National 973 Key Basic Research Program under Grant 2007CB310900, NSFC under Grants 61073024 and 60973037, Program for New Century Excellent Talents in University under Grant NCET-07-0334, Information Technology Foundation of MOE and Intel under Grant MOE-INTEL-09-03, National High-Tech R&D Plan of China under Grant 2006AA01A115, Important National Science & Technology Specific Projects under Grant 2009ZX03004-002, China Next Generation Internet Project under Grant CNGI2008-109, and Key Project in the National Science & Technology Pillar Program of China under Grant 2008BAH29B00.

REFERENCES

1. I. Foster, Yong Zhao, I. Raicu and S. Lu, Cloud computing and grid computing 360-degree compared, in *Proceedings of the Grid Computing Environments Workshop (GCE '08)*, 2008, pp. 1–10.

2. A. Szalay, A. Bunn, J. Gray, I. Foster, I. Raicu. The Importance of Data Locality in Distributed Computing Applications, in *Proceedings of NSF Workflow*, 2006.
3. A. S. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, D. Slutz, and R. J. Brunner, Designing and mining multi-terabyte astronomy archives: The Sloan Digital Sky Survey, in *Proceedings of the SIGMOD International Conference on Management of Data*, 2000, pp. 451–462.
4. J. Dean and S. Ghemawat, MapReduce: Simplified data processing on large clusters, *Communications of the ACM*, **51**(1):107–113, 2008.
5. S. Ghemawat, H. Gobioff, and S. T. Leung, The Google File System, in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, Lake George, NY, October, 2003, pp. 29–43.
6. D. A. Patterson, Technical perspective: The data center is the computer, *Communications of the ACM*, **51**(1):105, 2008, pp. 105–105.
7. Hadoop: http://lucene.apache.org/
8. Yahoo!, Yahoo! Developer Network, http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html (accessed September 2009).
9. Hadoop, Applications powered by Hadoop: http://wiki.apache.org/hadoop/PoweredB
10. Presentation by Randal E. Bryant, Presented in conjunction with the 2007 Federated Computing Research Conference, http://www.cs.cmu.edu/~bryant/presentations/DISC-FCRC07.ppt.
11. L. Barroso, J. Dean, and U. Holzle, Web search for a planet: The Google cluster architecture, *IEEE Micro*, **23**(2), 2003, pp. 22–28.
12. MapReduce in Wikipedia, http://en.wikipedia.org/wiki/MapReduce (accessed September 2009).
13. Hadoop in Wikipedia, http://en.wikipedia.org/wiki/Hadoop (accessed September 2009).
14. CNET news, http://news.cnet.com/8301-13505_3-10196871-16.html (accessed September 2009).
15. CloudStore (Formerly Kosmos File System), http://kosmosfs.sourceforge.net/
16. Amazon Simple Storage Service, http://aws.amazon.com/s3/
17. Ruby MapReduce Implementation, http://en.oreilly.com/rails2008/public/schedule/detail/2022 (accessed September 2009).
18. Cloudera Homepage, http://www.cloudera.com/
19. OpensolarisHadoop Live CD, http://opensolaris.org/os/project/livehadoop/
20. Amazon Elastic MapReduce, http://aws.amazon.com/elasticmapreduce/
21. Disco Project Homepage, http://discoproject.org/
22. C. Jin and R. Buyya, MapReduce programming model for.NET-based cloud computing, in *Proceedings of the 15th International European Parallel Computing Conference (EuroPar 2009)*, Delft, The Netherlands, August 25–28, 2009, pp. 417–428.
23. C. Jin, C. Vecchiola, and R. Buyya, MRPGA: An Extension of MapReduce for Parallelizing Genetic Algorithms, in *Proceedings of 4th IEEE International Conference on e-Science*, Indiana, USA, 2008, pp. 214–221.

24. X. Chu, K. Nadiminti, J. Chao, S. Venugopal, and R. Buyya, Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications, *Proceedings of the 3rd IEEE International Conference and Grid Computing*, Bangalore, India, December 10–13, 2007, pp. 151–159.
25. Manjrasoft Products, http://www.manjrasoft.com/products.html (accessed September 2009).
26. Skynet, http://skynet.rubyforge.org/
27. Rinda Doc page, http://www.ruby-doc.org/stdlib/libdoc/rinda/rdoc/index.html
28. Tuplespace in Wikipedia, http://en.wikipedia.org/wiki/Tuple_space (accessed September 2009).
29. GridGain, http://www.gridgain.com/
30. QT concurrent Page, http://labs.trolltech.com/page/Projects/Threads/QtConcurrent
31. C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, C. Kozyrakis, Evaluating MapReduce for Multi-core and Multiprocessor Systems, in *Proceedings of the 13th International Symposium on High-Performance Computer Architecture (HPCA)*, Phoenix, AZ, February 2007, pp. 13–24.
32. M. Kruijf and K. Sankaralingam, MapReduce for the Cell B. E. Architecture, TR1625, Technical Report, Department of Computer Sciences, the University of Wisconsin—Madison, 2007.
33. B. S. He, W. B. Fang, Q. Luo, N. K. Govindaraju, and T. Y. Wang, Mars: A MapReduce framework on graphics processors, in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, Toronto, Ontario, Canada, 2008, pp. 260–269.
34. H. C. Yang, A. Dasdan, R. L. Hsiao, and D. S. P. Jr, Map-reduce-merge: Simplified relational data processing on large clusters, in *Proceedings of SIGMOD*, 2007, pp. 1029–1040.
35. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, Dryad: Distributed data-parallel programs from sequential building blocks, in *Proceedings of the European Conference on Computer Systems (EuroSys)*, Lisbon, Portugal, March, 2007, pp. 59–72.
36. S. Chen, and S. W. Schlosser, Map-Reduce Meets Wider Varieties of Applications, IRP-TR-08-05, Technical Report, Intel Research Pittsburgh, May 2008.
37. R. E. Bryant, Data-Intensive Supercomputing: The Case for DISC, CMU-CS-07-128, Technical Report, Department of Computer Science, Carnegie Mellon University, May 2007.
38. A. W. McNabb, C. K. Monson, and K. D. Seppi, Parallel PSO Using MapReduce, in *Proceedings of the Congress on Evolutionary Computation*, Singapore, 2007, pp. 7–14.
39. Presentations by Steve Schlosser and Jimmy Lin at the 2008 Hadoop Summit, http://developer.yahoo.com/hadoop/summit/ (accessed September 2009).
40. M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, Improving mapreduce performance in heterogeneous environments in, *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation*, 2008, pp. 29–42.
41. R. Figueiredo, P. Dinda, and J. Fortes, A Case for Grid Computing on Virtual Machines, in *Proceedings of the 23rd International Conference on Distributed Computing Systems*, 2003, pp. 550–559.

42. M. F. Mergen, V. Uhlig, O. Krieger, and J. Xenidis, Virtualization for high-performance computing. SIGOPS operating systems review, **40**(2):8–11, 2006.
43. W. Huang, J. Liu, B. Abali, and D. K. Panda, A case for high performance computing with virtual machines, in *Proceedings of the 20th ACM International Conference on Supercomputing (ICS'06)*, Cairns, Australia, June 2006, pp. 125–134.
44. S. Ibrahim, H. Jin, B. Cheng, H. J. Cao, W. Song and L. Qi, Cloudlet: Towards MapReduce implementation on Virtual machines, in *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing (HPDC-18)*, Germany, 2009, pp. 65–66.
45. Tashi Homepage, http://www.pittsburgh.intel-research.net/projects/tashi/, 2009.
46. Baidu search engine, www.baidu.com
47. NetSeerHomePage, http://www.netseer.com/
48. A9.com Homepage, www.a9.com
49. Facebook Homepage, www.facebook.com/
50. Manjrasoft customers, http://www.manjrasoft.com/customers.html (accessed September 2009).
51. Medvoxel homepage, http://www.medvoxel.com/
52. Pointoyalty Homepage, http://www.pointloyalty.ru/
53. Traficon Homepage, http://www.traficon.com/
54. Amazon Elastic MapReduce, http://aws.amazon.com/ec2/
55. IBM Blue Cloud Announcement, http://www-03.ibm.com/press/us/en/pressrelease/22613.wss, (accessed September 2009).

PART IV

MONITORING AND MANAGEMENT