# Parallel Programming on Cloud Computing Platforms
## ─ Challenges and Solutions

Yi Pan
Department of Computer Science
Georgia State University
34 Peachtree Street, Room 1442
Atlanta, GA 30302-3994, USA
Email: pan@cs.gsu.edu

Junbo Zhang
[1]School of Information Science and Technology
Southwest Jiaotong University, Chengdu 610031, China
[2]Department of Computer Science
Georgia State University, Atlanta, GA 30303, USA
Email: JunboZhang86@163.com; jbzhang@cs.gsu.edu

*Abstract*—**Cloud computing has been very successful in handling big data such as in searching engines and database applications. Programming models on cloud computing systems are originally designed for data parallel applications, and hence may not be suitable for compute-intensive applications. How to expand the horizon of cloud computing systems so that more applications can be executed on them is a major challenge facing us. Parallel programming models have been studied on many different platforms; however, research on programming models on cloud systems is still in its infancy. How to effectively implement parallel codes on clouds is still an issue. Automatic parallelization of sequential codes into parallel codes on clouds is also a challenge. In this paper, we will point out the shortcomings and limitations of current cloud computing programming models and propose possible solutions to handling parallel processing and real-time computing tasks on cloud computing systems.**

***Keywords-*** *cloud computing; parallel programming; code generation; code translation*

## I. Introduction

Cloud computing services such as platform as a service does provide users a convenient environment for developing applications; users don't need to spend time and money on purchasing and maintaining machines. In addition, users also do not have to purchase the latest licenses for operating systems and software; these features provided by a cloud service allow developers to focus on producing their applications. Cloud computing platforms such as Windows Azure even allow users to develop their applications using common programming languages such as .net, java, and python. In addition, Windows Azure also provides a management portal which offers a user friendly interface to manage the deployed application. With the features, users can easily develop their applications with a familiar programming language and deploy applications in the cloud. Cloud computing is also one of the best solutions for individuals who require lots of computational power and storage space to run heavy processes or analyze large data. These tasks are almost impossible to complete in personal computers because of its hardware limitation. In this situation, users can utilize the cloud service to achieve the aforementioned goals. So far, cloud computing has been very successful for many data parallel applications such as web searching and database applications.

Because cloud computing is mainly for large data center applications, the programming models used in current cloud systems have many limitations and are not suitable for many scientific applications. Without automatic parallelization software for many applications, the usage of cloud computing will be limited. In other words, we cannot utilize the massive computing power in the system unless we can perform effective distributed computing and program the applications easily. In this paper, we point out the limitations of current cloud computing programming models, and propose several solutions. We are currently designing and implementing an automatic translator for Matlab to MapReduce. We plan to test the resultant codes generated by our translator on several applications. Our hope is that the automatic generated cloud computing codes are comparable to codes produced by hand in terms of performance.

## II. Cloud Computing Programming Models

In order to perform parallel computing, parallel programming models are essential for users to implement their parallel codes. For example, MPI is used on distributed computing platforms, while OpenMP is used for parallel programming on shared memory systems. On cloud systems, there is no exception here. Otherwise, the codes developed run only on one core and hence execute sequentially without exploiting the power of massive number of processors on a cloud system. In the following, we will review a few programming models on popular cloud systems.

Windows Azure [1, 2] is an important programming model and contains two components: a Compute Component and a Storage Component. The two major role types in a compute component are a web role and a worker role, which are utilized on Windows Azure for computational purposes. The Web role provides the web service with Internet Information Services (IIS). It allows programmers to create user interfaces for customers to utilize the web service which is hosted on the Windows Azure environment. The Worker role is used to process applications in the background. Most computing processes are executed on worker role instances. There are three different types of storages for Windows in a storage component in Azure, which are Queue, Table, and Blob storages. Queue storage plays a significant role by providing asynchronous work dispatch in order to enable communication between independent role instances. Blob (Binary Large Object) storage provides developers a solution to store large size data such as videos, audios, and also large amounts of unstructured text in Windows Azure. Table storage has the feature that maintains service state by providing structured storage. Users can create a table on Windows Azure which contains a set of entities. Communications among different processes are through these storages.

For the purpose of processing large data, Google developed a software framework called MapReduce [3, 4] to support large distributed applications using a large number of computers, collectively referred to as a cluster. As one of the most important cloud computing techniques, MapReduce has been a popular computing model for cloud computing platforms and is effective to analyze large amounts of data.

With the MapReduce model, the computation takes a set of input <key, value> pairs, and produces a set of output <key, value> pairs. Computation in a MapReduce model includes two major functions: Map function and Reduce function. In a Map function, it takes an input <key, value>pair and produces a set of intermediate <key, value> pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function. In the Reduce function, after accepting an intermediate key and a set of values for that key, it merges together the values which have the same key, forms a possibly smaller set of values and outputs the final <key, value> pairs.

Followed by Google's work, many MapReduce runtime systems have rapidly emerged. For example, (1) Apache Hadoop [5] was developed for data-intensive distributed applications, and it is an open source software framework, which helps construct the reliable, scalable distributed systems. (2) Phoenix [6] is a shared-memory implementation of MapReduce model for data-intensive processing tasks, which can be used to program multi-core chips as well as shared-memory multiprocessors (SMPs and ccNUMAs). (3) Aiming at providing a generic framework for developers to implement data- and computation-intensive tasks correctly, efficiently, and easily on the GPU, Mars [7] is developed for graphic processors (GPUs) using MapReduce framework. Mars hides the programming complexity of the GPU behind the simple and familiar MapReduce interface. Hence, the developers can write their code on the GPU without any knowledge of the graphics APIs or the GPU architecture. (4) MapReduceRoles4Azure (MR4Azure) [8] is a decentralized, dynamically scalable MapReduce runtime for Windows Azure that was developed using Azure cloud infrastructure services as the building blocks. MR4Azure uses Azure Queues for map and reduce task scheduling, Azure Tables for metadata & monitoring data storage, Azure Blob storage for input, output and intermediate data storage and the Windows Azure Compute worker roles to perform the computations.

Dryad [9] is another popular cloud computing programming model for a general purpose runtime for execution of data parallel applications. It supports parallel applications that resemble Directed Acyclic Graphs (DAGs) in which the vertices represent computation units, and the edges represent communication channels between different computation units. Consisting of the Dryad distributed execution engine and the .NET Language Integrated Query (LINQ), DryadLINQ [10] was developed by Microsoft, which is a simple, powerful, and elegant programming environment for writing large-scale data parallel applications running on a large number of clusters.

MapReduce and Dryad are two popular platforms in which the dataflow takes the form of a directed acyclic graph of operators. These platforms lack built-in support for iterative programs, which arise naturally in many applications including data mining, web ranking, graph analysis, model fitting, and so on. To overcome this disadvantage, (1) Iterative the MapReduce model was introduced in Twister [11], which is a lightweight MapReduce runtime. It provides the feature for cacheable MapReduce tasks, which allow developers to –produce or create iterative applications without spending much time on reading and writing large amounts of data in each iteration. A version for Windows Azure called Twister4Azure [19] also has been released. (2) In addition, Microsoft also developed an iterative MapReduce runtime for Windows Azure, code-named Daytona [12], which is designed to support a wide class of data analytics and machine learning algorithms. It can scale out to hundreds of server cores for analysis of distributed data. (3) HaLoop [13] is a modified version of the Hadoop MapReduce framework, designed to serve iterative applications. It not only extends MapReduce with programming support for iterative applications, but also dramatically improves their efficiency by making the task scheduler loop-aware and by adding various caching mechanisms.

## III. LIMITATIONS OF CURRENT PROGRAMMING MODELS

The current programming models used in cloud computing systems such as MapReduce is designed for handling large data parallel applications such as web searching. Hence, they have many limitations and are not suitable for many scientific applications, which have many data dependency relationships among the data they handle. For example, MapReduce cannot express many applications such as matrix operations easily. It has a low physical node utilization, implying a low return on investment. Many communication patterns which are required in compute-intensive applications are not supported in MapReduce. Parallel applications can utilize various communication constructs to build diverse communication topologies, e.g., a matrix multiplication application. The current cloud runtimes, which are based on data flow models such as MapReduce and Dryad, do not support this behavior.

Coding in Microsoft Azure is also very tedious and time consuming. The system defines Tables and Queues for communication, but users have to arrange the communication and synchronization of their codes to make sure that data dependency and operation sequence are kept as in the original sequential codes.

Furthermore, developers have to learn how to program with these models and spend time on understanding characteristics of different cloud platforms. Without automatic parallelization software for many applications, the usage of cloud computing will be limited. In other words, we cannot utilize massive computing power in the system unless we can perform effective distributed computing and program the applications easily.

Due to the many different programming models, the formats of the source codes are quite different. One user has to learn the different syntax and semantics of various cloud programming models in order to develop codes on different platforms. It is tedious and labor costly. To solve this problem, automatic translation of different source codes with different formats is a possible solution.

Real-time data processing on cloud systems is also a challenge since most real-time applications require parallel processing to speed up the execution times and process incremental data inputs. We need an infrastructure that can perform massive-scale real-time processing, which MapReduce is suitable. However, it is designed for batch processing over a large dataset stored on a disk, which can incur a long delay to return the desired information from the time the relevant data arrives at the system.

## IV. Possible Solutions

Considering the limitations and shortcomings of current cloud computing runtime systems, we propose the following ideas and approaches to solve or alleviate the problems mentioned above.

### *A. Generalization of MapReduce*

One way is to generalize MapReduce's functionalities so that more applications especially some compute-intensive applications can be parallelized. The problem is that the more the model is generalized, the more complicated it becomes to implement or it becomes impossible to implement the runtimes. In the following, we introduce some existing solutions and later propose new ideas as follows.

(1) Many practical computing problems concern large graphs such as the Web graph and various social networks that have billions of vertices, and trillions of edges. It is a big challenge to process these graphs efficiently. Using the MapReduce model, graph algorithms can be written as a series of chained MapReduce invocations [18]. However, MapReduce is essentially functional, so expressing a graph algorithm as a chain of MapReduce primitives requires passing the entire state of the graph from one stage to the next — in general requiring much more communication and associated serialization overhead. In addition, the need to coordinate the steps of a chained MapReduce adds programming complexity. For reasons of usability and performance, Google introduced a different computational model Pregel [14] which is suitable for this task. Similar to MapReduce, Pregel is a distributed programming framework, focused on providing users with a natural API for programming graph algorithms while managing the details of distribution invisibly, including messaging and fault tolerance. It is a much more efficient support for iterative computations over the graph.

(2) Cloud runtimes like MapReduce and Dryad are well suited for efficiently transforming or analyzing an entire corpus: these systems can simultaneously use a large number of machines to process huge amounts of data quickly. Despite this scalability, MapReduce cannot process small updates individually, re-running a MapReduce pipeline on each small batch of updates results in unacceptable latency and wasted work. By using the incremental method, Google built Percolator [15], which is a system for incrementally processing updates to a large data set. Google also deployed it to create the Google web search index. Percolator avoids the expense of repeated scans by, essentially, creating indexes on the keys used to cluster documents.

(3) Iterative MapReduce (*iMapReduce*) is a modified MapReduce framework for iterative processing. In many applications such as matrix operations and graph algorithms, due to data dependency, many iterative processing is required to compute a task. Twister [11] and HaLoop [13] use iMapReduce to improve performance by reducing the overhead of creating jobs repeatedly, by eliminating the shuffling of static data, and by allowing asynchronous execution of map tasks.

(4) Further, the first alpha version of Apache Hadoop 2.0 [5] is released which is called MapReduce 2.0 (MRv2) or YARN, and focuses on the next generation MapReduce technology. The fundamental idea of MapReduce 2.0 is to split up the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate daemons. The idea is to have a global Resource Manager and per-application Application Master. An application is either a single job in the classical sense of MapReduce jobs or a DAG of jobs. The ResourceManager and per-node slave, the NodeManager (NM), form the data-computation framework. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The per-application ApplicationMaster is, in effect, a framework specific library and is charged with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks.

(5) One possibility is to design new programming models based on existing parallel programming models such as Bulk Synchronous Parallel (BSP) programming model. These programming models provide a mechanism to describe a parallel program. We need to implement the runtimes to support these models so that a computation can be carried out in parallel on a cloud system. The task may be too complicated, and hence certain limitations may be imposed. For example, we may restrict the communication patterns to simple and regular ones. This is appropriate for certain applications.

### *B. Translation between Sequential Codes and Cloud Codes*

Machine translation among certain programming languages is another possibility especially for many data-parallel application programming languages such as SQL and Matlab. The MapReduce framework requires that users implement their applications by coding their own map and reduce functions. Although this low-level hand coding offers a high flexibility in programming applications, it increases the difficulty in program debugging and is also time consuming. High-level declarative languages can greatly simplify the effort on developing applications in MapReduce without hand-coding programs. Recently, several SQL-like declarative languages and their translators have been built and integrated with MapReduce to support these languages. For example, Hive [17] is a production SQL-to-MapReduce translator, which has greatly increased the productivity of writing MapReduce programs. Researchers also noticed that in practice the auto-generated MapReduce programs for many queries are often extremely inefficient compared to hand-optimized programs due to inefficient SQL-to-MapReduce translations which would create many unnecessary jobs. For this reason, YSmart [16] is developed. It is a correlation aware SQL-to-MapReduce translator used to translate SQL code to Hadoop MapReduce code and has been patched in to the Hive [17] data warehousing system. YSmart significantly reduces redundant computations and I/O operations in the MapReduce execution.

We have designed and implemented a translator, called M2M, for automatically translating Matlab codes into MapReduce codes. There are two major operations in Matlab: vector operation and matrix operation. According to the characteristics of vector operations, we have developed a simple version for translating vector operations, called M2M, which can translate some single vector operations, such as computations of MAX, MIN, SUM, MEAN, and STD in a vector. In our future work, we will improve the translator of vector operations and make it suitable for most single vector operations. We will study MapReduce-based parallelization techniques for matrix operations and enhance M2M to support translation of matrix operation.

To translate the complicated statements such as loops in a sequential code, we will think about using a similar strategy as used in OpenMP and High Performance Fortran, *i.e.*, a hint on how to parallelize the loop. The dependency analysis inside the loop will depend on users' inputs in order to alleviate the complication of the translation. Besides loop parallelization, task parallelism can also be exploited if the users indicate that certain procedures or functions can be executed in parallel.

### *C. Automatic Conversion among Cloud Codes*

Automatic code conversion among different cloud computing runtime systems is also a possibility especially for using mixed cloud platforms. Most developers just learn one cloud computing programming language, and do not need to learn a new one as cloud platforms change. For example, Twister [11] and Twister4Azure [19] are both developed by Indian University. They are both using the Iterative MapReduce framework, but for different cloud platforms. Developers need to code in Java and C# for Twister and Twister4Azure, respectively. If an automatic code converter Twister-to-Twister4Azure exists, it would help developers deploy applications to two cloud platforms by using one code. In our future work, we will try to develop an automatic code converter for different cloud platforms.

### *D. Incremental MapReduce*

In MapReduce, data needs to be loaded to the cluster before any queries can be executed, resulting in a high delay to start query processing. Similarly, answers to a long-running query are returned only when the entire job is completed, hence unsuitable for real-time interactive analysis and monitoring problems. Clearly, recomputing the results from scratch every time the input changes or new data arrives is just not a smart idea. We need to develop an incremental MapReduce to perform parallel processing on real-time data which have strong data dependency and urgency to complete in real time. In incremental MapReduce, we also need to add the ability to perform incremental, real-time processing as data arrives at the system. The new system can return timely results and insights while scaling to enormous datasets for incremental data updates. We plan to develop such a system and test it using real-world workloads such as web page clicks. Our goals are to analyze the incremental data efficiently through the new capabilities and reduce- I/O operations and provide real-time results.

### *E. Domain Specific Code Generation*

Many applications have certain communication patterns or computation properties. For example, system dynamics modeling can be done through decomposing a domain into many subdomains and the communications happen only among neighboring processors (subdomains). A class of graph algorithms can also be grouped together and described with certain properties. Right now, one has to spend a lot of time to implement the cloud codes based on the sequential codes. If we can analyze these codes, capture their main features, and provide a framework to describe the codes, we will be able to translate the framework into the cloud codes directly. If the project is successful, a user does not need to know the cloud programming details anymore, and the translation will be accurate and fast. Of course, the domain has to be specific but wide enough to cover many applications in the same domain. If the domain is too narrow, it will be easy to translate, but with limited usage.

## V. CONCLUSIONS

Although cloud computing has been a commercial success for many data-parallel applications such as web searching engines and database applications, its use in speeding up scientific computing applications is still in its infancy. Due to the fact that a lot of runtimes such as MapReduce are designed for data centers and data-intensive applications, a lot of scientific applications cannot be easily parallelized. It is a challenge facing many scientists who need cloud computing to solve their compute-intensive applications. One possibility is to design new cloud programming models and supporting runtimes. Another possibility is to redesign parallel algorithms which are suitable for current cloud computing technologies such as MapReduce. Maybe we need to work on both directions so that cloud computing can also be effectively used in scientific computations. This paper outlines some of the limitations of current programming models in cloud systems and proposes some possible approaches. Currently, we are working on designing and implementing some of our proposed solutions and will compare their performance with current cloud technologies. We firmly believe that cloud computing will be a success not only in data-intensive applications, but also in compute-intensive and real-time applications.

## REFERENCES

[1] Understanding Windows Azure. [Online]. Available: http://www.windowsazure.com/en-us/develop/net/fundamentals/intro-to-windows-azure/

[2] D. Chappell, Introducing the Azure services platform. White paper, Oct. 2008.

[3] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, in: Proceedings of Operating Systems Design and Implementation (OSDI), San Francisco, CA, 2004, pp. 137–150.

[4] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM 51 (2008) 107–113.

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

[6] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, C. Kozyrakis, Evaluating MapReduce for multi-core and multiprocessor systems, in: Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture, HPCA '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 13–24.

[7] B. He, W. Fang, Q. Luo, N.K. Govindaraju, T. Wang, Mars: a MapReduce framework on graphics processors, in: Proceedings of the 17thinternational conference on Parallel architectures and compilation techniques, PACT '08, ACM, New York, NY, USA, 2008, pp. 260–269.

[8] T. Gunarathne, T.L. Wu, J. Qiu, G. Fox, MapReduce in the Clouds for Science, Proceedings of CloudCom 2010 Conference, IUPUI Conference Center, Indianapolis. 2010

[9] M. Budiu, Y. Yu, A.B Irrell, D. Fetterly, Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks, European Conference on Computer Systems (EuroSys), Lisbon, Portugal, March 21-23, 2007

[10] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, J. Currey, DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language,Symposium on Operating System Design and Implementation (OSDI), San Diego, CA, December 8-10, 2008.

[11] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.H. Bae, J. Qiu, G. Fox, Twister: a runtime for iterative MapReduce, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10, ACM, New York, NY, USA, 2010, pp.810–818.

[12] Daytona, http://research.microsoft.com/en-us/projects/daytona/

[13] Y. Bu, B. Howe, M. Balazinska, M. D. Ernst, HaLoop: Efficient Iterative Data Processing on Large Clusters by In VLDB'10: The 36th International Conference on Very Large Data Bases, Singapore, 24-30 September, 2010.

[14] G. Malewicz, M. Austern, A. J.C. Bik, J. C. Dehnert, I. Horn, N. Leiser, G. Czajkowski, Pregel: a system for large-scale graph processing, Proceedings of the 2010 international conference on Management of data, ACM, New York, NY, USA, pp. 135-146

[15] D. Peng, F. Dabek, Large-scale Incremental Processing Using Distributed Transactions and Notifications, Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, USENIX (2010)

[16] R. Lee, T. Luo, Y. Huai, F. Wang, Y. He, X. Zhang, YSmart: Yet another SQL-to-MapReduce Translator. Proceedings of 31st International Conference on Distributed Computing Systems (Best Paper Award in ICDCS 2011), Minneapolis, Minnesota, June 20-24, 2011.

[17] Hive, http://hive.apache.org/

[18] J. Cohen, Graph Twiddling in a MapReduce World. Computing in Science & Engineering, July/August 2009, 29–41.

[19] Twister4Azure, http://salsahpc.indiana.edu/twister4azure/index.html

## BIOGRAPHIES

**Yi Pan** was born in Jiangsu, China. He entered Tsinghua University in March 1978 with the highest college entrance examination score among all 1977 high school graduates in Jiangsu Province. Currently, he is the chair and a professor in the Department of Computer Science and a professor in the Department of Computer Information Systems at Georgia State University. Dr. Pan received his B.Eng. and M.Eng. degrees in computer engineering from Tsinghua University, China, in 1982 and 1984, respectively, and his Ph.D. degree in computer science from the University of Pittsburgh, USA, in 1991.

Dr. Pan's research interests include parallel and cloud computing, wireless networks, and bioinformatics. Dr. Pan has published more than 140 journal papers with about 50 papers published in various IEEE/ACM journals. In addition, he has published over 150 conference papers and edited/authored 37 books. He has received many awards from organizations such as IEEE, NSF, AFOSR, JSPS, IBM, ISIBM, IISF and Mellon Foundation. Dr. Pan has served as an editor-in-chief or editorial board member for 15 journals including 6 IEEE Transactions and a guest editor for 10 special issues for 9 journals including 2 IEEE Transactions. He has organized numerous international conferences and workshops and has delivered over 20 keynote speeches at international conferences.

Dr. Pan is a "Great Master Face-to-Face" Series Speaker (2012), an IEEE Distinguished Speaker (2000-2002), a Yamacraw Distinguished Speaker (2002), a Shell Oil Colloquium Speaker (2002), and a senior member of IEEE. He is listed in Men of Achievement, Who's Who in Midwest, Who's Who in America, Who's Who in American Education, Who's Who in Computational Science and Engineering, and Who's Who of Asian Americans.

**Junbo Zhang** received his B.Eng. degree from Southwest Jiaotong University, China, in 2009. He is currently a Ph.D. candidate at the School of Information Science and Technology, Southwest Jiaotong University, China, supervised by Prof. Tianrui Li. He is also a visiting Ph.D. student at the Department of Computer Science, Georgia State University, USA, under advisor Prof. Yi Pan. He is interested in data mining, cloud computing, granular computing and rough sets. He is now a student member of ACM and CCF. Since 2009, he has published over 20 research papers in referred journals (*e.g.*, Information Sciences, International Journal of Approximate Reasoning, International Journal of Intelligent Systems), and conferences.

This page is intentionally left blank