A Programming Model for the Cloud Platform

Xiaodong Liu

School of Computer Engineering and Science
Shanghai University, Shanghai 200072, China
liuxiaodongxht@qq.com

Abstract

Programming models for cloud computing has become a research focus recently. Cloud computing promises to provide on-demand and flexible IT services, which goes beyond traditional programming models and calls for new ones. Some progress has been made in cloud computing programming models for large-scale data processing, but little was done on models of predictable performance. With the advantages on predictable performance, easily programming and deadlock avoidance, the BSP model has been widely applied in parallel databases, search engines, and scientific computing. This paper targets to adapt the BSP model into cloud environment. The scheduling of computing tasks and the allocation of cloud resources will be integrated into the BSP model. A BSPCloud programming model with predictable performance is proposed.

Keywords: *Programming Model; Cloud Computing; BSPCloud; Bulk Synchronous Parallel*

1. Introduction

Cloud computing integrates vast computing and/or storage resources together, which provides services on demand via networks. Developers request resources on demand and pay for it by hours. Developers can also increase or decrease resources according to their demand. Cloud computing provides convenience for application development and run. Meanwhile, it brings new challenges for cloud computing programming models. Research on cloud computing programming models has made some progress, such as Google's MapReduce [1] and Microsoft's Dryad [2]. However, there are still some issues to be further studied. Firstly, the current cloud computing programming models mainly focus on processing mass data. Computation intensive and I/O intensive application programming on cloud computing have been a new topic. Secondly, when one programming a cloud application, it's very important for the programmer to rely on a simple yet realistic cost model. Study on performance predictable cloud computing programming model is of great significance.

The Bulk Synchronous Parallel (BSP) [3] model is originally proposed by Harvard's Valiant. Its initial aim is to bridge parallel computation software and architecture. The advantages of BSP model are mainly on three aspects: Firstly, its performance can be predicted. Secondly, no deadlock occur when message passing. Thirdly, it is easy to program. Because of the above advantages, improvement models based on BSP were used in many programming environments. In order to program on heterogeneous environment, the authors of [4] propose a Heterogeneous Bulk Synchronous Parallel (HBSP) model. The authors of [5, 6]extend the BSP model through the migration to solve programming on heterogeneous grid environment. The authors of [7] apply the BSP model on parallel database and the authors of [8] apply BSP model on search engines.

This paper proposes a new cloud computing programming model BSPCloud, whose performance can be predicted. Our BSPCloud not only can be used for data intensive application but also can be used for computation intensive and I/O intensive application.

The BSP model communicates between any pairs of computing nodes. So, it can't exploit communication locality, which means a computing node does not communicate with all, and it only communicates with "adjacent" (*e.g.*, the same server or the same data center) nodes. Our BSPCloud Using hierarchical communication mechanism, which makes communication occur between adjacent nodes as far as possible. In order to facilitate the use of communication locality, this paper organizes computing node into a tree according to communication ability, which will be further discussed in session 2.1.

The novelty of BSPCloud can be summarized as follows:

- The BSPCloud programming model performance is predictable. The programmer can rely on a simple yet realistic cost model when designs a cloud computing application.
- The BSPCloud model adapts to a broad variety of application, for example, data intensive application, computation application, I/O intensive application, and it can be expanded.
- The BPSCloud uses hierarchical communication mechanism, which exploits communication locality. This paper also proposes a virtual resource tree according to communication ability.

The rest of the paper is organized as follows. Section 2 describes the BSPCloud programming model. Section 3 presents the BSPCloud performance cost model. Section 4 describes the related work. Section 5 concludes our work.

2. BspCloud Programming Model/Architecture

BSPCloud is a programming model for cloud computing, and it's goal is to provide a programming model whose performance can be predicted. The programmer can rely on a simple yet realistic cost model when designs a cloud computing program.

A schematic of the BSPCloud model organization is shown in Figure 1.

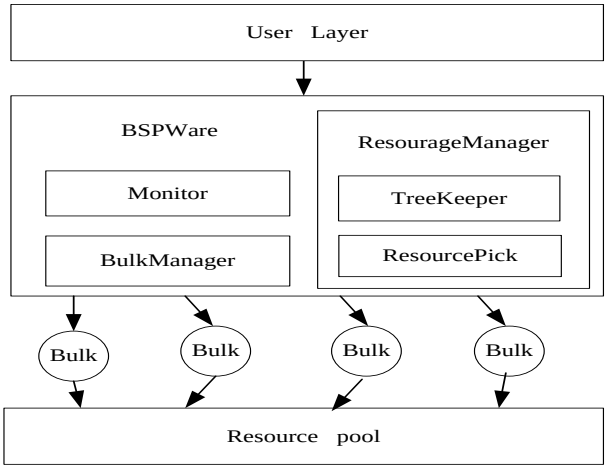

Figure 1. BspCloud Programming Model

BSPWare is the core of the system architecture, which is responsible for scheduling of computing tasks and allocation of cloud resources. Monitor is used to monitor the entire cloud platform resources. TreeKeeper is mainly used to construct and maintain resources tree. ResourcePick is responsible for selecting resources from virtual resource tree, which is used to participate in computation. BulkManager is the control center of

the application program, and it divides application program into many bulks which run parallel and it also responsible for fault tolerant.

2.1. Resources Organizational Strategy

In cloud computing environment, network bandwidth is relatively rarely. In order to make full use of network resources, our BSPCloud uses hierarchical mechanism. To motivate further discussion, let us take an example of cloud computing environment topological structure (as shown in Figure 2).

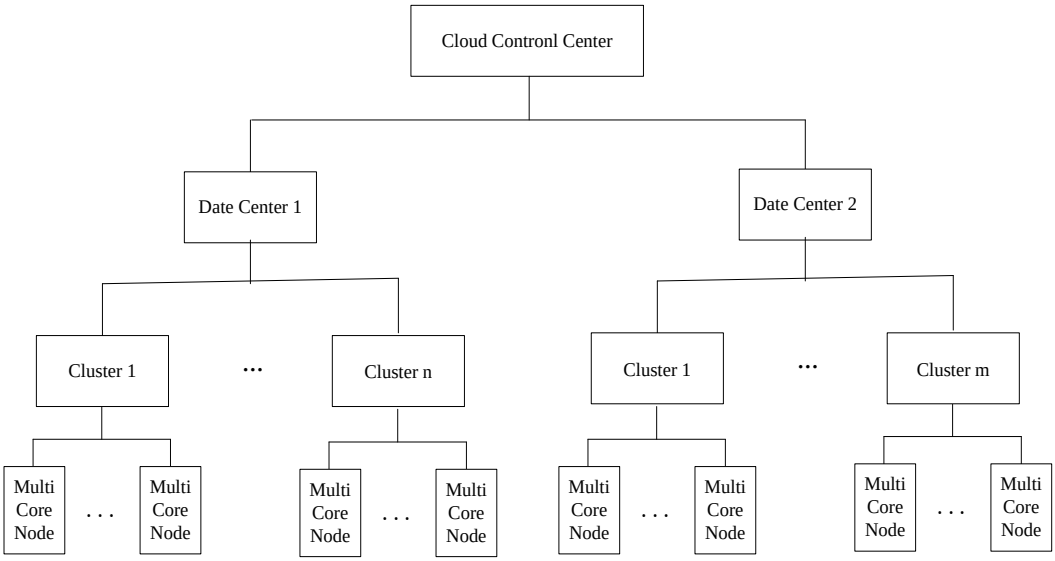

Figure 2. Cloud Computing Topological Structure

The above graph is an abstract from the reslistic cloud computing platform, and it has two data centers(data center 1 and data center 2), which locate in different regions, data center 1 has n computing clusters and data center 2 has m computing clusters, and each cluster is composed of many multi core nodes. Communication quality is different between nodes because of different locality. For example, communication between two nodes which locate in the same cluster and the same data center is faster than which locate different data center. In order to make full use of network resources, this paper organizes computing nodes as a virtual resource tree which is managed by TreeKeeper. Figure 3 shows dynamic changes of virtual resource tree. Each circle in this figure represents a computing node and rectangular represents control nodes which are used to manage computing nodes in a hierarchical manner. This paper signs busy nodes in black and idle nodes in white. When all computing nodes of one control node are busy, the control node is signed black. When users submit an application to cloud platform, ResourcePick selects computing resource for application from virtual resource tree.

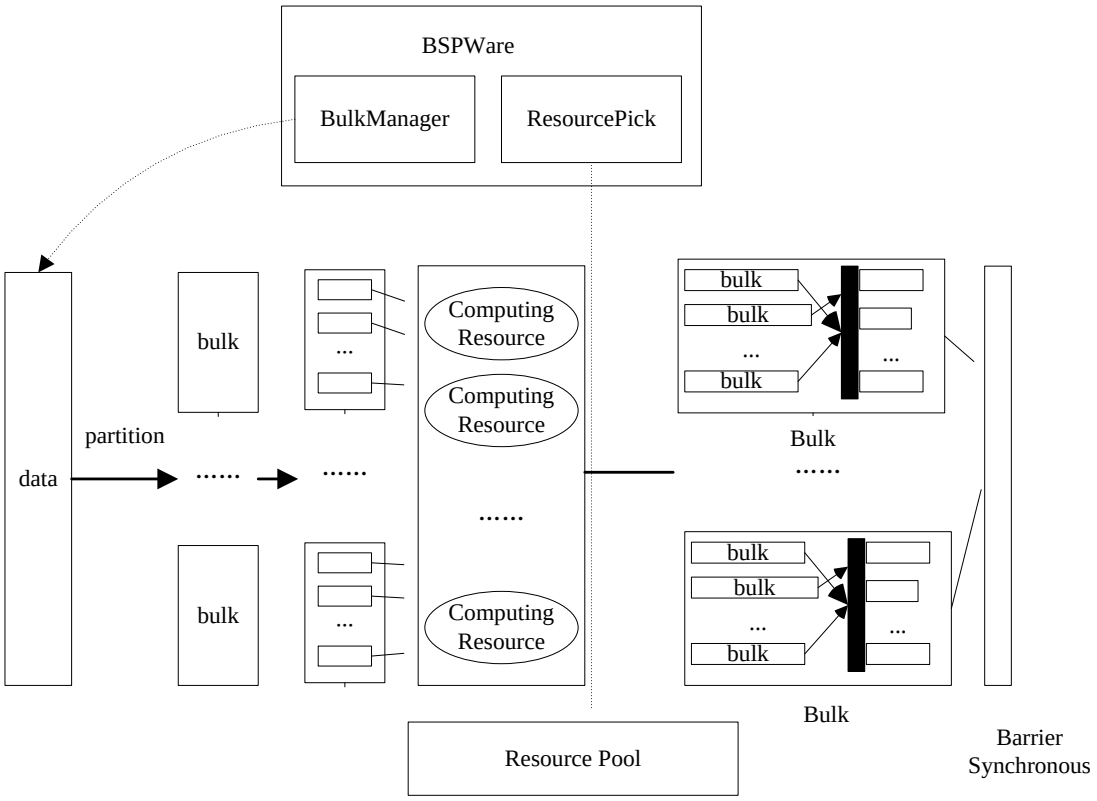

Figure 3. Execute Overview

2.2. Computing Tasks Partitioning Model

In cloud environment, the data are usually very large, and it unit is often chunk (*e.g.*, each chunk of Google's GFS is 64M). This paper assumes that data size which is needed to be processed is N, and divides tasks into n bulks $B(B_1, B_2 ... B_n)$. In the case data can be partitioned arbitrarily. This paper assumes partition $X(X_1, X_2 ... X_n)$ make load balance, and X_i is given as

$$X_i = (\alpha \times \frac{f_i}{\sum_{i=1}^{n} f_i} + \beta \times \frac{g_i}{\sum_{i=1}^{n} g_i}) \times N \quad (1)$$

where f_i is the frequency of B_i, g_i is the network through rate of B_i, α and β are the scale parameters of the computing phase and communicating phase.

Because data unit is often chunk in cloud environment, the above load balance is difficult to achieve. One way is to take the approximate method, but this method has a defect. Let us pick a simple example of a five bulks which is used to process 448M data and each chunk size is 64M. This paper assumes partition $(90,85,95,90,88)$ make load balance. However, if this paper uses approximate method, the first four bulks partition will be $(64,64,64,64)$, and the last bulk will have to be assigned 192M data.

In our BSPCloud model, this paper uses following data partition model:

Minimize $\| X - Y \|$ (2)

Subject to $\sum_{i=1}^{n} Y_i = N$ (3)

$Y_i \geq 0, \forall i$ (4)

The objective function given by Equation (2) is to find an optimization partition $Y(Y_1, Y_2 ... Y_n)$, and the constraints (3) are used to guarantee partition cannot exceed its total amount. The constraints (4) are used to guarantee each partition is not negative.

For make tasks partition be easy, this paper uses hierarchical tasks partitioning strategy. For example, data center assign computing tasks to its cluster according to computing ability of each cluster, and when the cluster receives the tasks, it assigns the tasks to computing nodes immediately.

2.3. Execution Overview

BulkManager automatically partition the input data into several bulks, which can be partitioned recursively. ResourcePick selects computing resources from the cloud platform. Bulks are mapped to resources by BulkManager. Bulks compute parallel and they communicate use hierarchical mechanism.

Figure 3 shows the overall flow of our BSPCloud. For simplicity, there are only two level bulks in Figure3. When the application program is submitted to the cloud platform,the following actions occur.

1) bulk partition

BulkManager partition the first level bulks $B_1, B_2 ... B_n$, and then the first level control node continue to partition the second level bulks (*e.g.*, B_1 is divided into $B_{11}, B_{12} ... B_{1m}$). The last layer bulks are leaf nodes (see in Figure 2), each bulk computes parallel.

2) resource select

ResourcePick selects resources from cloud platform according to application demands. Selected resources are organized into many resource bulks, and the number of which is equal to data bulks.

3) communication phas

BSPCloud makes communication occur between “adjacent” nodes as far as possible. It’s communication model uses hierarchical mechanism. Bulks only in the same level can communicate. The advantage of this is reducing communication overhead and improving the scalability.

4) fault tolerance policy

Since BSPCloud execution is composed of a set of super-steps, BSPCloud sets a checkpoint after each super-step. When error occurs, program needn’t executes from the beginning and it only needs to execute from the last checkpoint.

3. Cost Analysis

The BSPCloud model for level d will be specified by $(b_1, g_1, l_1, s_1), (b_2, g_2, l_2, s_2) ... (b_d, g_d, l_d, s_d)$. b_i is the bulk number of the i^{th} level, g_i is the network throughput rate of the i^{th} level, l_i is the time required for barrier synchronization of the i^{th} level, s_i is the number of super-step. For simplicity, the BSPCloud assumes the numbers of sub bulks of each bulk are same. The time cost can be decided by follow equation

$$T(1) = W_1 + M_1 \times g_1 + l_1 \quad (5)$$

$$T(d) = s_d \times \max_{1 \le i \le b_{d-1}} (T(d-1)) + M_d \times g_d + l_d \times s \text{ (d>1)} \quad (6)$$

$$W_1 = \sum_{i=1}^{s_1} \max_{1 \le i \le b_1} (\omega_i^{(s)}) \quad (7)$$

$$M_d = \sum_{s=1}^{s_d} \max_{1 \le i \le b_d} (m_i^{(s)}) \quad (8)$$

$\omega_i^{(s)}$ is processing time of the i^{th} bulk of super-step s, $m_i^{(s)}$ is network throughput of the i^{th} bulk of super-steps.

4. Related Work

Research on programming models for cloud computing has been hot in recent years. Google's MapReduce[1] hides the details of parallelization, fault tolerance, and load balance,. Users specify the computation in terms of map and reduce function, and the runtime system automatically parallelizes the computation across large-scale clusters of machines. Hadoop[9] is the open-source project of MapReduce. MapReduce two stage computation is rigid, and it only allows one input and one output, Microsoft's Dryad[2] is more flexible, and it allows input and output are arbitrary number, a Dryad job is a directed acyclic graph where each vertex is a program and edges represent data channels. There is some programming model s special on massive data processing, Yahoo's Pig Latin[10] and HadoopDB[11] combine high level declarative style of SQL and low-level procedural style of map-reduce. Google's Pregel [12] is a distribute programming model of graph processing which is based on BSP model. Hama[13] is a distribute parallel computing model which is based on Hadoop, it is processing graph use BSP model.[14] presents a parallel programming model on grid based on BSP.

5. Conclusions and Future Work

When one programming a cloud application, it is very interesting for the programmer to rely on a simple yet realistic cost model. In this paper, a cloud computing programming model which performance predictable is proposed, this paper calls it BSPCloud. BSPCloud adapts the BSP model into cloud environment. The scheduling of computing tasks and the allocation of cloud resources are integrated into the BSP mode. In order to exploit communication locality, BSPCloud commnication uses hierarchical mechanism, which means a computing node does not communicate with all, but only communicate with "adjacent" nodes. In the future, we will implement the programming model and deploy it in the cloud computing platform.

Acknowledgements

This work is supported by Innovation Action Plan supported by Science and Technology Commission of Shanghai Municipality (No.11511500200).

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters", Communications of the ACM, vol. 51, no. 1, **(2008)**, pp. 107-13.
[2] M. Iiard, M. Budiu and Y. Yuan, "Dryad: distributed data-parallel programs from sequential building blocks", Oper Syst Rev, vol. 41, no. 3, **(2007)**, pp. 59-72.
[3] L. G. Valiant, "A bridging model for parallel computation", Communications of the ACM, vol. 33, no. 8, **(1990)**, pp. 103-11.
[4] T. L. Williams, R. J. Parsons, "The Heterogeneous Bulk Synchronous Parallel model", ROLIM J. Parallel and Distributed Processing, Proceedings. Berlin; Springer-Verlag Berlin, **(2000)**.
[5] D. Righir, L. Pilla and A. Carissimi, "MigBSP: A Novel Migration Model for Bulk-Synchronous Parallel Processes Rescheduling", New York: Ieee, **(2009)**.
[6] O. Bonorden, "Load balancing in the bulk-synchronous-parallel setting using process migrations", IEEE International Parallel and Distributed Processing Symposium (IEEE Cat No07TH8938), **(2007)**, pp. 1-9.
[7] M. A. H. Hassan and M. Bamha, "Parallel processing of group-by join queries on shared nothing machines", F, Springer-Verlag New York Inc., **(2008)**.
[8] V. G. Costa, A. Prinrista and M. Marin, "A parallel search engine with BSP", F, IEEE, **(2005)**.
[9] Hadoop, [F]. http://hadoop.apache.org/.
[10] C. Olston, B. Reed and U. Srivastava, "Pig latin: a not-so-foreign language for data processing", F, ACM, **(2008)**.
[11] A. Abouzeid, K. Bajda-Pawlikowski and D. Abadi, "HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads", Proceedings of the VLDB Endowment, vol. 2, no. 1, **(2009)**, pp. 922-33.
[12] A. Malewicz. Gregorz, H. Matthew, J. C, Bik Aart, H. llan, L. Naty and C. Grzegorz, "Pregel: A system for large-scale graph processing, Proceedings of the 2010 International Conference on Management of Data, 2010, pp: 135-145.
[13] S. Sangwon, J. Yoon Edward, K. Jaehong, J. Seongwook, K. Jin-Soo and M. Seungryoul, "HAMA: An efficient matrix computation with the MapReduce framework", 2nd IEEE International Conference on Cloud Computing Technology and Science, **(2010)**, pp. 721-726.
[14] W. Tong, J. Ding and L. Cai, "A Parallel Programming Environment on Grid", Proc. of ICCS, LNCS, vol. 2658, no. 1, **(2003)**, pp. 225-234.