BSPCloud: A Hybrid Programming Library for Cloud Computing*

Xiaodong Liu†, Weiqin Tong and Yan Hou

Department of Computer Engineering and Science
Shanghai University, Shanghai, China
liuxiaodongxht@qq.com, wqtong@mail.shu.edu.cn, houyan_2008362@163.com

Abstract. Programming models for cloud computing has become a research focus recently. Current programming models for cloud computing mainly focus on improving the efficiency of the cloud platforms but little has been done on the performance predictability of models. In this paper, we introduce a programming model—BSPCloud. BSPCloud is a hybrid of distributed-memory and shared-memory parallel programming library. Coarse granularity tasks are completed by the distributed-memory model, and each coarse task is further divided into finer granularity tasks to exploit the shared-memory model. BSPCloud makes full use of the multi-core architecture which can improve the efficiency of the cloud platform. More importantly, the performance of BSPCloud is predictable. A proof-of-concept of BSPCloud library has been implemented in java. This paper introduces the model and implementation of the BSPCloud library and gives the experiment results of performance predictability and speedup.

Keywords: Programming model, Cloud computing, BSPCloud

1 Introduction

Increasing Internet business motivate server consolidation in data centers, which has become the foundation of cloud computing. Through virtualization-based cloud computing, multiple computers allow running as virtual machines (VM) in a single physical computer. With the ability to reduce hardware resource and provide on-demand service, cloud computing has become very popular. Traditional programming models can't adapt to cloud computing very well. In order to make full use of cloud computing resources, it is urgent to research on new programming models for the cloud computing platform. Some progresses have been made on cloud computing programming models [1, 2]. However, these programming models mainly

* This work is supported by Innovation Action Plan supported by Science and Technology Commission of Shanghai Municipality (No.11511500200) and the Ocean Public Welfare Project of The Ministry of Science and Technology (NO.201105033-4，NO.201105033-5)
† Corresponding author. (liuxiaodongxht@qq.com)

focus on processing massive data but little has done on the performance predictability of models. It's very important for the programmer to rely on a simple yet realistic cost model when one programming a cloud computing application. Thus research on programming models whose performance can be predicted is of great significance.

The Bulk Synchronized Parallel (BSP) [3] model has the advantages of performance predictability, easily programming and deadlock avoidance. Because of these advantages, the BSP model has been used in many programming environment [4, 5, 6]. In this paper, we propose a cloud computing programming model—BSPCloud, which is based on BSP model.

The execution of the BSP model is consisted of three ordered phases: computation, communication and synchronization. Each computing node first processes its local data, and then is global communication. After the communication phase, the globe synchronized occurs to assure that the data has been sent to the destination and the data can be used in next super-step. The original BSP Library is used to cluster composed of single core nodes. Nowadays, the computer architecture has changed a lot. Multiple processing cores on a single chip is very ubiquitous. The cloud computing data center is composed of thousands of multi-core nodes. If each core acts as a bulk, multiple bulks in the same processor will contend I/O in the commutation phase and lead to performance degrade. In order to take advantage of multi-core architecture characteristics, BSPCloud uses a hybrid of distributed-memory BSP and shared-memory BSP architecture. Coarse granularity tasks are completed by distributed-memory BSP and each computing node further divided tasks into finer granularity tasks to exploit the shared-memory BSP model. The novelty of BSPCloud can be summarized as follows:

- The performance of BSPCloud can be predictable, when one programming a cloud computing application, a simple yet realistic cost model can be relied on.
- BSPCloud uses a hybrid of distributed-memory and shared-memory architecture, which can make full use of the multi-core cluster architecture.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 describes the BSP programming model and explains why hybrid method is adopted in BSPCloud. Section 4 presents the BSPCloud programming model framework. Section 5 gives the experiment results. Section 6 concludes our work.

2 Related work

Google's MapReduce [1] is a programming model and an associated implementation for processing massive data. MapReduce hides the details of parallelization, fault tolerance and load balancing. Users only need to implement two functions: Map and reduce, MapReduce can automatically parallelize the computation across thousands of clusters of machines. The MapReduce has its own limitations. Its one input, two stage data flow is a little rigid and it is difficult to reuse and maintain. Microsoft's Dryad [2] is more flexible than MapReduce. A Dryad job is a directed acyclic graph where each vertex is a program and edges represent data channels. Concurrency arises from Dryad scheduling vertices to run simultaneously

on clusters of machines. Dryad allows arbitrary number of input and output and it is mainly applied in coarse-grain data-parallel applications.

Google's Pregel[7] is a distribute parallel programming model of graph processing which based on BSP model. Hama [8] is BSP computing framework on top of Hadoop Distributed File System (HDFS) for massive scientific computations such as matrix, graph and network algorithms. However, Pregel and Hama both provide BSP library based on their already existing framework.

3 BSP Model

The Bulk Synchronous Parallel (BSP) [3] model is first proposed by Harvard's valiant which is used to bridge parallel architecture and programming language. A computation of BSP model consists of a sequence of super-steps. The execution of each super-step is divided into three ordered phases: local computation, communication and barrier synchronization.

One of the most important advantages of BSP model is that its performance can be predictable. The cost model of BSP depends on three parameters: p, g and l. The number of processor/memory pairs is given by p, and g is defined as network throughput rate, and l is the global synchronization time. The time of each super-step can be expressed by the following formula:

$$T = \sum_{s=1}^{S} \max_{1 \le i \le p} \omega_i^s + \sum_{s=1}^{S} \max_{1 \le i \le p} h_i^s \times g + l \times s \quad (1)$$

Where the computing time of processor i is given by w_i, h_i is the data needs to be communicated and s is the total number of super-steps.

The original BSP Library is used to cluster composed of single core nodes. Nowadays, computer architecture has changed a lot and the multi-core processor is very ubiquitous. If each core acts as a bulk, the performance of BSP model will very poor. This is caused by the following reasons:

Firstly, multiple bulks in the same processor will contend I/O resource in the communication phase. Secondly, the synchronized time extends with the number of bulks increase.

In order to improve the performance of our BSPCloud, we use a hybrid of distributed-memory BSP and shared-memory BSP system. Each multi-core computing node acts as a bulk, and they communicate through network, which used to complete coarse granularity tasks. Each multi-core computing node further divides tasks into finer granularity tasks to exploit shared-memory computation.

4 BSP FrameWork

BSPCloud is a parallel programming model for cloud computing platform which combine distributed-memory BSP and shared-memory BSP. In this section, we describe BSPCloud architecture and its implements.

4.1 BSPCloud Architecture

The BSPCloud model uses hierarchical control and communication mechanism. As is shown in Fig. 2, BSPCloud is composed of a BspJobTracker and a set of BulkTrackers. The BspJobTracker has two components: schedule and control. Schedule is responsible for deciding when the BspJob is running. Control is responsible for splitting tasks and controlling it run. When the schedule selects a BspJob to run, control assigns it to a set of BulkTrackers and controls their running. The BulkTracker also has two components: control and monitor. Control assigns the BulkTracker's tasks into a set of bulk threads and monitor is used to monitor the status of the bulk thread. Bulk thread uses shared memory communication mechanism. When the bulk thread needs to communication with another bulk thread, it saves the data in memory. In the next super-step, the data can be obtained by another bulk thread. Communicate between BulkTrackers use the network. Hierarchical communication mechanism of BSPCloud can make full use of multi-core architecture, which can enhance speedup and scalability of the BSPCloud.

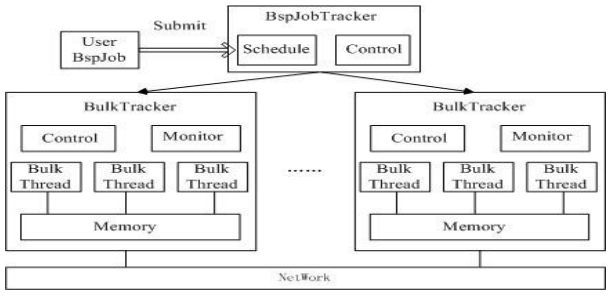

Fig. 1. BSPCloud system architecture.

4.2 BSPCloud Communicate Model and its Implements

To make full use of multi-core architecture, BSPCloud use a hybrid approach. In multi-core computer node, BulkTracker generate a set of bulk threads which used to perform finer granularity tasks, they communication occur in memory, which can avoid I/O and reduce barrier synchronized time. Communication among BulkTracker use network which used to perform coarse granularity tasks.

1) Distributed-memory communication and its implements

In our BSPCloud, each BulkTracker has a communication channel with others and they communicate through messing passing mechanism. We implement BSP message passing use java socket. Each BulkTracker has a communication thread which acts as a server socket. When a BulkTracker communicates with another BulkTracker, it creates a socket and sends the data to another BulkTracker through java socket.

2) Share-memory communication and its implements

In our BSPCloud, communication among bulk threads use shared-memory model. As is shown in figure 2, BSPCloud assigns a public memory area which every bulk thread can visit it. When bulk threads need communicate with other bulk threads, they put the data into public memory area. In the next super-step, these data can be got by other bulk threads.

We have implemented share-memory model in BSPCloud library, we defined a class Global as public store area, and in which a HashMap used is defined. The value of HashMap store the data need to communicate, the key of HashMap denote which bulk thread the data belong to. When a bulk thread communicates with another thread, it puts data and another thread's id in the HashMap, the data can be attained by another bulk thread through id in the next super-step.

4.3 BSPCloud synchronization

We have implemented synchronize operation with the same principle both distributed-memory and shared-memory. When synchronize operation is invoked, it sends a message to node 0 which acts as master node and waits for response from node 0. When node 0 received all synchronize message, it broadcasts next super-step message to all nodes, when compute nodes receive response message, and they can enter the next super-step.

5 Experiment evaluations

We have implemented BSPCloud in java and this section presents experimental results evaluting the performance predictable, speedup and scalability and the performance of BSPCloud affected by multi-core.

5.1 Experimental Setup

A set of experiments have been conducted by a virtualization cluster using three physical systems in our laboratory and the details of parameters are as follows:

host1: 8-core equipped with two 4-core Inter(R) Xeon(R) cpu E5620 running at 2.40GHz,32G memory.

host2:4-core AMD phenom Ⅱ X4 cpu B97 running at 3.20G Hz,10G memory

host3: 4-core Inter(R) i5 processor running at 3.10GHz ,4G memroy.

all system are running the 64-bit version of Ubuntu 10.04.4. the hosted virtualization is qemu-kvm-1.1. To evaluate the performance of BSPCloud , we implement matrix multiplication using the BSPCloud library.

5.2 Benchmarking and prediction

To estimate the parameters, a benchmarking program has been implemented in BSPCloud library. The results are shown in Table1.

Shared-memory parameters				Distributed-memory parameters			
P	r (Mflop/s)	g (Kflops)	l (ms)	P	r (Mflop/s)	g (Kflops)	l (ms)
1	72.5	0	0	1	69.2	0	0
2	71.3	0	3	2	68.4	639	14
4	68.1	0	6	4	68.1	632	36
8	67.1	0	8	8	67.9	625	68

Table 1 Benching Parameters

To evaluate the time cost predictable, we computing the matrix multiply C=A×B, where the size of matrix A and B both 4000×4000. We use shared-memory(SM) model and distributed-memory (DM) model respectively. For SM model, we apply three VMs. the cores are 2, 4 and 8 respectively and we compute using 2 threads, 4 threads and 8 threads respectively corresponding. For DM model, we apply 8 VMs and each has one core. We compute using 2 VMs, 4 VMs, and 8 VMs respectively. We compare the computing results with the theoretical values computing in equation (1). Fig 2 shows the results.

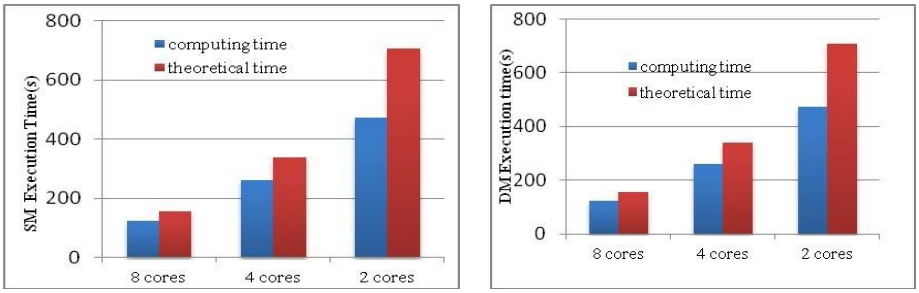

Fig.2. Time cost predictable evaluate using shared-memory and distributed-memory

BSPCloud does not make very accurate prediction on the running time. This is because some factors have been overlooked such as data load time, connection creating time and VM schedule.

5.3 BSpeedup and Scalability

We also evaluate the performance of BSPCloud. We fix the size of matrix A and B, they are both 4000×4000, and then we scale the number of processor cores. For SM model, we use 2 threads, 4 threads and 8 threads respectively. For DM model, we use 2 VMs, 4 VMs and 8 VMs respectively. For hybrid model we use one VM, 2 VMs and 4 VMs and each has two cores. We create one thread in each core in all our experiments. Figure 3 shows the results. When apply one VM which has one core in an 8 core computer and only it runs, the performance is very bad because the core schedule and cache switch, so the speedup of BSPCloud is very high.

We also fix the number of cores are 8 and scale the size of matrix used three model. The results are show in figure4.

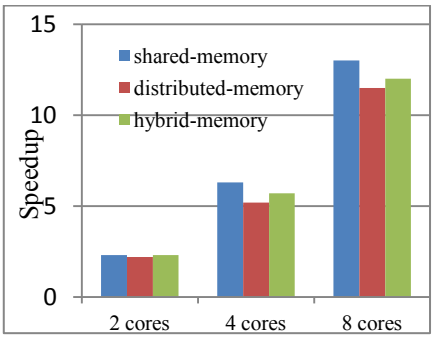

Fig. 3. Speedup of BPCloud.

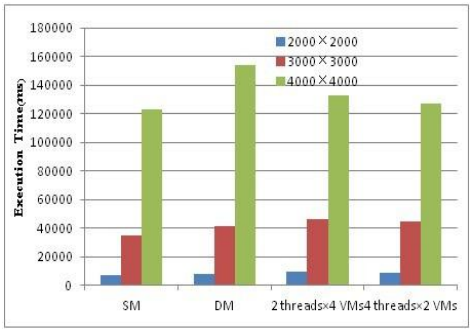

Fig.4. Vary the size with fix cores

6 CONCLUSIONS

In this paper, we propose a programming model for cloud computing environment—BSPCloud. BSPCloud is a hybrid of distributed-memory model and shared-memory model system. BSPCloud can make full use of multi-core clusters and has the advantage of performance predictability. The experiments results show that the cost time of BSPCloud can be predictable. We also give the speedup of BSPCloud.

References

1. Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters [J]. Communications of the ACM, 51(1): 107-13.(2008)
2. Iiard M, Budiu M, Yuan Y, et al. Dryad: distributed data-parallel programs from sequential building blocks [J]. Oper Syst Rev, 41(3): 59-72.(2007)
3. Valiant L G. A bridging model for parallel computation [J]. Communications of the ACM, 33(8): 103-111.(1990)
4. Righir D, Pilla L, Carissimi A, et al. MigBSP: A Novel Migration Model for Bulk-Synchronous Parallel Processes Rescheduling [M]. New York: IEEE, 2009.
5. Hassan M A H, Bamha M. parallel processing of" group-by join" queries on shared nothing machines, F, [C]. Springer-Verlag New York Inc.(2008)
6. Costa V G, Prinrista A, Marin M. A parallel search engine with BSP, F, [C]. IEEE.(2005)
7. Malewicz. Gregorz, Austern, Matthew H., Bik Aart J.C, Horn llan, Leiser Naty, Czajkowski Grzegorz, Pregel: A system for large-scale graph processing, Proceedings of the 2010 International Conference on Management of Data, pp: 135-145.(2010)
8. Seo Sangwon, Yoon Edward J., Kim jaehong, Jin Seongwook, Kim Jin-Soo, Maeng Seungryoul, HAMA: An efficient matrix computation with the MapReduce framework, the 2nd IEEE International Conference on Cloud Computing Technology and Science, pp: 721-726.(2010)