2013 IEEE 12th International Symposium on Network Computing and Applications

A Hierarchical Distributed Loop Self-Scheduling Scheme for Cloud Systems

Yiming Han and Anthony T. Chronopoulos

Department of Computer Science
University of Texas at San Antonio
San Antonio, TX, USA
Email: yhan@cs.utsa.edu, atc@cs.utsa.edu

***Abstract*—Cloud systems have demonstrated the powerful computation and storage capability in many scientific applications. In this paper, we propose a hierarchical distributed loop self-scheduling scheme to achieve good load balancing by applying weighted self-scheduling scheme on a heterogeneous cloud system. This scheme also considers the distribution of the output data, which can help reduce communication overhead. We evaluated the scheme with two scientific applications: Matrix Multiplication and Quick Sort. The results shows that our schemes achieve better load balancing and better overall performance than standard loop self-scheduling scheme.**

***Keywords*-Self-Scheduling; Distributed; Hierarchical; Cloud System.**

I. Introduction

Scientific loops are usually computation-intensive which may take a long execution time. Distributed systems, such as cluster, grid and cloud, are widely used in many scientific loops. Thus, scientific loop paralization, which schedules and assigns work among processors/workers, becomes an important issue. One of the difficult problems is load balancing. Efficient loop scheduling schemes can improve the utilization of resources and minimize the total execution time.

Cloud computing is emerging as a powerful technology to meet the requirements for high-performance computing and massive storage. It provides scalable, flexible, reliable and on demand computing and storage resources over a network. Many scientific computation-intensive and data-intensive applications are accomplished on cloud systems [1] [2]. a cloud system could be considered as a dynamic heterogeneous distributed system. A cloud system may also provide a homogeneous computing environment at the start. However, it may be upgraded and replaced to exhibit more heterogeneity [3]. The availability and performance of virtual machines can change over time. Also a cloud system is transparent to cloud users, which means cloud users still perceive it as a homogeneous environment. Thus, it is likely to create load imbalance if we ignore the heterogeneity. Previous research reported some schemes on a heterogeneous cluster and grid systems. Also, [4] tested a distributed scheme for cloud systems.

There are some commercial cloud providers, such as Amazon EC2, Microsoft Azure, Salesforce Service Cloud and Google Cloud. Some open source cloud projects for research and development also exist, for example, OpenStack, Eucalyptus, CloudStack and Ganeti [5] and references there in. There is also much ongoing research for cloud systems. In [6], a provisioning technique that automatically adapts to workload changes related to applications with Quality of Services (QoS) in large, autonomous, and highly dynamic environments is proposed. [7] extends Grid workflow middleware to compute clouds in order to speed up executions of scientific workflows. A hierarchical scheduling algorithm for applications, to minimize the energy consumption of both servers and network devices is proposed in [8]. The problem of provisioning physical servers to a sequence of jobs, and reducing the total energy consumption is studied in [9]. In [10], a Master-Worker model is used for a case study of an application of a parallel simulation optimization deployed on a private Cloud. [11] reported that the effect of some critical parameters (allocation percentages, real-time scheduling decisions and co-placement) on the performance of virtual machines. The performance of cloud computing services for scientific computing workloads is studied in [12]

Cloud computing platforms provide computing service to users by virtualization technology [13] [14]. For high performance computing applications, we can use cloud to virtualize clusters on cloud systems. These virtual machines can share the same physical hardware or different physical hardware with various system load and user load and cloud system use a fair-share balancing algorithm that gives equal time to each virtual machine. However, because of limited resources, the virtualized cluster is not private and the resources are shared by many users, which means the virtualized cluster may act as a heterogeneous computing environment at running time. Thus, the heterogeneity should be taken into account to improve resource utilization and reduce load imbalance. MapReduce [15] is a general concurrent programming framework for scheduling job-tasks on cloud systems. Previous research [16] [17] reported that the performance on virtual machines is lower than the physical system.

The rest of the paper is organized as follows. In section II, we describe the hierarchical distributed schemes. In section

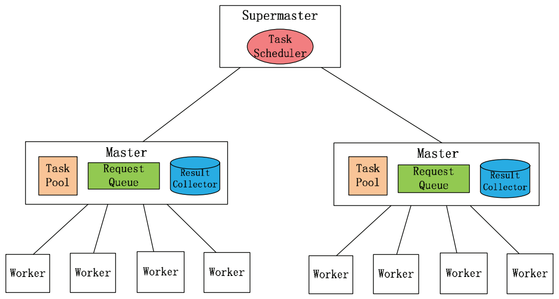


Figure 1. Hierarchical Architecture

III, experiments and results are presented. In section IV, conclusions are drawn.

II. Hierarchical distributed schemes

We consider a logical hierarchical architecture as a good model for scalable systems and we propose a new hierarchical approach for addressing the bottleneck problems in the Master-Worker schemes.

Instead of making one master process responsible for all the workload distribution, several master processes are introduced. Thus, the hierarchical structure contains a lower level, consisting of worker processes, and several superior levels, of master processes. On top, the hierarchy has an overall *supermaster*. The workers' role is to perform the computations following a Master-Worker self-scheduling method for the problem that is to be solved. This scheme is called a *Hierarchical Distributed Scheme*.

Figure 1 shows this design for two levels of master processes, one supermaster and two master nodes. The task scheduler resides in the supermaster and it uses distributed scheduling schemes (DTSS/DFSS/DGSS) [4] [18] to compute small scheduled chunks for each master node and send to master nodes' Task Pools. When the Task Pool of a master node is empty, it asks for more work (from the supermaster) in order to fill the Task Pool until there is no more work. The master node accepts a worker request, places it into the request queue and gets a scheduled chunk from the Task Pool and serves the top request from Request Queue. Also, the master node is in charge of gathering the computed results from workers. There are multiple Request Queues and Result Collectors distributed in different master nodes, which can share the responsibilities.

III. Evaluation

A. Cloud Environment

We use FlexCloud of Institute for Cyber Security(ICS) at University of Texas at San Antonio. The ICS FlexCloud is one of the first dedicated Cloud Computing academic research environments. It offers significant capacity and similar design features found in Cloud Computing providers, including robust compute capability and elastic infrastructure design.

B. Experimental Setup

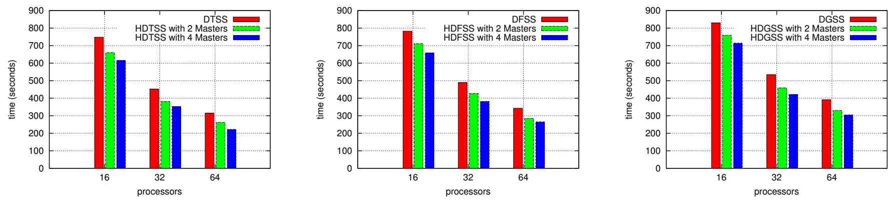
Two applications, Quick Sort and Matrix Multiplication are used to evaluate the overall performance. Quick Sort has 20K lines of random arrays and the size of Matrix Multiplication is 15K * 15K.

We use 5 different physical machines are on FlexCloud. We created 16 VMs on each physical machine sharing the same LAN. Each VM corresponds to a separate core. The purpose is showing the network heterogeneity in the experiment. The communication overhead between VMs in the same physical machine (intra-node shared memory communication) is lower than in different physical machines (inter-node distributed memory communication). For 64 workers using 4 masters hierarchical distributed scheme, each master has 16 workers. Master VM and its 15 worker VMs are in the same physical machine and the other worker VM is in another physical machine. The most of work distribution communication and the results collection communication is intra-node shared memory communication, instead of communication across nodes. The result collection communication work is distributed in masters, instead of in a single master node by standard scheme. Each VM is loaded with Ubuntu Linux 12.04 image. Stress [19], a work generator, is used to create a heterogeneous computing environment. Stress is a deliberately simple workload generator. Stress was developed by University of Oklahoma. It imposes a configurable amount of CPU, memory, I/O, and disk stress on the system. Each worker can get work proportional to its available computing power. The supermaster VM resides on the 6th physical machine from the masters and workers. This machine has a large memory and we used no 'Stress' load because we want to minimize the scheduling overhead.

C. Results

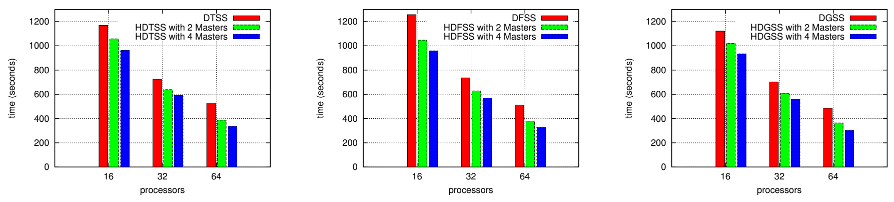
The following loop scheduling schemes are implemented. distributed schemes: DTSS, DFSS, DGSS; hierarchical distributed schemes: HDTSS, HDFSS, HDGSS. All the schemes are implemented by C++ and MPI. All timings are in seconds.

Figure 2 and Figure 3 present the execution times of non-hierarchical distributed schemes (DTSS, DFSS, DGSS) and hierarchical distributed schemes(HDTSS, HDFSS, HDGSS) on 20K Quick Sort and 15K * 15K Matrix Multiplication. It can be observed that the hierarchical distributed schemes show substantial performance improvement over non-hierarchical distributed schemes. For TSS scheme, HDTSS(2 masters) and HDTSS(4 masters) are 18% and 33% faster(in average) than DTSS respectively in Quick Sort. And HDTSS(2 masters) and HDTSS(4 masters) are 24% and 38% faster than DTSS respectively in Matrix Multiplication. For FSS scheme Quick Sort, HDFSS(2 masters) and HDFSS(4 masters) are 16% and 26% better than DFSS respectively. And for Matrix Multiplication using DFSS, the improvements are 26% and 43%. For GSS scheme, improvements



a. $DTSS$ & $HDTSS$ b. $DFSS$ & $HDFSS$ c. $DGSS$ & $HDGSS$

Figure 2. The total execution time for Quick Sort using non-hierarchical distributed and hierarchical distributed schemes



a. $DTSS$ & $HDTSS$ b. $DFSS$ & $HDFSS$ c. $DGSS$ & $HDGSS$

Figure 3. The total execution time for Matrix Multiplication using non-hierarchical distributed and hierarchical distributed schemes

are similar. The reason for this improvement is that hierarchical distributed schemes distribute work into multiple master nodes, which decentralizes the work distribution and the collection of the results. Thus, with more master nodes, there is better work load balancing.)

We next analyse the total execution time in terms of the master time, the communication time and overhead time. Let T_{master} denote the total execution time of a master, which means that the workers managed by this master have finished all the work assigned to them and the results have been returned to the master. We note that, $T_{exec} = max\{T_{master_1}, T_{master_2}, \ldots, T_{master_m}\} + T^{'}$, where T_{exec} denotes the total execution time (measured by the supermaster) for m masters hierarchical distributed scheme and where $T^{'}$ is the time for scheduling, work distribution, start up and termination overheads in the supermaster. Thus, T_{master} represents most of the work execution time in the experiment, because the scheduling overhead in the supermsater is low. Thus, the load balancing depends on both computation in workers and the communication time to return the results to the masters. We use the maximum master times difference, $T_{diff} = max\{T_{master_1}, T_{master_2}, \ldots, T_{master_m}\} - min\{T_{master_1}, T_{master_2}, \ldots, T_{master_m}\}$, to measure the work load balancing in the experiment. If T_{diff} is small, the major work is distributed evenly and the utilization is better. Figure 4 and Figure 5 present the T_{diff} for non-hierarchical distributed schemes (DTSS, DFSS, DGSS) and hierarchical distributed schemes(HDTSS, HDFSS, HDGSS). For non-hierarchical distributed schemes, T_{diff} is the same as T_{master}. It can be observed that the differences in the case of non-hierarchical distributed schemes are quite substantial. The work is centralized using the single master and the communication and synchronization overhead is high. On the other hand, in the case of hierarchical distributed schemes, the results collection is distributed among several masters. Thus T_{diff} is small and the work load is more balanced.

IV. Conclusion

In this paper, we studied hierarchical distributed loop scheduling schemes and compared them to standard schemes by implementing in a cloud computing environment. The hierarchical distributed loop scheduling schemes reduce the communication time for returning results and the work distribution time in the cloud system environment. MapReduce is a programming model which offers an alternative to MPI implementation of many data parallel applications. In the future, we plan to implement our schemes in MapReduce and compare to MPI for scientific loops.

Acknowledgement

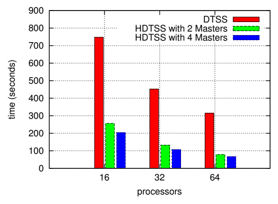
We gratefully acknowledge the following: (i) support by NSF grant (HRD-0932339) to the University of Texas at San Antonio; and (ii) time grants to access the facilities of Institute for Cyber Security(ICS) of University of Texas at San Antonio and FutureGrid at Indiana University, Bloomington.

References

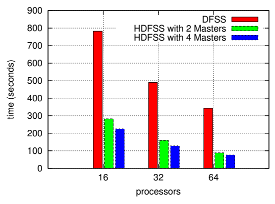
[1] K. Keahey, "Cloud computing for science," *Proceeding of 21st Scientific and Statistical Database Management Conference*, vol. 5566, pp. 478–478, 2009.

[2] H. Qian, H. Zu, C. Cao, and Q. Wang, "Css: Facilitate the cloud service selection in iaas platforms," *Proceeding of IEEE International Conference on Collaboration Technologies and Systems*, 2013.

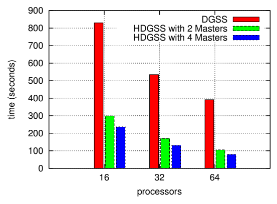
[3] S. Yeo and H.-H. Lee, "Using mathematical modeling in provisioning a heterogeneous cloud computing environment," *Computer*, vol. 44, pp. 55–62, 2011.



a. $DTSS$ & $HDTSS$

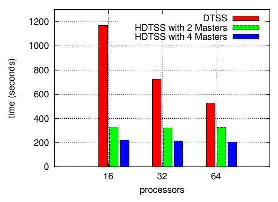


b. $DFSS$ & $HDFSS$

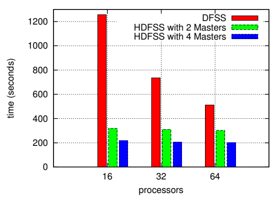


c. $DGSS$ & $HDGSS$

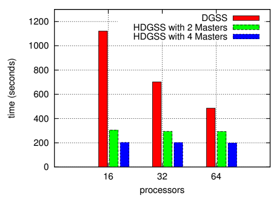
Figure 4. The maximum difference in masters execution time for Quick Sort using non-hierarchical distributed and hierarchical distributed schemes



a. $DTSS$ & $HDTSS$



b. $DFSS$ & $HDFSS$



c. $DGSS$ & $HDGSS$

Figure 5. The maximum difference in masters execution time for Matrix Multiplication using non-hierarchical distributed and hierarchical distributed schemes

[4] Y. Han and A. Chronopoulos, "Distributed loop scheduling schemes for cloud systems," *23th IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW'13*, pp. 20–26, 2013.

[5] K. Hwang, J. Dongarra, and G. C. Fox, "Distributed and cloud computing: From parallel processing to the internet of things," *Morgan Kaufmann*, 2011.

[6] R. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and Qos in cloud computing environments," *International Conference on Parallel Processing, ICPP'11*, pp. 295–304, 2011.

[7] S. Ostermann, R. Prodan, and T. Fahringer, "Extending grids with cloud resource management for scientific computing," *2009 10th IEEE/ACM International Conference on Grid Computing*, pp. 42–49, 2009.

[8] G. Wen, J. Hong, C. Xu, P. Balaji, S. Feng, and P. Jiang, "Energy-aware hierarchical scheduling of applications in large scale data centers," *International Conference on Cloud and Service Computing, CSC'11*, pp. 158–165, 2011.

[9] Y.-C. Hsu, P. Liu, and J.-J. Wu, "Job sequence scheduling for cloud computing," *Proceedings of the 2011 International Conference on Cloud and Service Computing*, pp. 212–219, 2011.

[10] G. V. Mc Evoy, B. Schulze, and E. L. M. Garcia, "Performance and deployment evaluation of a parallel application on a private cloud," *Concurrency and Computation:Practice and Experience*, pp. 2048–2062, 2011.

[11] T. C. Kousiouris, George and T. Varvarigou, "The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks," *Journal of Systems and Software*, pp. 1270–1291, 2011.

[12] A. Iosup, S. Ostermann, M. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed Systems*, pp. 931–945, 2011.

[13] J. E. Simons and J. Buell, "Virtualizing high performance computing," *ACM SIGOPS Operating Systems Review*, pp. 136–145, 2010.

[14] H. Qian, C. Cao, L. Liu *et al.*, "Exploring the network scale-out in virtualized servers," *Proceeding of International Conference on Soft Computing and Software Engineering*, 2013.

[15] W.-C. Shih, S.-S. Tseng, and C.-T. Yang, "Performance study of parallel programming on cloud computing environments using MapReduce," *International Conference on Information Science and Applications, ICISA'10*, pp. 1–8, 2010.

[16] J. Ekanayake and G. Fox, "High performance parallel computing with clouds and cloud technologies," *Proceedings of the first International Conference on Cloud Computing*, pp. 20–38, 2010.

[17] C. Evangelinos and C. N. Hill, "Cloud Computing for parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2," *5th International Conference on Computability and Complexity in Analysis*, pp. 159–168, 2008.

[18] Y. Han and A. Chronopoulos, "Scalable loop self-scheduling schemes implemented on large-scale clusters," *23th IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW'13*, 2013.

[19] Stress, http://www.hecticgeek.com/2012/11/stress-test-your-ubuntu-computer-with-stress/.