Course 7001 Mini Project
Performance Evaluation of Hadoop on Virtual Machines

Yuzhe Tang
902641656

1. Introduction

MapReduce[1] is a popular programming framework that is intended for automatical paralellization of computation in the cloud. MapReduce deals with data intensive applications; huge amount of data is first loaded from remote DFS, then copied as intermediate results from Mapper to Reducer, and finally written back to DFS. Along with this large amount of data transfer, many I/O operations are incurred across MapReduce instances.

MapReduce runtime usually operates in a data center environment, such as Amazon EC2, where virtual machine instances, rather than physical machines, are exposed. Virtual machines have great advantages on isolation of failed instances, server consolation for better power utilization. However, they also hide from upper-level applications the data locality on physical networks, introducing unnecessary I/O operations. More importantly, I/O operations in the virtualized architecture incur extra system overhead[2], which may not be feasible for data-intensive applications like MapReduce.

In this report, we study the performance harm of MapReduce on virtual machines, and try to tune the virtualized system to minimize such degradation. We are aware of recent research work on addressing the same problem [3]. The authors also come up with physical level-aware MapReduce framework [4], that locally combine intermediate data before shuffling them across different machines. Nexus[5], [6] is an even more ambitious framework that explores new interface between MapReduce-based applications and virtualized substrates; the substrates should expose flexible interface to enable application-specific scheduling, while providing virtualization. As such, physical data locality is exposed and can be used to achieve better IO performance. All the above research is based on the assumption that performance degradation of MapReduce on VMs are unacceptable. In this report, we conduct experiments on Xen-based virtualization environment and try to quantify such performance decrease. We also tune the system and provide a more suitable configuration that lead to better performance.

2. Experimental Evaluation

We present our evaluation results in this section. Our main goal is to quantify performance degradation of imposing MapReduce on virtual machines.

2.1. Setup

Our preliminary experiments are based on a single machine with Xen installed. The machine is equipped with two single-core 2.66GHz Intel(R) Duo processors, 2GB of memory and 250GB of disk. In VM-based environment, we use Xen release "2.6.26-2-xen-amd64", and run Ubuntu 8.04 as virtual machines. We only use domU as tested VMs, and install hadoop 0.20.1 on each of them. In evaluation, the hadoop is configured as in table 1. The reason we choose not to replicate data is in order to increase the cross-domain IO. The benchmark used is the standard "word count" program, which is available in hadoop distribution. The tested dataset contains files of plain text, with total size being 160MB, and which are also used in [7].

2.2. Evaluation Results

In the experiment, we issue the "word count" program into the system three times and report the average running time as results. In each run, we reformat and re-initialize the Hadoop Filesystem, because it seems Hadoop utilizes caching to interfere performance[1]. The results are illustrated in Figure 1, in which the running time first decreases and then increases, as the number of virtual machines goes up. The turning point is at 4 virtual machines. Note that in our system, the default number of Mapper is configured also at value 4. Before the turning point, running time decreases because the more virtual machines means more resources, and Mappers could be scheduled to dedicated VMs, rather

1. In the test, the execution time of 1st run and 10th run of the same program under identical configuration could be significantly different.

dfs.replication	1	replication level
mapred.map.tasks	4	default number of mapper
mapred.reduce.tasks	2	default number of reducer

Table 1: Basic Hadoop Configuration

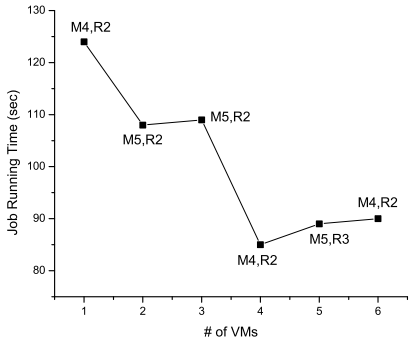

Figure 1: Running time under different number of VMs on the same physical machine. In the diagram, Mi, Rj means there are i mapper and j reducer in the run.

than share a single VM. For instances, when there is only one VM, all mappers are scheduled to this VM, which increases the running time. After number of VMs pass the turning point, Mappers becomes less than VMs. As VMs grow more, there are more vacant VMs that occupy the resources like CPU, which could otherwise be used for VM with Mapper/Reducer tasks. As such, running time could go up, (though slowly). In the whole process, note that the number of Mapper and Reducer basically remain constant, implying total number of processes do not change (though there are slight changes, which is due to Hadoop scheduler.)

3. Related works

MapReduce has gained a great deal of popularity in both research and industrial community.

MapReduce on Different Substrates. MapReduce is originally intended for data center environment, a typical shared-nothing architecture. Recently, several research has been seen to extend MapReduce to other substrate, including CMP-based shared-memory[7], GPU[8], and virtual machines[3], [4].

System Design of MapReduce. Following the original MapReduce framework, several improvement has been made for better performance in different settings. [9]proposes a new scheduler that uses speculative execution to fit MapReduce better in heterogeneous environment. While MapReduce is originally for batch processing, [10] targets at interactive query processing and proposed online MapReduce. [11] addresses the problem of lack of data locality in existing MapReduce framework, and present a new job scheduling algorithm to tackle it. Energy efficiency of MapReduce are discussed in [12].

MapReduce in Different Application Scenarios. MapReduce is a simplified programming model for parallel data intensive applications[1]. Recently, several research efforts have been conducted to apply MapReduce in more complicated scenarios, such as machine learning [13] and databases[14], [15].

4. Implications and Future Work

The experiments need to be more complete, and next steps are to conduct evaluation on different physical machines and under specific conditions. Besides that, the preliminary experiment in this report shows that performance could vary as clients initiate their clusters with different number of VM instances. Because virtualization blinds outsider from knowing about the underlying resources, it poses difficulty for both applications to optimize performance and clients to better config. A typical question would be **"How many instances should a EC2 user initiate, in order to achieve desired performance?"**. A possible solution would be to speculatively execute small sample programs, and as such guide clients to make decision based on the snooped performance. Solutions of this kind are more practical than recent work [5], [6], [3] which either need significantly modify the Hadoop code, nor build a counterpart from scratch.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI*, 2004, pp. 137–150.

[2] L. Cherkasova and R. Gardner, "Measuring cpu overhead for i/o processing in the xen virtual machine monitor," in *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2005, p. 24. [Online]. Available: http://portal.acm.org/citation.cfm?id=1247360.1247384

[3] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi, "Evaluating mapreduce on virtual machines: The hadoop case," in *CloudCom*, 2009, pp. 519–528.

[4] S. Ibrahim, H. Jin, B. Cheng, H. Cao, S. Wu, and L. Qi, "Cloudlet: towards mapreduce implementation on virtual machines," in *HPDC*, 2009, pp. 65–66.

[5] B. Hindman, A. Konwinski, M. Zaharia, and I. Stoica, "A common substrate for cluster computing," in *Proceedings of USENIX HotCloud'09: Workshop on Hot Topics in Cloud Computing*, San Diego, CA, 06/2009 2009.

[6] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. Joseph, S. Shenker, and I. Stoica, "Nexus: A common substrate for cluster computing," Submitted. [Online]. Available: http://cs.berkeley.edu/ andyk/nexus.pdf

[7] C. Ranger, R. Raghuraman, A. Penmetsa, G. R. Bradski, and C. Kozyrakis, "Evaluating mapreduce for multi-core and multiprocessor systems," in *HPCA*, 2007, pp. 13–24.

[8] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: a mapreduce framework on graphics processors," in *PACT*, 2008, pp. 260–269.

[9] *Improving MapReduce Performance in Heterogeneous Environments*. San Diego, CA: USENIX Association, 12/2008 2008. [Online]. Available: http://dblp.uni-trier.de/db/conf/osdi/osdi2008.html#ZahariaKJKS08

[10] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "Mapreduce online," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-136, Oct 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-136.html

[11] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Job scheduling for multi-user mapreduce clusters," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-55, Apr 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-55.html

[12] Y. Chen, L. Keys, and R. H. Katz, "Towards energy efficient mapreduce," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-109, Aug 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-109.html

[13] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," in *NIPS*, 2006, pp. 281–288.

[14] H. chih Yang, A. Dasdan, R.-L. Hsiao, and D. S. P. Jr., "Map-reduce-merge: simplified relational data processing on large clusters," in *SIGMOD Conference*, 2007, pp. 1029–1040.

[15] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in *SIGMOD Conference*, 2008, pp. 1099–1110.