

Towards Energy-Efficient Management of MapReduce Workloads

Jordà Polo, Yolanda Becerra, David Carrera, Vicenç Beltran, Jordi Torres and Eduard Ayguadé
jorda.polo@bsc.es
Barcelona Supercomputing Center (BSC) - Technical University of Catalonia (UPC)
Barcelona, Spain

1. INTRODUCTION

Data center computing has dramatically transformed the way many critical services are delivered to customers, and at the same time has posed new challenges to data center infrastructures. The result is a complete new generation of large scale infrastructures, bringing an unprecedented level of workload and server consolidation, that demand new programming models, management techniques and hardware platforms.

While MapReduce was originally used primarily for batch data processing, it is now also being used in shared, multi-user environments in which submitted jobs may have completely different priorities: from small, almost interactive, executions, to very long programs that take hours to complete. This change makes task scheduling, which is responsible for selecting tasks for execution across multiple jobs, even more relevant. Task selection and slave node assignment govern a job's opportunity to make progress, and thus influences job performance. But it also brings new opportunities to control the energy consumption of MapReduce cluster through the management of jobs' performance, as well as task and data consolidation.

In this paper we present our plans to leverage the Adaptive Scheduler [9] for Hadoop to address the problem of managing energy efficiency in multi-job and multi-user MapReduce clusters. The current version of Adaptive Scheduler provides automatic performance management of MapReduce workloads that is driven by the high-level performance objectives defined by the users. It was developed on top of Hadoop [3], and is downloadable at [1].

2. TECHNICAL BACKGROUND

MapReduce is a framework originally designed by Google to exploit large clusters to perform parallel computations. It is based on an implicit parallel programming model that provides an easy and convenient way to express certain kinds

of distributed computations, particularly those that process large data sets. Hadoop [3] is a state of the art MapReduce runtime and the supporting distributed file system (HDFS [2]) that helps with the distribution of tasks and data across nodes. The runtime and the distributed file system also provide fault tolerance and reliability, which are crucial in such a large-scale environment. Hadoop runtime consists of a single master process (the JobTracker) and a large number of slave processes (TaskTrackers). HDFS uses a master process to manage data distribution (NameNode) and slave processes to host data locally in each node (DataNodes). When a MapReduce application (or 'job') is submitted to the runtime, it is split into a large number of Map and Reduce tasks, which are executed by the slave nodes. The runtime is responsible for dispatching tasks to slave nodes and ensuring their completion and, thus, is responsible for selecting tasks for execution across multiple jobs. Task selection and slave node assignment govern a job's opportunity to make progress, and thus influences job performance.

3. PERFORMANCE MANAGEMENT

The Adaptive Scheduler. relies on estimates of individual job completion times given a particular resource allocation, and uses these estimates so as to maximize each job's chances of meeting its performance goal. The main objective of the task scheduling mechanism is to enable a MapReduce runtime to dynamically allocate resources in a cluster of machines based on the observed progress rate achieved by the jobs, and the completion time goal associated with each job. The Adaptive Scheduler is described in detail in [9], and downloadable from [1].

3.1 Designing the Adaptive Scheduler

The Adaptive Scheduler dynamically estimates the expected completion time for each job during its execution. It does not rely on information from previous executions nor user estimates, but takes advantage of the fact that MapReduce jobs are a collection of a large number of smaller tasks. Specifically, we hypothesize that from a subset of tasks that have completed thus far, we can predict the properties of remaining tasks. The task scheduler consists of two components: a scheduling policy that assigns a priority to each job, and an allocation algorithm that assigns free slots to jobs based on their priority. The priority of each job is calculated based on the number of slots to be allocated concurrently to each job over time so that it can make its completion time goal. The scheduler assigns available task slots to jobs according to their priority.

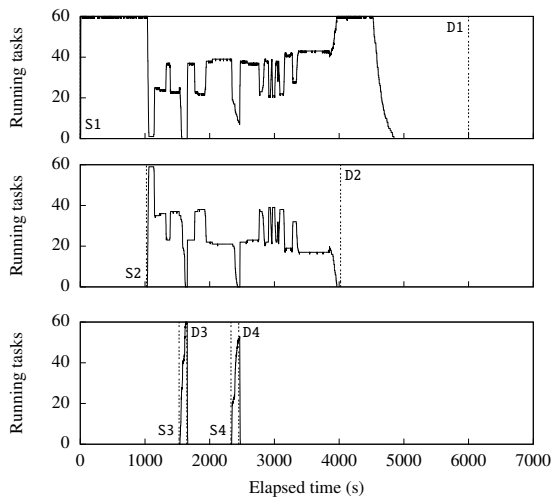


Figure 1: Allocated Resources (slots) using performance-driven co-scheduling

3.2 Results

We use three MapReduce applications for the evaluation of the core functionalities of the Adaptive Scheduler. One of the applications, Join, shows regular distributions of task lengths (each task takes approximately the same amount of time to execute), and the other two, WordCount and Simulator, present an irregular distribution. For this experiment, we use a homogeneous Hadoop cluster consisting of 60 IBM JS21 blades, each equipped with 2-way 2.1Ghz PPC970FX processor and 4GB of RAM. JobTracker and NameNode run in a separate node.

The workload (see [9] for more details) selected is composed of a big Simulator job (J1), which is configured to have a relaxed completion time goal of 6,000s; a sample WordCount job (J2) configured with a completion time goal of 3,000s; and two identical runs of the Join application (J3 and J4), each with a completion time goal of 150s. Completion time for each job is derived from the completion time goal that each application exhibits when is run in isolation. Figure 1 shows the number of slots allocated to each application over time. Jobs J1, J2, J3 and J4 are submitted at times S1, S2, S3 and S4 respectively, and the completion time goals are D1, D2, D3 and D4 respectively.

At the beginning of the test every slot is allocated to J1, as there is nothing else in the system. When J2 is submitted (at S2), it shares resources with J1, and together they use every slot in the system. When the two Join instances are submitted they get most of the resources until they complete. When J2 finishes, close to its goal again, J1 is once again allocated the entire system, and so it meets its goal.

4. ENERGY MANAGEMENT PLANS

Next generation data centers will be composed of hybrid hardware, in terms of heterogeneous nodes [7] as well as accelerator-enabled servers [5]. Hardware heterogeneity can be exploited with performance goals, but also with energy efficiency goals.

Our future research plans include the extension of the Adap-

tive Scheduler following four different lines towards the introduction of energy-saving goals into its workload management engine. The four aspects we plan to address are: 1) dynamically adapt the performance of applications based on their completion time goals; 2) based on the resource requirements of the workload, consolidate applications on nodes with different performance/power consumption ratios; 3) consolidate data and workloads to release nodes [8] that can be removed from MapReduce clusters and switched off; and 4) offload main processors through the use of next generation hardware accelerators when possible, with the goal to use specific architectures that are able to deliver higher performance with low power consumption.

As the Adaptive Scheduler for Hadoop is already able to work in collaboration with the Emotive Cloud [4] management middleware, our future plans include the complete integration of energy-efficiency goals into this platform including, but not limited to, a heterogeneous [6] mix of MapReduce, transactional and long running workloads.

5. REFERENCES

- [1] *Adaptive Scheduler. Apache issue MAPREDUCE-1380.* <https://issues.apache.org/jira/browse/MAPREDUCE-1380>.
- [2] Apache Software Foundation. Hadoop Distributed File System (HDFS) Architecture. http://hadoop.apache.org/core/docs/current/hdfs_design.html.
- [3] Apache Software Foundation. Hadoop MapReduce Tutorial. <http://hadoop.apache.org/mapreduce>.
- [4] Autonomic Systems and eBusiness Platforms research line. Barcelona Supercomputing Center (BSC). Emotive cloud. <http://www.emotivecloud.net>.
- [5] Y. Becerra, V. Beltran, D. Carrera, M. Gonzalez, J. Torres, and E. Ayguadé. Speeding up distributed mapreduce applications using hardware accelerators. In *ICPP '09: Proceedings of the 2009 International Conference on Parallel Processing*, pages 42–49, Washington, DC, USA, 2009. IEEE Computer Society.
- [6] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguadé. Enabling resource sharing between transactional and batch workloads using dynamic application placement. In *Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 203–222, 2008.
- [7] B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini. An energy case for hybrid datacenters. In *Workshop on Power Aware Computing and Systems (HotPower'09)*, Big Sky, MT, USA, 2009, 10/2009 2009.
- [8] J. Leverich and C. Kozyrakis. On the energy (in)efficiency of hadoop clusters. In *Workshop on Power Aware Computing and Systems (HotPower'09)*, Big Sky, MT, USA, 2009, 10/2009 2009.
- [9] J. Polo, D. Carrera, Y. Becerra, J. Torres, E. Ayguadé, M. Steinder, and I. Whaley. Performance-driven task co-scheduling for mapreduce environments. In *NOMS '10: Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium*, Osaka, Japan, 2010. IEEE.