A survey on DyScale: Hadoop Job Scheduler for Different Multicore Processors

Supriya.R & Mr. Kantharaju.H.C
Vemana Institute of Technology, Bengaluru, Karnataka 560034

Abstract: Multi-core processing is a growing industry trend as single-core processors rapidly reach the physical limits of possible complexity and speed. The functionality of modern processors are often driven by a given power budget that forces designers to estimate different trade-offs, e.g. to select between power efficient cores or many slow or fewer faster, power hungry cores, or a combination of them. DyScale is a new scheduling framework that exploits new opportunities and performance benefits of using servers with heterogeneous multi-core processor for MapReduce processing. These heterogeneous cores are used for forming dissimilar virtual resource pools; each resource pool is based on a unique core type. These virtual pools consist of resources of distinct virtual Hadoop clusters that operate over the same datasets and that can share their resources if needed. Resource pools can be exploited for multiclass job scheduling. Since the same data can be accessed with the slots either “slow” or “fast”, spare resources slot can be shared between different resource pools. It evaluates Performance benefits of DyScale versus First in First out (FIFO) and capacity job schedulers that are broadly used in the Hadoop community.

Keywords— MapReduce; Hadoop; heterogeneous systems; scheduling; performance

1. Introduction

The emergent modern system on a chip may include heterogeneous cores that execute the same instruction set while exhibiting different power and performance characteristics. The current SoC design is exploit a variety of choices within the same power envelope and to analyze decision trade-offs, e.g., to choose between either many slow, low-power cores, which consume much more power per core, or to select a combination of them. A typical MapReduce workload contains tasks with different performance goals: large, batch jobs that are throughput oriented, and smaller interactive jobs that are response time sensitive. Therefore, it is unclear whether under normal circumstances a typical MapReduce application may benefit from processors with faster cores. Therefore, the heterogeneous multi-core processors that have both fast and slow cores become an interesting design point for supporting different performance objectives of MapReduce jobs. DyScale that exploits capabilities offered by heterogeneous cores within a single multi-core processor for achieving a variety of performance objectives.

MapReduce and its open source implementation Hadoop offer a scalable and fault-tolerant framework for processing large data sets. MapReduce jobs are automatically parallelized, distributed, and executed on a large cluster of commodity machines. Originally, Hadoop was designed for batch-oriented processing of large production jobs. To improve the execution time of small MapReduce jobs, one cannot use the “scale-out” approach, but could benefit using a “scale-up” approach. DyScale scheduler operates potential benefits of heterogeneous multi-core processors for “faster” processing of the small, interactive MapReduce jobs, while at the same time offering an improved throughput and performance for large, batch job processing.

2. Background: MapReduce

MapReduce [1] jobs are distributed and executed across multiple systems. In the MapReduce model, computation is expressed as two main stages: map and reduce. The map stage is partitioned into map tasks and the reduce stage is partitioned into reduce tasks. The map and reduce tasks are executed by map slots and reduce slots.

In MapReduce, the programmer specifies a Map function that processes input data to generate intermediate data in the form of tuples and a Reduce function that merges the values associated with a key. In the map stage, each map task reads a split of the input data, applies the user-defined map function, and generates the intermediate set of key/value pairs. The map task then sorts and partitions these data for different reduce tasks according to a partition function.

In the reduce stage, each reduce task fetches its partition of intermediate key/value pairs from all the map tasks and merges the data with the same key.
This process is called as shuffle/sort phase. After that, it applies the user-defined reduce function to the merged value list to produce the aggregate results it is called the reduce phase. Then, the reduce outputs are written back to a distributed file system.

Job scheduling in Hadoop is shown in Figure 1. Here scheduling is performed by a master node called Job Tracker. Job Tracker is responsible for taking in requests from a client and assigning Tasktracker, with tasks to be performed. The worker Tasktracker periodically connects to the master JobTracker to report current status and the available slots. The JobTracker decides the next job to execute based on the reported information and according to a scheduling policy. Tasktracker is a daemon that accepts tasks (Map, Reduce and Shuffle) from the JobTracker. The Tasktracker keeps sending a heartbeat message to the JobTracker to notify that it is alive. Along with the heartbeat it also sends the free slots available within it to process tasks. Tasktracker starts and monitors the Map & Reduce Tasks and sends progress/status information back to the JobTracker.

![Figure 1. Job scheduling in Hadoop](image)

**Issues with heterogeneity**

At first glance, it might appear that the dynamic load-balancing [2] approach makes MapReduce frameworks inherently well-suited to heterogeneous clusters: slower (low-power) nodes would be assigned fewer tasks, while faster (high-performance) nodes would be assigned more tasks, leading to good load balance subject to the limits imposed by task granularity. Adding heterogeneous hardware resources actually degrades performance. This poor performance underscores the fact that MapReduce frameworks designed and optimized for homogeneous clusters do not directly scale to heterogeneous clusters.

3. Related Work

M. Zaharia has conducted experiment by for improving MapReduce in heterogeneous environment [3]. Today’s most popular computer applications are Internet services with millions of users. The sheer volume of data that these services work with has led to interest in parallel processing on commodity clusters. This system addresses the problem of how to robustly perform speculative execution to maximize performance. Hadoop’s scheduler starts speculative tasks based on a simple heuristic comparing each task’s progress to the average progress. Although this heuristic works well in homogeneous environments where stragglers are obvious, it shows that it can lead to severe performance degradation when its underlying assumptions are broken. It focuses on eliminating the negative effect of stragglers on job completion time by improving the scheduling strategy with speculative tasks. This technique is applicable to our case as well, especially for operating with shared spare resources that are formed by different types of slots. In future work, this system plan to evaluate more sophisticated methods of estimating finish times.

Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, conducted an experiment Improving MapReduce performance through data placement in heterogeneous Hadoop clusters [4]. The MapReduce framework can simplify the complexity of running distributed data processing functions across multiple nodes in a cluster, because MapReduce allows a programmer with no specific knowledge of distributed programming to create his/her MapReduce functions running in parallel across multiple nodes in the cluster. MapReduce automatically handles the gathering of results across the multiple nodes and return a single result or set. More importantly, the MapReduce platform can offer fault tolerance that is entirely transparent to programmers. This paper focus on improve the MapReduce performance through a heterogeneity-aware data placement strategy: faster nodes store larger amount of input data. In this way, more tasks can be executed by faster nodes without a data transfer for the map execution. It address e addresses the problem of how to place data across nodes in a way that each node has a balanced data processing load. Given a data intensive application running on a Hadoop MapReduce cluster, our data placement scheme adaptively balances the amount of data stored in each node to achieve improved data-processing performance.

G. Lee, G. Chun, and R. H. Katz, conducted an experiment on “Heterogeneity-aware resource allocation and scheduling in the cloud [5].” Data analytics are key applications running in the cloud computing environment. To improve performance and cost-effectiveness of a data analytics cluster in the cloud, the data analytics system should account for heterogeneity of the environment and workloads. In addition, it also needs to provide fairness among jobs when multiple jobs share the cluster. In this work it mainly focus on resource allocation and job
scheduling on a data analytics system in the cloud to embrace the heterogeneity of the underlying platforms and workloads. It propose to divide the resources into two dynamically adjustable pools and use the new metric “progress share” to define the share of a job in a heterogeneous environment so that better performance and fairness can be achieved. This approach only allocates resources based on the job storage requirement. Polo et al. [6] modify the MapReduce scheduler to enable it to use special hardware like GPUs to accelerate the MapReduce jobs in the heterogeneous MapReduce cluster. Jiang et al. [7] developed a MapReduce-like system in heterogeneous CPU and GPU clusters.

Q. Chen, D. Zhang, M. Guo has conducted experiment call SAMR [8], a self-adaptive MapReduce scheduling algorithm which splits the job into lots of fine-grained map and reduce tasks, then assigns them to a series of nodes. Meanwhile, it reads historical information which stored on every node and updated after every execution. Then, SAMR adjusts time weight of each stage of map and reduce tasks according to the historical information respectively. Thus, it gets the progress of each task accurately and finds which tasks need backup tasks. What’s more, it identifies slow nodes and classifies them into the sets of slow nodes dynamically. According to the information of these slow nodes, SAMR will not launch backup tasks on them, ensuring the backup tasks will not be slow tasks any more.

4. Conclusion

The conclusion of this paper is as follows. DyScale is a new scheduling framework can be implemented on top of Hadoop. DyScale creates different virtual pools based on the core-types for multi-class job scheduling. The main aim of this framework is taking advantage of capabilities of heterogeneous cores for achieving a variety of performance objectives. It creates virtual clusters, have access to the same data stored in the underlying distributed file system, and therefore, any job and any dataset can be processed by either fast or slow virtual resource pools, or their combination.

5. References


