Dynamic Resource Provisioning for Concurrent MapReduce Frameworks

Bogdan Ghiţ
Delft University of Technology
the Netherlands
b.i.ghit@tudelft.nl

Dick Epema
Delft University of Technology
the Netherlands
d.h.j.epema@tudelft.nl

ABSTRACT

Running multiple instantiations of the MapReduce framework (MR-clusters) concurrently in a multicluster system or datacenter enables workload and data isolation, which is attractive for many organizations. We provision MR-clusters such that they receive equal levels of service by assigning each such cluster a dynamically changing weight that indicates its fair share of the resources.

1. INTRODUCTION

Despite the high scalability of the MapReduce framework in large infrastructures such as multicluster systems and datacenters, *isolating* MapReduce workloads and their data is very attractive for many organizations. In this paper, we propose a *dynamic* resource management approach for provisioning multiple MR-clusters in a single multicluster or datacenter, such that distinct MR-clusters receive equal levels of service. Towards this end, we differentiate each MR-cluster by assigning a dynamically changing weight to it that indicates its fair share relative to all active MR-clusters.

Running multiple MR-clusters concurrently within the same physical infrastructure enables *four types of isolation*, viz. with respect to performance, to data management, to fault tolerance, and to versioning [1]. Deploying each MR-cluster on a static partition of the system may lead to poor resource utilization, as some MR-clusters may be accessed more frequently than others, creating an imbalance between the levels of service they receive. To dynamically change the resource allocations at runtime, we need to understand which factors can be used to differentiate the MR-clusters. Assuming no apriori knowledge about the workloads, we propose three factors that can be used to establish the fair shares of the MR-clusters: the size of their workloads, the utilization of their allocated resources, and their service levels.

As MapReduce is usually employed for data-intemsive applications, one of the main issues is to limit the overhead of reorganizing the data when *growing* or *shrinking* an MR-cluster when its share changes.

2. BACKGROUND

In this paper we take an experimental approach to resource provisioning MR-clusters. The testbed for our experiments is the multicluster system DAS-4 [1], which is a wide-area computer system dedicated to research in parallel and distributed computing that is currently in its fourth generation and that consists of six clusters distributed in institutes and organizations across the Netherlands.

KOALA [3] is a grid resource manager developed for multicluster systems such as the DAS-4 with the goal of designing, implementing, and analyzing scheduling strategies for various application types (e.g., map-reduce, cycle scavenging jobs, workflows). We have recently extended KOALA with a MapReduce runner [1], which is a specialized module for scheduling and deploying MR-clusters on demand.

3. FAIR RESOURCE ALLOCATIONS

Our basic goal when provisioning resources to multiple MR-clusters in a single multicluster or datacenter is to give them equal levels of service (e.g., throughput, response time). In order to achieve this, we want to assign each MR-cluster a dynamically changing weight that indicates the share of the resources it is entitled to.

3.1 System Model

When growing or shrinking an MR-cluster, both the storage and compute layers need to be adjusted. Consequently, we distinguish three types of nodes in the system. The *core nodes* are fully functional nodes allocated for the MR-cluster deployment, with components in both layers of the MR-cluster. The *transient nodes* are temporary compute nodes used to grow MR-clusters after their initial deployment. Removing transient nodes does not change the distribution of the data. The *transient-core nodes* are temporary fully functional nodes, also used to grow active MR-clusters. Their removal requires replicating the data they locally store.

3.2 Notion of Fairness

The MR-clusters are entitled to shares of the total datacenter capacity proportional to their given weights. For an MR-cluster i, the difference between the fraction $r_i(t)$ of resources it currently has and the share of resources it should have based on its weight $w_i(t)$ at time t is defined as its *temporal discrimination* [2]: $D_i(t) = r_i(t) - w_i(t)$.

We define the discrimination of MR-cluster i during a time

[1] http://cs.vu.nl/das4

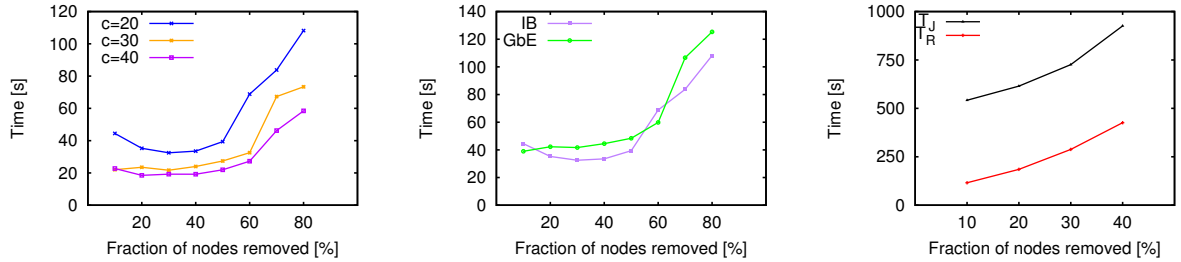

Figure 1: The average per-node reconfiguration time depending on the fraction of removed nodes for (a) MR-clusters with c core nodes and for (b) two networks on MR-clusters with 20 core nodes. The job execution time T_J and the reconfiguration time T_R when removing different fractions of nodes from a 20-node MR-cluster running a WordCount job, after 25% of the map tasks have completed (c).

interval $[t_1, t_2]$ by

$$D_i(t_1, t_2) = \int_{t_1}^{t_2} (r_i(t) - w_i(t))dt. \quad (1)$$

Setting $t_1 = a_i$ and $t_2 = d_i$ with a_i and d_i the moments of the request for the deployment and the shutdown of the MR-cluster, respectively, we obtain the *overall discrimination* of the MR-cluster. The fairness of the system over a certain interval is indicated by the *global discrimination*, which is defined as the variance $Var(D)$ across all active MR-clusters during that interval.

3.3 Policies for Setting Weights

We define three sets of policies for setting the weights of active MR-clusters, which are based on: the demands of the workloads submitted to them (*demand-driven*, e.g., in terms of the numbers of jobs or tasks in queue), the usage of the resources they are currently allocated (*usage-driven*), and the service they obtain in terms of one of our metrics (*service-driven*). In all of these policies, the weights of the MR-clusters and the global discrimination are recomputed for every interval of length T. Only when the global discrimination exceeds a threshold τ are the allocations of the MR-clusters actually changed according to the new weights.

3.4 Resizing Active MR-clusters

MR-clusters with positive discrimination have to shrink their resources. There are two options for doing so: non-preemptive, with the data replication running in parallel with the workload execution, or preemptive, with the shrinking phase starting after the running tasks are killed. MR-clusters with negative discrimination have to grow their resources. Based on the type of additional nodes used, there are two options: growing with transient or with transient-core nodes.

4. EXPERIMENTS

In our experiments, we assess the performance of our three sets of policies for different values of T and τ with (non-)preemptive shrinking and core/transient growing. As a basis, we perform a set of *micro-experiments* in which we assess for instance the reconfiguration overhead when shrinking an active MR-cluster, and the performance of running single MapReduce applications based on the number of core and transient nodes allocated to it. For the former micro-experiment, we set up MR-clusters of different sizes with 10 GB per node replicated 3 times. We find that the cluster

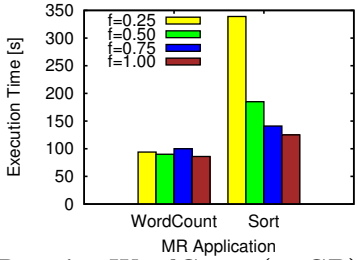

Figure 2: Running WordCount (40 GB) and Sort (50 GB) on 40-node MR-clusters with different fractions f of core nodes.

size and the fraction of nodes removed have a significant impact on the reconfiguration time (Figure 1a). Furthermore, increasing the network bandwidth to 20 Gb/s improves the reconfiguration time with only less than 20% (Figure 1b). Shrinking an active MR-cluster while running a job increases the reconfiguration time (Figure 1c). For the latter micro-experiment, we set up MR-clusters of different fractions of core nodes and we run two common MapReduce applications. We find that WordCount scales better on transient nodes than Sort (Figure 2).

5. CONCLUSION

Dynamic resource provisioning to multiple instantiations of the MapReduce framework deployed in single multiclusters or datacenters is of both practical and theoretical interest. In this paper we propose a form of such provisioning that targets equal levels of service for the active MR-clusters.

6. ACKNOWLEDGMENT

This work was partially funded by the Dutch national research program COMMIT.

7. REFERENCES

[1] B. Ghit, N. Yigitbasi, and D. Epema. Resource Management for Dynamic MapReduce Clusters in Multicluster Systems. In *IEEE High Performance Computing, Networking, Storage and Analysis (SCC), SC Companion, 2012.*

[2] D. Raz, H. Levy, and B. Avi-Itzhak. A Resource-Allocation Queueing Fairness Measure. In *SIGMETRICS 2004.*

[3] O. Sonmez, H. Mohamed, and D. Epema. On the Benefit of Processor Co-Allocation in Multicluster Grid Systems. *IEEE Trans. on Parallel and Distributed Systems*, 21:778–789, 2010.