Online Management of Jobs in Clusters using Virtual Machines

Fabien Hermenier

Cluster computing is an attractive solution to meet the growing computational requirements of scientific applications. In this setting, a user organizes a job as a collection of tasks that each requires a finite amount of resources for a bounded amount of time. A scheduling algorithm is responsible of selecting the jobs to execute on the cluster by finding, for each task composing the jobs, a node with a sufficient amount of free resources. Static scheduling algorithms allocate each task to a node and a constant amount of resources for all the duration of the job. Such strategies lead to a waste of resources and a non-optimal schedule of the jobs as each task does not use all of the allocated resources at all times. On the opposite, dynamic scheduling algorithms manipulate in live the state of the jobs using preemption and the location of their tasks using process migration. These algorithms provide a finer use of resources by allocating them according to the current demand instead of the users estimate. In practice, dynamic scheduling strategies are hard to deploy on clusters. First, the actions that manipulate the tasks in live are tedious to implement in a non-intrusive way. Second, their executions are time and resource consuming and misusing them lead to a high computational overhead. Finally, each cluster has its own architecture and some specific objectives or constraints that may not fit with the available scheduling strategies.

In this thesis, we investigate to ease the development and the use of dynamic scheduling strategies. First we propose to use virtual machines (VMs) to execute the jobs in their legacy environment. In addition, virtualization brings the necessary actions to manipulate VMs as dynamic scheduling strategies manipulate tasks: live migration allows to relocate a VM that embeds a task with a negligible downtime, while the suspend-to-disk and the resume actions provide jobs preemption. To ease the development of specific dynamic scheduling strategies, we propose to use constraint programming (CP), a flexible approach to model and solve combinatorial problems. With this approach, the developer only focuses on modelling its problem by stating the constraints (logical relations) that must be satisfied by the solutions. Hence, we have developed a constraint programming model for a core dynamic scheduling problem as well as additional composable constraints to specify the strategy by restricting the state or the location of VMs. Finally, we introduce the concept of dynamic reconfiguration, a generic method to perform the transition between the current schedule of the VMs and the new computed one. Relying on a model that estimates the duration and the impact on performance of each action, our module, also based on CP plans the actions to ensure their feasibility and to minimize the total duration of their execution.

We have validated our approach with the implementation of a prototype named Entropy[1] which was used to implement two use cases. The first provides dynamic consolidation by packing all the running VMs on the minimum number of nodes while satisfying their CPU and memory resources needs. This strategy reduces the energy consumption of the clusters when unused nodes are turned off. Our second use case reduces the complexity of developing dynamic scheduling strategy. Using the dynamic reconfiguration, the developer only provides an algorithm to select the jobs to run while the reconfiguration module finds a host for each selected VM and provides an efficient transition to reach the new schedule.

We have evaluated Entropy through several experiments using both simulations with up to 200 nodes and 400 VMs and clusters with up to 35 nodes and 70 VMs. We have observed that our approach for dynamic reconfiguration reduces by up to 70% the duration of a reconfiguration as compared to the heuristic *First Fit Decrease* (FFD). This has led to a reactive system that quickly fixes non-optimal schedules and provides an efficient use of resources for each strategy we have developed. For dynamic consolidation, we have compared our implementation with the common heuristic FFD. Our solution performs twice as much reconfigurations and uses half as much resources to execute the same workload. For our second strategy, we have developed a sample scheduling algorithm, similar to *First Come, First Serve* but which executes a workload of 11 jobs 40% faster.

[1] `http://entropy.gforge.inria.fr`