Universiteit Gent
Faculteit Ingenieurswetenschappen
Vakgroep Informatietechnologie

Dynamische planning van rekenintensieve toepassingen in gedistribueerde computationele omgevingen

Dynamic Scheduling of Computationally Intensive Applications in Distributed Computing Environments

Maria Chtepen

Proefschrift tot het bekomen van de graad van
Doctor in de Ingenieurswetenschappen:
Computerwetenschappen
Academiejaar 2010-2011

Summary

In this dissertation we address the issue of efficient execution of computationally intensive applications within distributed environments, such as clusters, peer-to-peer networks, grids. More specifically, we investigate how to incorporate runtime information in the job scheduling process. Typical for large distributed systems is that their computational capacity and availability can strongly vary over time. Furthermore, their resources are often shared among different user groups, resulting in high diversity and varying complexity of application (jobs) runs.

There are different challenges related to the process of distribution execution, which are situated at the organizational as well as at software and hardware levels. In this context, we focus on two important problems, which have significant impact on the performance and flexibility of distributed environments: fault-tolerance and scheduling of applications with dependencies (workflows). Several dynamic checkpointing, replication and scheduling solutions are proposed that take into account the dynamic nature of distributed resources and applications, by reconsidering previously taken decisions at run-time.

To study the behaviour of grid environments in dynamic scenarios, we have developed a discrete-event simulator, called DSiDE. DSiDE forms a flexible and portable framework for modeling and simulation of distributed computing environments. Thus far, it mainly contains a set of built-in grid components, which, however, can easily be extended with components for other types of distributed systems, such as P2P networks and clouds. The main advantages of DSiDE, compared to other existing general and grid-specific simulation frameworks, are its extensibility, genericity, relatively short simulation times and ability to easily model dynamic system and application behaviour. Dynamic behaviour supported by DSiDE includes varying resource load and availability, varying job arrival frequency, changing application dynamics, *etc*. In general, the simulator is composed of three separate modules:

- **DGen**: grid models and dynamic behaviour modeling event distributions are provided as input into the simulator using an XML-based format. Recurrent events from the input file are translated by DGen into a set of individual events that can be loaded into the DExec module.
- **DExec**: is the kernel of the simulator, where simulations are run by processing all registered events one after the other.
- **DMExec**: allows for automatic execution of either several predefined simulation experiments or of the same experiment with different seeds for ran-

dom number generators.

The first issue addressed is fault-tolerance in grid systems. We consider unreliable grids, *i.e.* grids where resources are subject to failure and restart, where applications of different duration are executed. The aim of this study is to give a quantitative indication of the effect of resource failure on grid performance and to justify the use of fault-tolerant techniques in these dynamic environments. The results have shown considerable performance degradation of jobs with long execution times in unstable systems. Also, they suggest that the frequency of resource failure has a larger effect on grid performance than the time it takes a resource to restore. As a consequence of this study, several dynamic fault-tolerance algorithms were proposed. The algorithms are based on well-known job replication and checkpointing techniques. The main issue with these techniques is the large overhead introduced when the number of replicas and the checkpointing intervals are chosen inappropriately. Many existing research efforts in this area are dedicated to determining the values of both parameters analytically, based on knowledge of the application and distributed environments at hand. Unfortunately, the resulting solutions are often based on unrealistically simplified assumptions or limited to a certain type of applications. Therefore, the algorithms introduced modify the number of replicas and the checkpointing intervals dynamically, based on run-time information on system load, and the history of resource failures respectively. Simulation results have shown that adaptive job replication is the most low-cost approach in systems with low and variable load, while in heavily loaded environments, checkpointing with dynamic interval can significantly reduce run-time overhead, compared to periodic checkpointing. The advantages of both techniques can be combined in a hybrid approach that can best be utilized when system properties are not known in advance.

Another issue addressed in this research is scheduling of workflow applications with input dependencies. Input dependencies imply that a task within a job requires inputs generated by another task before it can proceed with its execution. This is a loosely-coupled type of dependencies with limited run-time communication overhead, which can gain significant performance improvement from distributed execution of parallel tasks. Therefore, we propose a dynamic algorithm for workflows that is based on the assumption that tasks within a job have varying computational complexity. The algorithm operates on applications for which execution progress can be monitored at run-time. Based on the monitored task progress and current resource capacity, tasks are (re)scheduled to resources to keep the execution times of parallel tasks with the same *dependents* in balance. The idea is that a *dependent* task cannot be executed until all its *parents* have generated the required outputs. Therefore, it is desirable to assign *parents* with low execution times to slow computational resources and keep fast resources for tasks requiring fast processing. The performance of the algorithm proposed was evaluated using a workload model derived from a real-world tool for modeling and virtual experimentation with environmental systems, called Tornado. The results suggested a makespan reduction of about 35% for the job as a whole. However, it is important to mention that the performance of the algorithm depends on the quality of task execution time predictions, since these predictions are used to balance the execution time of running tasks. Initially, execution time predictions are constructed by extrapolation from the last task progress measurements. Unfortunately, this simple method is sensitive to internal application dynamics and variations in resource load, which leads to overzealous and overhead-prone task migrations within each scheduling iteration. To reduce this overhead, a more effective prediction method is introduced. The method is based on non-linear curve-fitting of predictions (constructed by extrapolation) against a number of predefined prediction evolution models. Simulation experiments with Tornado workload suggested up to 15% performance improvement in case the curve-fitting based prediction method is applied to the dynamic algorithm proposed.