The Performance Impact
of Advance Reservation Meta-scheduling

Quinn Snell, Mark Clement, David Jackson, and Chad Gregory
Brigham Young University, Provo, Utah 84602
{snell, clement, gregory}@cs.byu.edu, jacksond@mhpcc.edu

Abstract. As supercomputing resources become more available, users will require resources managed by several local schedulers. To gain access to a collection of resources, current systems require metajobs to run during locked down periods when the resources are only available for metajob use. It is more convenient and efficient if the user is able to make a reservation at the soonest time when all resources are available. System administrators are reluctant to allow reservations external to locked down periods because of the impact reservations may have on utilization and the Quality of Service that the center is able to provide to its normal users. This research quantifies the impact of advance reservations on and outlines the algorithms that must be used to schedule metajobs. The Maui scheduler is used to examine metascheduling using trace files from existing supercomputing centers. These results indicate that advance reservations can improve the response time for metajobs, while not significantly impacting overall system performance.

1 Introduction

Recently, there have been a number of research groups focusing their efforts on utilizing the combined resources of multiple supercomputing facilities [1,2,3,4]. The motives for this avenue of research are obvious. First, and perhaps foremost, is the belief that the proper combination of resources and their aggregate processing power will yield a system that is both scalable and potentially more efficient. A system such as the one proposed in this paper would provide enhanced scheduling services for at least three kinds of jobs: those that require more resources than are available at any one site, jobs that require a combination of resources that are not available at any one site, and finally, jobs from users that desire a better overall response time than could be obtained by limiting their jobs to using the resources found at any single site.

Regardless of the job type, using the combined resources of more than one site requires cooperation that is not inherent in local resource scheduling systems. A system that coordinates and works with the local schedulers is required. Such a system is often called a metascheduler. The metascheduler makes this aggregation of resources available to what are termed throughout this paper as 'metajobs'. Each metascheduler maintains a job queue to which these metajobs are submitted by users where they are stored until they complete. A metajob can
be simply a normal batch job submitted to the meta scheduler or it may be modified to utilize special functionality that can only be found in a metascheduling environment. There are three key differences between a metajob and a standard batch job: 1) The machine or local scheduler under which the metajob will run is not known at job submit time. 2) The metajob may contain a utility function which instructs the metascheduler as to what aspects of the resources are most important to it. These aspects may include cost, machine speed, time until availability, etc. 3) The resources utilized by the metajob may span more than one machine or local scheduler.

Some current metascheduling systems work by dedicating a set of local resources to be used and scheduled by the metascheduler. With these metaschedulers, local management and administration staff determine both a maximum amount of resources allowed for metascheduled work and a set of timeframes during which these resources can be used. During each of these timeframes, the metascheduler assumes full control of all allowed resources, preventing their use by locally submitted jobs. These resources remain unavailable regardless of the metascheduled workload. Supercomputer system managers have reported average utilizations ranging between 5% and 25% on those nodes that are dedicated for metacomputing systems. Due to the fragmentation of resources, low utilizations occur even when there is a backlog of local jobs. A metascheduling system based on this model clearly wastes valuable resources, significantly lowering the overall system utilization and increasing the average job turnaround time for local jobs.

This dedicated resource metascheduler schedules resources as if it were a local scheduler, according to its own private set of policies and priorities. It does not leverage the knowledge or capabilities of the local scheduler. The local scheduler has, in reality, completely relinquished control of the resources that have been dedicated to the metascheduler. This brings up more issues. First, lower utilization can be expected since the local compute resources are now fragmented. Additionally, the resource fragments are exclusive of each other meaning that those resources dedicated for metascheduled jobs cannot be used by locally submitted jobs even if they are idle; the same holds true for metascheduled jobs and resources dedicated for local workload. Also, under existing ‘dedicated resource’ metaschedulers, each resource fragment is scheduled according to an independent, private set of policies. There is no cooperation between the local schedulers and metaschedulers, and there is no knowledge of each others policies, priorities, or workload. In consequence of these conditions, there is no opportunity for cross-fragment scheduling optimizations such as backfill or intelligent node allocation.

Ursala, a metascheduler developed at Brigham Young University and used for the research described in this paper, cooperates with the local schedulers and introduces metascheduled jobs into the locally-produced workload. A system capable of doing this can achieve better overall utilization of resources because full information and control are maintained at the local scheduler level. Local scheduling optimizations such as backfill can then occur. Instead of ded-