Improving Speedup and Response Times by Replicating Parallel Programs on a SNOW*

Gaurav D. Ghare and Scott T. Leutenegger

Department of Computer Science
University of Denver
Denver, CO 80208-0189
{gghare,leut}@cs.du.edu

Abstract. Idle computation cycles of a shared network of workstations are increasingly being used to run batch parallel programs. For one common paradigm, the batch program task running on an idle workstation is preempted when the owner reclaims the workstation. This owner interference has a considerable impact on the execution time of a batch program, especially in the case of large parallel programs. Replication of batch program tasks has been used to reduce the impact of owner interference. We show analytically that replication can significantly improve parallel program speedup. Perhaps surprisingly, replication can also improve efficiency for certain workloads. We present analysis to quantify the amount of speedup and efficiency improvement. Furthermore, we provide analysis to help determine whether extra available workstations should be used for increasing job parallelism or for task replication.

1 Introduction

Networks of workstations (NOWs) have been used to run parallel programs for some time [1,2,3,4,5,6,7,8,9,10]. The NOW may be dedicated as in the case of Beowulf [11] or shared as in the case of Condor [12,13]. When the NOW is shared with workstation owner processes, it is referred to as a shared network of workstations (SNOW). The speedup of parallel programs running on a SNOW can be greatly affected by workstation owner interference. In this paper we consider a SNOW where parallel jobs are run in an opportunistic fashion as in Condor. In such an environment, workstation owner processes have preemptive priority over batched parallel programs. Workstation reclamations by owner processes stop execution of the batch job task and hence may significantly impact the parallel job response time.

One approach to prevent a workstation reclamation from impacting parallel job performance is to replicate some or all of the parallel tasks [5,6]. When replication is used, parallel job performance is not affected by reclamations as long as, at least one replica of each task completes without interference.

In [5] the author compares competition protocols and migration protocols for sequential and distributed programs on variable-speed processors. A SNOW is

* This work was supported by the NSF grant ACI-9733658.

D. Feitelson, L. Rudolph, and U. Schwiegelshohn (Eds.): JSSPP 2004, LNCS 3277, pp. 264–287, 2005.

treated as a collection of variable-speed processors where background programs have lower priority than foreground owner programs. Competition protocols use replication to reduce the impact of owner interference. Simulation results of the performance of competition for distributed programs are presented. However there is no mathematical analysis of the performance of competition protocols for distributed programs. Competition policy issues such as, how to allocate workstations for replicating tasks of a program, are not studied.

The authors in [6] demonstrate that owner interference considerably impacts the performance of a parallel program running in a SNOW environment. Futhermore, they demonstrate that using task replication can significantly improve job response time. The study only considers the loss of up to one workstation in a batch of tasks. Tasks are assumed to be of different sizes allowing for a mix of short and long service demand tasks. Performance of replication for programs with tight coscheduling requirements is not studied. They consider one no-replication and two replication strategies to alleviate the negative impact of owner interference: no-replication (NR) adds the extra workstations to the general pool, single replication (SR) replicates the largest task in a batch using one extra workstation and uses other extra workstations in the general pool, and complete replication (CR) replicates the K largest tasks using K extra workstations. They show with experimental workloads that CR performs better than SR. Our work differs in that we provide analysis and proofs instead of simulation, we consider synchronized workloads, we study various tradeoffs in how best to allocate K extra workstations, and we consider the tradeoff between using extra processors for parallelism versus replication.

Rosenberg [10] develops a model for devising a schedule that maximizes the amount of work accomplished from an embarassingly parallel workload. Owner interference is considered as an adversial process between the owner and the user running background programs. The instantaneous probability of workstation reclamation is assumed to be known. In [2] the authors consider sharing a bag of identically complex tasks in a heterogeneous network of workstations (HNOW). The problem considered is accomplishing as much work as possible on the HNOW, during a prespecified fixed period of time. Neither of [2,10] consider the effect of replication for reducing the impact of owner interference on parallel program performance.

In this paper we show that replication can result in significant speedup improvements and we analytically quantify parallel task replication benefits for two workload models: tightly-coupled barrier synchronized and loosely-coupled barrier synchronized. We allow for multiple workstation reclamations during the execution of a batch of tasks. We assume knowledge of the probability that a workstation may be reclaimed before a task completes, but do not assume knowledge of the instantaneous probability of workstation reclamation as in [2,10]. We analytically study how to distribute extra workstations not only among tasks of a program but also between two programs.

In a dedicated parallel processing machine, increasing a parallel program's workstation allocation beyond the program's maximum parallelism will reduce