Runtime Support for Replicated Parallel Simulators of an ATM Network on Workstation Clusters

Kam Hong Shum[1] and Shuo-Yen Robert Li[2]

[1] Computer Laboratory, University of Cambridge, Cambridge CB2 3QG, UK
[2] Dept. of Information Engineering, Chinese University of Hong Kong, Hong Kong

Abstract. An effective approach of speeding up the simulation of an ATM network on workstation clusters is presented. In this approach, multiple simulation runs are performed by replicated parallel simulators (RPSs) concurrently. Since the execution platform of the simulation is in a shared-network environment, the RPSs must compete with other applications for resources. The RPSs support adaptive execution by reconfiguring the grain-size of their logical processes dynamically. In addition, scheduling policies are proposed to facilitate efficient allocation of workstations to the RPSs. Experiments are conducted to evaluate the performance of three proposed scheduling policies in the scenarios of homogeneous and heterogeneous workstation clusters.

1 Replicated Parallel Simulators

Many discrete event simulations are regarded as computationally intensive. To reduce the turn-around time of simulation, parallelism is introduced to perform simulation operations in multi-processor or multi-computer machines. In a parallel simulation, a model is decomposed into logical processes and then the logical processes execute in distributed processors. Another approach of parallelism applied to simulation is to run multiple serial simulation programs on multiple processors in parallel and average the results at the end of the runs. This approach is referred to as replicated serial simulation (RSS) [3]. The major advantage of this approach is providing a simple implementation to reduce the overall turnaround time of multiple simulation runs. However, the RSS approach may not be adequate if the time and computational complexity of the simulation model is too demanding to be executed serially.

With the advent of recent technology, workstations have become a powerful and yet inexpensive computational resource. In this paper, a parallelism in which simulation runs are executed by replicated copies of parallel simulation on workstation clusters in parallel is proposed. This approach aims at combining the benefits of the parallel simulation approach and the RSS approach, which are reduction in the turnaround time of each simulation run and the overall turnaround time of all the runs. Although the proposed approach can be applied to different simulation applications, the RPSs described in this paper are designed for the purpose of evaluating the performance of the traffic flow control and call set-up algorithms for an ATM network proposed in [2].

2 The Runtime System and Scheduling Policies

An RPS adapts to the dynamics of resource availability such as changing workloads on individual workstations by reconfiguring the agglomeration of tasks, called *grains*, at runtime to improve workload distribution. An RPS can reconfigure its grains in two different ways [4]: first, the grains and their neighboring grains race one another until all their tasks are accomplished. If the speed of processing the tasks between the grains are different, the workloads of the slow grains will be shared with the faster grains through task relocation. Second, the tasks can be grouped into predefined partitioning levels and an RPS can switch from one partitioning level to another partitioning level. Although these partitioning levels restrict the number of possible task groupings, they reduce the time in searching for suitable grain-to-workstation mappings at runtime.

In addition, scheduling is required to allocate workstations fairly and efficiently to parallel applications in a shared-network environment. A system called *Comedians* (Competitive Environment for Distributed and Adaptive Applications) [5] is developed to tackle the problem of workstation allocation and, at the same time, to maximize the speedup of individual parallel applications. In this paper, the Comedians system is used to support the scheduling and adaptive execution of RPSs. The Comedians system coordinates the execution of parallel applications on workstation clusters. In the system, workstations are partitioned into clusters according to their processing speed. An application can run on more than one workstation clusters, but one of the clusters is specified as local cluster and the others as remote clusters.

Like the RSS, the *first N replications initiated* (FNI) scheduling method [3] is applied by the RPSs to obtain statistically accurate simulation results. In this scheduling policy, N results of simulation runs are recorded from the first N replications initiated. The value of N is determined by the termination condition of simulation, for example when a desirable confidence interval of output results is obtained. Since the FNI does not specify any control over workstation allocation, an RPS will compete with all the other applications on the Comedians system based on its runtime performance. To harness the dynamic computational resource of workstation clusters, special scheduling policies are proposed to build on the top of the FNI. These policies are enforced by the Comedians system. Three scheduling policies are proposed and compared in this paper; they are FNI-SA *(Static Assignment)*, FNI-DA *(Dynamic Assignment)*, and FNI-DAC *(Dynamic Assignment with Coalition)*.

In the FNI-SA policy, an RPS can only be executed in the workstations of its local cluster, but different RPSs can be assigned to different local clusters. Whereas, the RPSs in the FNI-DA policy can be executed in the workstations of multiple clusters dynamically. The FNI-DA and the FNI-DAC policies are basically the same except that the FNI-DAC policy allows the RPSs from the same user to form coalition partners. Since RPSs from the same user have a single objective – the completion of all simulation runs, the RPSs should not compete with one another for workstations. Instead the RPSs with earlier initiated simulation run should have higher priority than the other to get resources because the

simulation terminates as soon as the first N replications finishes their runs. This speciality allows the RPSs to be executed more efficiently on workstation clusters. If a coalition is formed, other parallel applications that are running on the Comedians system are regarded as alien applications to the coalition. However, the alien applications regard the coalition as independent parallel applications.

3 Experiments and Results

The experiments are carried out on DEC3100 and DEC Alpha workstation clusters in which all the workstations are connected by Ethernet in different subnets. Unexpected interference from other users is minimized throughout the experiments so as to ensure a controlled environment. All RPSs are written in a single program multiple data (SPMD) model meaning that all workstations run the same piece of program. They can be partitioned into 1 grain (partitioning level 0), 3 grains (partitioning level 1), and 9 grains (partitioning level 2). The RPSs and the Comedians system use PVM [1] as their message passing interface.

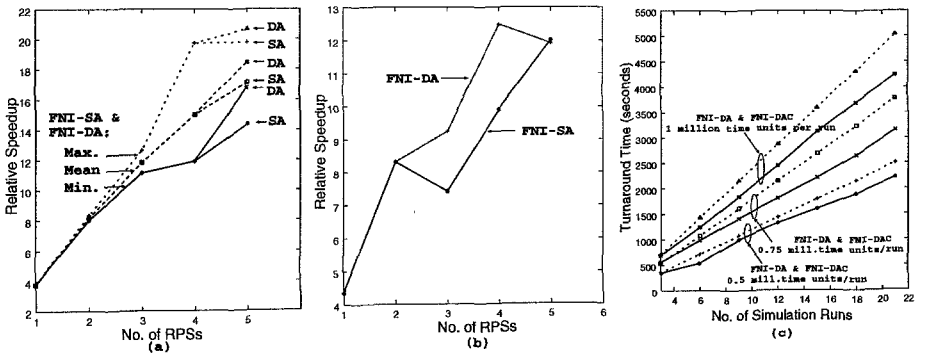

Fig. 1. Comparison of (a) FNI-SA and FNI-DA policies (Scenario One); (b) same as (a) but with an alien application; (c) FNI-DA and FNI-DAC policies (Scenario Two)

The scheduling policies are tested in two scenarios. In the first scenario, simulation runs are running on two homogeneous DEC3100 workstation clusters; each cluster consists of twenty-four workstations. Figure 1a shows the relative speedup of the FNI-SA and the FNI-DA policies when different number of RPSs are running. The figure records the maximum, the mean, and the minimum relative speedup of the RPSs for finishing ten simulation runs, each run performs half a million of simulation time units. The performance of the FNI-SA and the FNI-DA policies is similar until the number of RPS becomes five. In this case, there are not enough workstations for all the RPSs to split to the highest partitioning level. As indicated in figure 1a, the speedup is improved by applying the FNI-DA policy because the RPSs can relocate their grains to the remote cluster for execution. The discrepancy of the speedup between the maximum

and the minimum is great when four RPSs are running because two of the RPSs perform one more simulation run than the other RPSs. These two policies are also tested in the situation when an alien application is running on one of the clusters. The alien application is a block-matrix multiplication program which has the same partitioning levels as the RPSs. Figure 1b depicts the relative speedup of the simulation when different numbers of RPSs are running with the alien application. The results show that the FNI-DA policy can sustain performance gain by allowing one of the RPSs to migrate its grains to the remote cluster when there are not enough workstations available in its local cluster. When the number of RPSs is greater than or equal to five, both clusters become overcrowded.

In the second scenario, the simulation is performed on two heterogeneous workstation clusters – one cluster of thirty DEC3100 workstations and the other cluster of six DEC Alpha workstations. Since the speedup of an RPS at partitioning level one in DEC Alpha workstations is higher than the speedup at partitioning level two in DEC3100 workstations, the RPSs will compete for DEC Alpha machines even at the expense of merging to a lower partitioning level. Due to the competition of workstations in the Comedians system, it is possible that some RPSs are constantly excluded from running in DEC Alpha machines. As a result, the overall turnaround time will be restricted by the slowest RPS. Experiments are conducted to compare this special case of the FNI-DA policy with the FNI-DAC policy. Figure 1c depicts the performance of the policies when three RPSs are running in this scenario. The figure suggests that the difference in overall turnaround time between these two policies rises slowly when the number of simulation runs increases.

In summary, the experimental results demonstrate that the overall performance of the simulation can be enhanced significantly by combining the parallelism and the replication of modelling. The results also indicate that the overall performance can further be improved if (i) the RPSs can relocate their grains to remote clusters dynamically; (ii) the RPSs can form a coalition so that the priority of workstation allocation decreases with the order of initiation.

References

1. A. Giest, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM 3.0 User's Guide and Reference Manual*, Feb. 1993.
2. S.-Y.R. Li and K.H. Shum. Distributed algorithms for flow control & call set-up in a broadband packet network. In *Proc. of 12th EFOC&N*, pages 182–186, 1994.
3. Y.-B. Lin. Parallel independent replicated simulation on a network of workstations. In *Proc. of 8th Workshop on Parallel and Distributed Simulation*, pages 73–80, 1994.
4. K.H. Shum. Adaptive distributed computing through competition. In *Proc. of the International Conference on Configurable Distributed Systems*, pages 200–207, Maryland, May 1996. IEEE Computer Society.
5. K.H. Shum and K. Moody. A competitive environment for parallel applications on heterogeneous workstation clusters. In *Proc. of the Heterogeneous Computing Workshop (HCW'96), IPPS'96, presented in the joint session with the 2nd Workshop on Job Scheduling Strategies for Parallel Processing*, Hawaii, April 1996.