

Homework #2 (distributed May 5, 2004; due May 14, 2004)

This assignment covers the ideas behind resource management models and their composition into a larger distributed (or Grid) application. These topics are covered by the lectures and readings from April 21, 2004 through May 12, 2004.

The completed assignment can be emailed to Professor Chien, or paper copies dropped off outside my office.

1. Utility computing as characterized by Rolia, et. al. comes in two varieties: full server and shared utility. Their work characterized some of the potential efficiencies in such utilities based on the use of a number of data centers, and detailed load measurements taking from those servers in regular intervals.
  - a. Based on data from Figure 6a, compare the potential efficiency for set of 16-way servers to that achievable for a set of 4-way servers. Discuss each of the four scenarios for both average and peak and indicate what the relative values in the graph mean in terms of the underlying sample data.
  - b. Consider a modification of Problem #1 from Homework #1 based on the statistics shown in Figure 6a. Using the effective cost of outsourcing a server you calculated in Problem 1.d, if we use the 80 percentage targets, what cost per interval where the desired performance is not met can be tolerated?
  - c. What if we used the 50 percentage targets numbers to size capacity for the shared utility?
2. We have been studying the possibility of adaptive distributed applications which can not only select a set of resources which give them good performance (or any range of capabilities), but also choose a new set when the current resources are no longer desirable. In this question, we will explore some of the costs involved in dynamic adaptation. First, assume we can have application which executes on a set of 16 parallel resources.
  - a. In order to begin execution, the program launcher considers clusters of machines in sets of 32. Each set of machines is monitored and “forecast” for its capabilities. Suppose that the forecasts each cluster of machines (one value for each cluster) occupy a uniform distribution for predicted speed normalized from 0 to 10. How many samples should the resource selector consider before choosing a cluster if it wants to bound the potential loss in not looking at one more to under 10%? Said another way, give a decision procedure which achieves this goal with a minimum number of samples. Explain.
  - b. If the quality of the resources (the speed) drifts at a rate of one unit of normalized speed per hour. That is, assume that the selected system starts as speed =8, and at each hour changes up or down one unit of speed. What is the right decision procedure for deciding that its time to jump to another cluster randomly chosen? (assume the migration costs nothing) Explain.
  - c. If migration costs the equivalent of two units of speed for one hour, what is the right decision procedure? Explain.
  - d. Suppose that in addition to monitoring the local speed, one can also monitor the speed of other resources. Assuming that you can know the forecast speed of four other 16-node clusters and they have the same drift rate as on part b, what is the right decision procedure for when to migrate (assuming the migration costs the equivalent of two units of speed for one hour). Explain your answer.
  - e. Explain how these decision procedures vary with the cost of information collection and performance drift rate.
3. Consider the challenge of intelligently scheduling a distributed grid application using resources that provide a range of different resource management models. In this question, we will explore how these underlying resource models affect the type of performance achievable. First, consider a parallel

application which for  $N$  processors ( $65 > N > 2$ ), achieves a speedup  $= N / (0.05 * N^{1/2})$ . We won't consider cases where the application uses less than two processors, and the code doesn't scale beyond 64 nodes.

- a. To help you get started, plot the execution speed of the application as a function of  $N$ , the number of processors used.
  - b. Consider a desktop grid in which there are plentiful resources, but the resources come and go asynchronously with a mean-time to departure for each processor of one processor/(two hours) at a continuous rate. That is if we observed a single processor, the mean-time to departure would be two hours. For ten processors, 12 minutes, and so on. Each time a processor departs, we must move our computation to another processor, stopping the entire computation for 2 minutes. How fast will we make progress if we run on 64 processors? What number of processors will yield the most rapid progress? Explain.
  - c. Consider a set of grid resources governed by batch scheduling. These resources are heavily utilized and thus have a queueing time in hours equal to the # of nodes requested. Thus for a 32-node job, the average queueing time would be 32 hours. Suppose that these queueing times are normally distributed with a standard deviation of 6 hours, and are combined with limits of 128 total CPU hours per run (that is, the smaller runs are allowed to run for longer wall clock time). If there is one cluster, how fast will we make progress if we run on 64 processors? What number of processors will yield the most rapid progress? Explain.
  - d. How will our performance improve if we can choose from several clusters?
4. Now, we will continue the exploration of the performance of the application described in question #3, but consider a set of resources governed by proportional share slice-based scheduling as in Planetlab. The shares are assumed to correspond to 1Mcps each, and the nodes are 1.2Gcps total capacity of which only 1Gcps is allocated to slices.
- a. Suppose that our job can be allocated 50 shares on a number of nodes ranging from 2 to 64, and we are guaranteed that no more than 1000 shares are allocated on each node. What performance should we expect? Does this depend on the nature of the parallel application? Explain.
  - b. Suppose that our parallel job is competing with 32 other identical jobs to get a set of shares in a system with 64 total machines. Let's assume that there is symmetry in the load generation – the more aggressive our instance of the job becomes, the more aggressive the others become. In this case, for simplicity, assume that the number of shares which can be allocated for each system is unlimited, and that the weighted fair queueing scheme allocates the oversubscribed resources in proportion. On how many processors should we request the 50 shares to achieve best performance?
  - c. Under what conditions of competitive load does it make sense to request different numbers of shares and processors? That is, what is the best strategy? (for example, if all the jobs could request 500 shares would this change the situation?) Explain.
  - d. Suppose instead that we limit the number of shares on each node to 1000, and batch queue any requests that cannot be immediately satisfied, does this change the best strategy?