CSC470 Fall, 2000

Computer System Modelling And Analysis

Laboratory #2
October 3/5 & 10/12, 2000

Purpose

In this laboratory, you are asked to build and experiment with a simulation model to evaluate strategies for effectively exploiting load balancing in a distributed system.

Context

Consider a system of identical computing nodes linked together by a local area network. Each node has its own processor, its own disks, and its own set of direct users. Such a system can be underutilized if there are times when some nodes are idle, yet others have two or more jobs ready to execute. This underutilization can be counteracted through the technique of *load-sharing* -- using the network to move jobs from overloaded nodes to execute on underloaded nodes. Load-sharing is not without its costs, as it takes time to determine the loads at other nodes, and to transfer a job from one node to another over the network.

The main parameters that describe such a distributed system are shown in the table below.

parameter	meaning
N	The number of (identical) nodes in the distributed system;
A	The average interarrival time between jobs submitted at *each* node (The distribution of interarrival times will be exponential);
D	The average service time of jobs at *each* non-idle node (The distribution of service times will be hyperexponential, with a variance the is four times the mean squared);
M	The average time to move a job from one node to another (The distribution of transfer time will be uniform between 0 and twice the mean).
I	The average time to determine the current queue length at another node (The distribution of probe time will be uniform between 0 and twice the mean);
C	The total number of job response times used to estimate the mean (in all runs and all batches);
α	The error tolerance for the confidence interval.

No Knowledge Strategies

There are several important strategies for load-sharing. They vary in their sophistication and performance. Some simple ones do not require any knowledge at one node of the current load situation at other nodes:

- *No Load-Sharing*
  The basic (degenerate) case is that of no load-sharing. That is, each job is processed at the node to which it arrives.
- *Single-Bounce*
  A second strategy is called single-bounce. Under the single-bounce strategy, when a job arrives at a node, it is processed locally if the node is idle. Otherwise, the job is sent to some other node selected at random and processed there (with no possibility of being forwarded a second time). A transfer delay is incurred by each job that is forwarded.

- *Hot-Potato*
  A third strategy, called hot-potato, is like single-bounce, except that jobs forwarded from another node are treated just like arrivals from outside the system (so that more than a single bounce can occur). A transfer delay is incurred by a job each time it is transferred.

If the cost of transferring a job is not negligible relative to the cost of executing a job, then both single-bounce and hot-potato may make performance worse rather than better. Note that these strategies neither assume nor require that one node have knowledge of the load at other nodes. A negative consequence of this is that sometimes jobs will be sent away from a node that is overloaded to one that is even more overloaded!

Knowledge-Based Strategy

If nodes have a way of obtaining good estimates of the current queue length at various other nodes in the system, then that information can be used to obtain better performance. A more complex strategy based on this principle is one called *receiver-initiated transfer* in which overloaded nodes try to shed jobs to underloaded nodes. The receiver-initiated transfer strategy is based on two control parameters: *threshold*, and *probe−cnt*.

The strategy works as follows (where Q_j is the number of jobs currently at node j):

Whenever an arrival occurs at node j, if $Q_j >$ *threshold*, then try to shed a job by probing *probe_cnt* other nodes (incurring a probe time for each node probed). Among the *probe_cnt* other nodes probed, let k be the index of the node with the fewest customers present. Then, if $Q_k < Q_j$, then transfer the newly arrived job from node j to node k (incurring a transfer time).

Preparation

Construct a simulation model in Csim that is capable of determining confidence intervals at a prescribed level of error tolerance on the average response times for jobs under each of the four load-sharing strategies (with any settings of the control parameters in the case of the last strategy). You may use either the *Method of Independent Replications* or the *Method of Batch Means* to determine your confidence intervals.

As test cases, investigate systems defined by the following parameters:

parameter	case 1	case 2	case 3
N	5	8	12
A	10	100	100
D	5	75	90
M	1	10	3
I	0	1	1
threshold	2	3	5
probe_cnt	1	3	5
C	20,000	20,000	20,000
α	.02	.05	.10

Think about how changes in the average cost of moving a job from one node to another (M) should influence the settings of the control parameters *threshold*. Similarly, contemplate how the average cost of obtaining load information from another node (I) affects the choice of a good value for *probe_cnt*.

Be prepared to use the models with different parameter value settings, and the comment on the relative performance achieved by the various algorithms and predicted by the model solutions.