The cost of interactive jobs on supercomputers

Allen B. Downey - Victor Hazlewood

ABSTRACT: *In some supercomputing environments, users are required to run editing, compiling and data-cleaning tasks on a workstation, and use supercomputers only for jobs that require them. This restriction is intended to improve the performance of the supercomputer, but it causes significant inconvenience for users. In this paper, we examine the workload submitted to the Cray C90 at the San Diego Supercomputer Center, and observe that ``workstation jobs'' consume less than 20% of the cycles on these machines. We conclude that the cost of supporting these jobs is small compared to the productivity improvement it provides for users.*

Introduction

Many supercomputer users have a workstation that they do not share with other users. For some applications, the performance of these workstations is as good as that of a supercomputer. Since the cost of a workstation cycle is so much less than the cost of a supercomputer cycle, it may seem like a good idea to encourage users to limit their use of supercomputers to jobs that exercise the particular strengths of the supercomputer.

Some supercomputer sites have taken explicit steps to discourage users from running interactive jobs on supercomputers. In this environment, users are expected to edit (and possible compile) their codes on workstations, execute on the supercomputer, and then perform data interpretation and various house-cleaning activities on workstations. In some cases, interactive jobs are not explicitly forbidden on the supercomputer, but the system may be configured to provide poor performance for them.

The advantage of such a system is an effective division of labor. Each job is handled by the most appropriate computer; the one that can run the job most economically. Of course, the disadvantage is that it imposes a burden on users, increases the number of steps in the edit-compile-debug cycle, and generally reduces productivity.

At the San Diego Supercomputer Center, we are trying to create an environment in which users can bring new applications to a supercomputer with minimal costs not only in terms of programming effort, but also in terms of user education and the other overheads associated with learning a new computing environment. Toward this goal, we try to provide a software environment on supercomputers that is indistinguishable from a workstation environment except in performance. Thus, supercomputers here are configured to support interactive jobs, and users are not discouraged from using them for all stages of the edit-compile-debug cycle, as well as other interactive activities.

In this paper, we will examine the impact of this decision by studying the workload running on the Cray C90 at SDSC. We would like to know what cost we are paying for supporting interactive jobs, and what would be the potential benefit of moving those jobs onto workstations. We consider two resources, CPU cycles and memory, and conclude that in both cases, the cost of supporting these jobs is small.

Allocation of cycles

We would like to know what fraction of the jobs that ran on the C90 could (or should) have run on a workstation, and what fraction of the supercomputer's time was spent on these jobs. We find that although the vast majority of jobs are ``workstation jobs,'' these jobs consume a small fraction of the total number of cycles.

We define a workstation job, conservatively, as any job that consumes fewer than two hours of CPU time, and that requires less than 128 MB[1] of memory. Our data are taken from the output of the csafef utility, which reports the resource use, duration, and other accounting information for each job submitted to the system. The particular version of this utility we use has been modified here at SDSC.

Table 1 shows a summary of the data from February, March and April of 1994, 1995 and 1996. Total CPU use is the sum of the CPU times of the jobs, reported in millions of seconds (Ms).

[1] All memory measurements in this paper are in power-of-two units, so 1 MB = 1,048,576 bytes, etc.

Utilization is just the ratio of total CPU use to the duration of the interval (three months); it does not take downtime into account.]

(Ms)			
Utilization	55%	80%	95%
Number of workstation jobs (CPU < 2hrs, mem < 128MB)	288130 (99.2%)	359359 (99.2%)	511026 (99.4%)
Total CPU use of workstation jobs (Ms)	13.4 (31.8%)	18.2 (30.0%)	14.1 (19.1%)

Table 1. CPU use by workstation jobs.

The C90 is becoming increasingly loaded over time. The number of jobs and the total CPU use of those jobs nearly doubled during the two years of our study. At the current level of utilization, the performance of the system as seen by users has begun to drop. Queue times have been increasing slowly for some time. Also, we see that the vast majority (99.4%) of jobs are ``workstation jobs'', and this fraction has been constant for some time. Thus it may appear worthwhile to move some of this workload to workstations, if possible.

The problem is that workstation jobs account for a small fraction of the C90's total time (less than 20% and falling). This fraction is not insignificant, but it is probably not possible, in practice, to shift this workload onto workstations. In the first place, our definition of a workstation jobs is deliberately generous. Secondly, we have ignored the effect a workload increase would have on workstation performance. This effect might be substantial, especially since a job that runs for 2 hours on the C90 is likely to run much longer on a workstation. Finally, we have ignored the overhead of supporting this division of labor; for example, the amount of time spent running ftp would be likely to increase in such an environment.

It is interesting to examine the characteristics of the ``supercomputer jobs'' in our workload. We divided these jobs into three groups, depending on their resource requirements. Table 2 shows how many jobs are in each group, and how much CPU time they consumed.

	1994	1995	1996
Lots of CPU (CPU > 2hrs, mem < 128MB) Number of jobs	905	1414	1384
CPU use (fraction)	60.8%	61.9%	69.5%
Large memory (CPU < 2hrs, mem > 128MB) Number of jobs	1446	1104	1513
CPU use (fraction)	3.2%	2.5%	2.5%
Both (CPU > 2hrs, mem > 128MB) Number of jobs	119	251	234
CPU use (fraction)	4.9%	5.9%	8.8%

Table 2. Classification of supercomputer jobs.

Of the supercomputing jobs, roughly half are big CPU users, and half are big memory users. But again, CPU consumption is dominated by long jobs.

Allocation of memory

The other resource workstation jobs consume is memory. Although an interactive job may not use many cycles, it does tend to be resident in memory for long periods, and may prevent some big-memory jobs from running.

In order to measure this impact, we took periodic snapshots of the c90's allocated memory every fifteen minutes for one day (Thursday, October 3, 1996). Each snapshot used ps to count the number of jobs in memory, the number of jobs in swap space, and the memory size of each job.

On average, there are 100 jobs in physical memory and 515 jobs swapped out, so there is considerable contention for the available memory. Of the 2 GB of physical memory, typically 1.9 GB are in use at a time; in other words, 95% of the memory is full 80% of the time.

What fraction of this memory is used by interactive jobs? We looked at the most common executable names and picked out the ones we could classify as shells, editors or compilers. The shells were csh, ksh, tcsh, sh and bash. The editors were vi and emacs. The compilers were cc, f90, cpp, fpp, cf77, cft77 and cft90. Of course, there are other editors, compilers, etc., but we only chose the executable names that appeared frequently enough to affect the results.

On average, the shells are using 8.3 MB of physical memory, the editors 2.3 MB and the compilers .41 MB. The total memory use of these interactive jobs is 11 MB, about 0.5% of the total. So while it would be useful to reduce memory contention, it does not appear that we can do it by expelling interactive jobs.

One of the ways UNICOS 9.0 reduces the memory use of jobs is by allowing multiple copies of the same executable to share text. For each of the executable names that appeared in our snapshots, we used which to locate the executable, and file to see if it supports shared text. Of course, many of the names in the snapshots are not in the path of the environment where we ran this experiment, so we may have missed some shared-text executables.

Nevertheless, we found the following shared-text executables: csh, ksh, sh, vi, emacs, view. In the snapshots, we counted the number of times each of these names appeared, and estimated the total memory saved by avoiding redundant text. On average, we save 5.9 MB, almost 0.3% of the total. These savings are insignificant because (1) there are few shared-text executables, (2) the executables that are shared-text tend to be small, and (3) there are seldom many copies of large executables in memory at the same time.

One exception to the last observation is mppexec, which was used to spawn parallel jobs on the Cray T3D that was (until recently) hosted by the C90. Since there was one copy of mppexec for each job on the T3D, it would have saved a significant amount of memory if mppexec had been shared-text (but it

was not). In the snapshots, the average time savings would have been 56 MB, or almost 3% of the physical memory.

Conclusions

One of the long-time goals of supercomputer centers is to facilitate the development of new applications on supercomputers by providing a productive development environment for parallel programmers. In order to address this goal, SDSC decided very early that it is necessary to support interactive jobs on supercomputers, including even some jobs that would achieve better performance on workstations (for discussion, see [2]).

In this paper, we have evaluated the costs imposed by supporting these jobs and found that (1) interactive and short CPU-bound jobs consume less than 20% of the CPU cycles, and this fraction is falling over time, and (2) the memory use of interactive jobs is insignificant, although they may account for a significant fraction of swap activity.

In order to further integrate workstations and supercomputers, SDSC has recently installed the Network Queueing Environment (NQE) from CraySoft. Hazlewood describes this installation in [1]. One unusual feature of this environment is that it provides common file space that is mounted from both supercomputers and workstations. Thus it is convenient for users to edit and compile programs on their workstations and execute them on supercomputers. Furthermore, we provide incentive for them to do so by charging higher rates for interactive jobs on supercomputers. Given this ``carrot and stick" approach, we expect the number of interactive jobs on supercomputers to decrease over time.

Ironically, we also expect to see batch jobs running on workstations. NQE provides a mechanism to run batch jobs (both sequential and parallel) on networks of idle workstations. This mechanism may make it possible to separate the workload into three components: interactive jobs, which will run on the user's workstation; small batch jobs, which will run on clusters of workstations; and large batch jobs, which will run on supercomputers. If the supercomputer workloads contain a larger fraction of large, long-running jobs, effective scheduling will be easier and it is likely that the utilization of these systems will increase.

References

1 Victor Hazlewood. Cluster computing: a survey and tutorial. In Sys Admin, pages 27-43, March 1997.

2 Wayne Schroeder and Michael Wan. Experience with the UNICOS 6.1 memory scheduler. In Proceedings of the 29th Semiannual Cray User Group Meeting, April 1992.