Workload Evolution on the Cornell Theory Center IBM SP2

Steven Hotovy
Cornell Theory Center
Ithaca, NY 14853
shotovy@tc.cornell.edu

Abstract. *The Cornell Theory Center (CTC) put a 512-node IBM SP2 system into production in early 1995, and extended traces of batch jobs began to be collected in June of that year. An analysis of the workload shows that it has not only grown, but that its characteristics have changed over time. In particular, job duration increased with time, indicative of an expanding production workload. In addition, there was increasing use of parallelism.*

As the load has increased and larger jobs have become more frequent, the batch management software (IBM's LoadLeveler) has had difficulty in scheduling the requested resources. New policies were established to improve the situation.

This paper will profile how the workload has changed over time and give an in-depth look at the maturing workload. It will examine how frequently certain resources are requested and analyze user submittal patterns. It will also describe the policies that were implemented to improve the scheduling situation and their effect on the workload.

1 Introduction

1.1 Background

There is a great deal of interest in workload characterization for parallel systems. Developers of job scheduling policies have a keen interest in the profiles of real parallel workloads as opposed to the synthetic data with which they often work. Managers of parallel systems wish to understand the workload so that they can make informed decisions about policy and procedures.

Parallel workloads are more challenging to characterize than their serial counterparts, if for no other reason than the multiplicity of processors. Some parallel systems, like the IBM SP2 at the Cornell Theory Center, contain a wide variety of different resources – memory per node, special software on fixed nodes, nodes devoted to I/O, etc. Such systems offer particular challenges for workload characterization.

Another complicating factor for parallel systems is that they are typically difficult to program, which means that program development and debugging activities will play a larger role than for serial systems. A mature workload, dominated by large production jobs, will take more time to develop on parallel systems. And even a mature workload may continue changing as users, after gaining confidence in system reliability and stability, attempt more ambitious simulations involving more processors running for longer periods of time.

To date, however, there has been very little published on actual parallel workloads. One obvious reason is that, until recently, parallel computers were used principally for research and development, not production. Nonetheless, some interesting research in workload characterization has been done. Cypher and co-workers [2] studied the memory, communication, computing and I/O resource requirements and utilization patterns of a number of scientific codes on distributed memory machines. In the Joint NSF-NASA Initiative on Evaluation (JNNIE) project [6], five NASA centers and the four NSF Supercomputer Centers identified 22 key applications and ran parallel versions of these programs on a diverse set of parallel machines. These applications involve key disciplines supported by these agencies, but are not meant to represent a typical workload. The most complete study of a production parallel workload was that done by Feitelson and Nitzberg [3]. They analyzed NQS trace records over the fourth quarter of 1993 for the 128-processor iPSC/860 at NASA Ames. In an earlier paper [7], the Cornell Theory Center (CTC) identified six classes of future key applications, and estimated the amount of resources these types of jobs would likely consume. These data represented an anticipated load, not the current workload, however. In a subsequent paper [4], the CTC analyzed the early workload on the SP2. As the SP2 has matured, this early workload has grown and changed considerably, so a more complete analysis of the maturing workload is called for.

This study highlights the evolution of the batch workload on the CTC IBM SP2 over an 8-month period. It expands on previous studies in that the workload is not viewed as a static entity, but one with several phases, each of which should be treated separately. The later, production-oriented phase is analyzed in more detail, providing information that will be useful to developers of job scheduling and resource management strategies.

This paper also examines the effect of policies on workload behavior, a subject which has received scant attention. Changes to policy were motivated by the fact that resource utilization for the later workload was being limited by inefficiencies in the resource management software. This paper discusses the motivation behind these policy decisions, and the effect they had on workload characteristics.

1.2 The SP2 System

The Cornell Theory Center IBM SP2 system [1] contains 512 nodes with an aggregate peak speed of 137 Gflops and a combined physical memory of 79 GBytes. The system, which was made available for production use in early 1995, is heterogeneous in that it contains a mixture of *wide* nodes and *thin* nodes. On the CTC SP2, *wide* nodes have been configured with 4 times the data cache and 4 times the data bandwidth between cache and memory compared to *thin* nodes. The nodes also vary in the amount of physical memory attached, from 128 MB up to 2048 MB.

The 512 processors themselves are segregated by the services they provide. The bulk of the processors, 430 in number, are devoted to running batch jobs. The remainder are used for interactive use, I/O gateways, special projects, and system testing.

The pool of batch nodes consist of both wide and thin nodes, and come in a variety of memory sizes. Table 1 details the node type/memory combinations.

The SP2 is also equipped with a High-Performance Switch, which directs data communication among the nodes. This switch runs under two protocols: the standard IP protocol (HPS-IP), and a more efficient one which resides in user space (HPS-USER).

Each node has 2GB of disk capacity. None of this storage is used for permanent files, however. Rather, this storage is used for swap space, caching of permanent files,

Node Type	128	256	512	1024	2048
Thin	352	30	0	0	0
Wide	0	22	21	4	1
Total	352	52	21	4	1
Percent	81.9	12.1	4.9	0.9	0.2

Table 1. Configuration of Batch Partition by Physical Memory (in MB) and Node Type

and temporary space for users. Permanent data are stored on external AFS file servers or on the NSL Unitree mass storage system.

Management of the batch system was provided by LoadLeveler [8], IBM's job management software. A fuller discussion of the LoadLeveler environment follows.

1.3 LoadLeveler Considerations

LoadLeveler provides a number of keywords to assist in the scheduling and execution of jobs. The most important ones from the standpoint of workload characterization are:

- Batch queue name
- The minimum number of processors required
- The maximum number of processors that can be used
- The switch protocol (USER or IP) to be used for parallel jobs
- The amount of memory requested
- The node type (THIN or WIDE)
- Mass storage (Yes or No)
- Node-locked software (ABAQUS or FIDAP)

LoadLeveler allows for dynamic definition of batch queues. Our initial definition of queues, all based on a maximum execution (wall-clock) time for a job, were:

- 15-minute
- 3-hour
- 6-hour
- 12-hour
- 18-hour

Within a given queue, there are no inherent limits to other resources, such as the number of processors, which the user may request. LoadLeveler will automatically cancel jobs that exceed the wall-clock limit for the queue, however. The LoadLeveler job scheduler is given control of assigning nodes from the entire pool of 430 processors. All queues draw from this same pool.

LoadLeveler employs space-sharing in the allocation of parallel jobs, i.e., once a node is assigned to a particular job, it is not available for further work until the job is completed. The scheduler uses a least heavily loaded algorithm to select candidate nodes. LoadLeveler imposes no limit on the number of jobs that may be queued in a given queue. LoadLeveler also employs an aging algorithm that increases the priority of jobs which have been waiting longer to receive service.

1.4 Initial Policies

The CTC established several initial policies which had an effect on batch performance. To equalize access to the SP2, the Center originally established a maximum of 2 jobs per user that could be executing simultaneously (although there was no limit per user for the number of jobs waiting in the batch queues). This limit was subsequently modified several times in response to a changing workload, as will be discussed below.

Another policy was that all batch queues, except for the 15-minute one, will be given the same initial priority. The 15-minute queue was assigned a higher initial priority because it is intended for program development and debugging where turnaround time is more critical.

2 An Overview of the Changing Workload

At the most general level, the workload can be characterized both by the number of jobs and by the amount of wall-clock time accumulated by those jobs. To quantify the latter, it is convenient to define the term **User Node Time**, which is the sum of the wall-clock times used by the job on each participating processor.

Daily LoadLeveler logs from June 18, 1995 through March 2, 1996 (except for the weeks of September 3, September 10 and December 3) were used in the analyses of this section. There was no data for the 2 weeks in September because of a problem with the data collection mechanism; data for the week of December 3 is not meaningful because much of the SP2 was dedicated to I-WAY work for Supercomputer '95.

2.1 Weekly Workload Averages

Figure 1 depicts the developing workload at this general level on a week-by-week basis. One clear trend is the consistently increasing use of the machine as measured by User Node Time through the beginning of October. Through October and November, however, the User Node Time oscillates between 40,000 and 50,000 node-hours per week despite that fact that maximum capacity is about 70,000 node-hours. The peak of 50,000 corresponds to 70 percent of capacity and the average of about 42,000 node-hours over these 2 months is only 60 percent. One can also observe a slow decline in User Node Time in 1996. This is caused by the fact that some of the nodes previously assigned to servicing batch jobs were taken away for special projects.

The number of jobs exhibits a different behavior, though. It varies between 2,000 and 3,000 jobs through mid-October, but then begins to decline, despite the fact that the User Node Time remains at its peak. At the end of 1995, the number of jobs settles down to about 1,000 per week, a factor of 3 decrease from the high in late August. 1996 shows a slight increase to about 1500 jobs weekly.

Figure 1 leads one to suspect that:

- The workload profile began to change in September and continues to change;
- The consistent, relatively low average utilization of about 60 percent may be due less to the the workload itself than to shortcomings in the system's ability to service the workload.

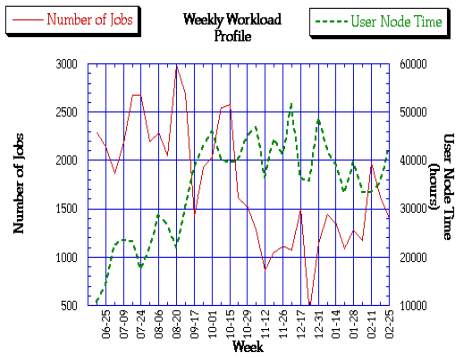

Fig. 1. Weekly Workload.

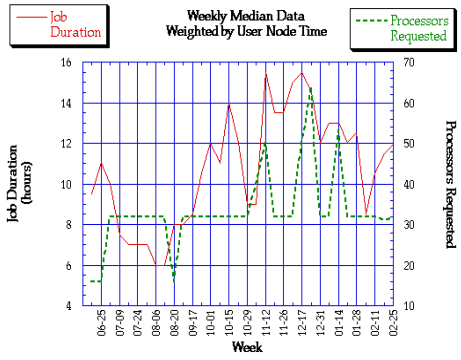

Fig. 2. Median Job Duration and Processors Requested.

2.2 The Changing Workload Hypothesis

Figure 2 lends credence to the first hypothesis. The solid line represents the median job duration weighted by User Node Time. A median value of 11 hours for the week of June 25th means that half the User Node Time comes from jobs which run in 11 hours or more. This value dropped throughout the summer months to a low of 6 hours in mid-August, at which point it began a steady increase to about 15 hours late in 1995. This is an increase of 250 percent over a 3 month period and reflects a rapid maturation in the workload from a code development/modest production environment to one dominated by large production jobs.

Figure 2 also shows the median number of processors requested weighted by User Node Time. In most cases, the median value is 32 processors, primarily because 32-way jobs account for about 30 percent of the total User Node Time in a given week. Yet on several occasions, the median value reached 50 and then 60 processors. This shows that not only are jobs running longer, they tend to be using more processors on the average. Taken with the average job duration data, this explains why the number of

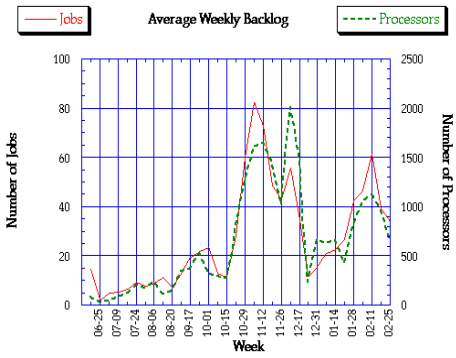

Fig. 3. Average Weekly Backlog.

jobs dropped from nearly 3,000 in August to 1,000 later in 1995 while the aggregate node-hours nearly doubled over the same period.

2.3 The System Limitation Hypothesis

One indication that system response, not the inherent workload, is the limiting factor would be a consistent backlog of jobs waiting to be serviced. Figure 3 displays the weekly average backlog, both by job and by processors requested. The backlog gradually increased through mid-September, at which point it rapidly accelerated, reaching a peak of 80 jobs and over 2000 processors late in 1995. After declining in December during the holidays, the backlog started a steady increase in 1996. While this was only a weekly average, it does suggest that instances where there were no jobs queued to run occurred very infrequently.

Figure 4, which shows wait time information, adds more support for the hypothesis. Through the middle of September, the average wait times for both the 15-minute queue and the production queues (all the rest) were relatively low. For both, the median wait times were significantly lower than the mean, typically less than 1 minute. During this period, one could characterize the wait time distribution as one highly skewed toward very short wait times, with a few exceptions which lasted a very long time (at times, more than a day).

For the next month, wait times fluctuated, never going beyond 200 minutes. From mid-October to mid-November, however, the wait times grew enormously to over 800 minutes. Wait times for the 15-minute queue settled down through the rest of the reporting period, while for the production queues it maintained a level of around 400 minutes.

There is one other notable change in the wait time data. From late October on, the median wait times for the production queues can be as much as 50 percent of the mean. This suggests a wait time distribution very different from the earlier one – a much larger percentage of people are faced with long wait times. This is further evidence that the system is unable to deliver more than 50,000 node-hours per week for the existing workload.

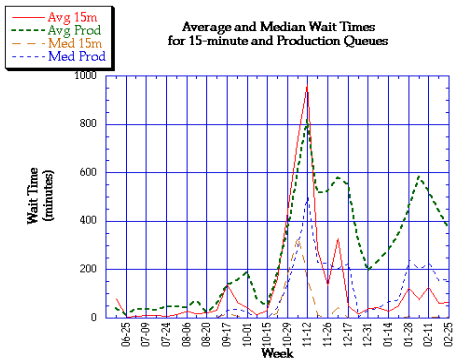

Fig. 4. Average and Median Wait Times.

2.4 Notes on Further Wait Time Analysis

The wait time data should be viewed with some care. Scarce resources – large numbers of processors, the single mass storage node, etc. – will have an effect on wait times. The CTC policies regarding the maximum number of simultaneously executing jobs and initial job priorities also influence wait times. LoadLeveler itself can cause certain jobs to wait a very long time – hence the disparity between median and mean wait times in the first half of the reporting period. In addition, user behavior can be a factor. For example, during periods of long turnaround times, a few users would submit LoadLeveler jobs and then immediately place them in Hold status. While the jobs were in Hold status (and increasing in priority because of aging), users would prepare their input files. When the files were ready, they would release the job, which would then get quick service because of its high priority. This action will create an artificially high wait time for the job.

3 Characterization of the Maturing Workload

As was mentioned above, the workload began to change in September from a code development/modest production environment to one featuring more substantial production jobs. In this section, we will analyze the workload for the time period from September 17th through December 2 in more detail, highlighting its heterogeneous nature.

3.1 Requests for General Resources

Table 2 describes the demand for the varying resources. Only 2.4 percent of the jobs requested the single mass storage node, but these jobs constitute 7 percent of the user-node hours. Thus larger jobs, which tend to generate larger amounts of I/O, make use of the mass storage feature. Wide nodes are requested by roughly 1 of every 8 jobs. This is a surprisingly high number, and probably reflects those applications for which CPU performance is enhanced by the larger data cache and greater cache-memory

bandwidth of the wide nodes. These jobs account for less than 5 percent of the user node time, however. Since there are fewer than 50 wide nodes, such jobs by necessity will be serial or modestly parallel. Hence they will generate less User Node Time than the typical job. In fact, only 35 percent of the User Node Time for wide nodes come from parallel jobs.

Resource	No. Jobs	Pct. Jobs	No. UNH	Pct. UNH
Mass Storage	427	2.4	32284	7.0
Wide Nodes	2303	12.8	24409	4.7
Node-locked S/W	80	0.5	4	0.0
0-128MB Mem	15099	84.1	441175	95.8
129-256MB Mem	1987	11.1	16220	3.5
257-512MB Mem	652	3.6	2296	0.5
513-1024MB Mem	100	0.6	314	0.07
1025-2048MB Mem	109	0.6	654	0.14

Table 2. Requests for Resources for Maturing Workload

3.2 Requests for Memory

The breakdown of memory requested is very illuminating. About 85 percent of the jobs, consuming over 95 percent of the user-node hours, requested 128MB of memory or less. There is corresponding less demand for nodes with larger amounts of main memory. It should be noted that memory requested reflects demand for resources, not necessarily what type of nodes were actually used for a particular job. LoadLeveler will try to use nodes with 128MB of memory to service a smaller job, but should there an insufficient number of 128MB nodes available, it will engage nodes with more memory.

One of our concerns is whether the varying resources of the SP2 are adequate to the demands of the workload. In this light, it is helpful to compare the usage of nodes by memory in Table 2 with the configuration in Table 1. One can see that the 128MB nodes are relatively over-requested since they account for about 82 percent of the nodes available but nearly 96 percent of the user-node hours. The single 2GB node is requested proportionally to its contribution of user-node hours. The remaining nodes, however, are definitely under-requested. One reason for this is the relatively small number of large nodes, which discourages significant parallel use of them. In fact, parallel jobs requesting more than 128MB of memory account for a miniscule 0.3 percent of all parallel User Node Time.

3.3 Requests for Processors

Another critical resource is sufficient nodes for parallelism. Figures 5 and 6 display the demand for particular numbers of processors. Several items stand out. One is that serial jobs constitute a non-negligible portion of the workload. In fact, over half the submitted jobs are serial, although they account for only 8.6 percent of the user-node hours. These serial jobs are, for the most part, legacy applications from the IBM

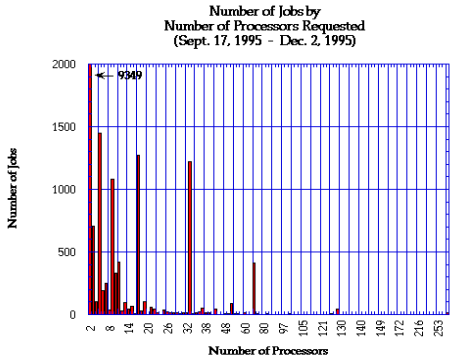

Fig. 5. Number of Jobs by Number of Processors Requested.

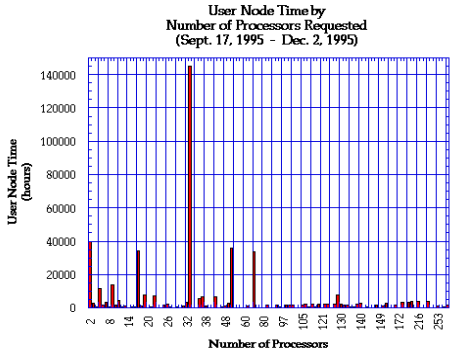

Fig. 6. User Node Time by Number of Processors Requested.

ES/9000 which was taken out of operation last year. Third-party applications, notably computational chemistry, also contribute to the serial load.

Another striking feature is the amount of work done by jobs requesting a number of nodes that are a power of 2, especially 32-way jobs. Power-of-2 jobs constitute about 55 percent of the User Node Time, 31 percent of which is due to 32-way jobs alone. It is not clear why power-of-2 jobs should be so prevalent – the SP2 does not place any restrictions on the number of nodes that can be requested, and its flat topology does not cause degraded performance for certain node combinations that other topologies (hypercube, 2-D mesh, etc.) do. It may be that the users want portability with other parallel systems which do have limitations; it may also be that powers of 2 naturally fit with the phenomenon being modeled.

3.4 Job Duration Profiles

The correlation of job duration with the number of processors requested is a matter of some interest to job schedulers. Figure 7 plots the average job duration as a function

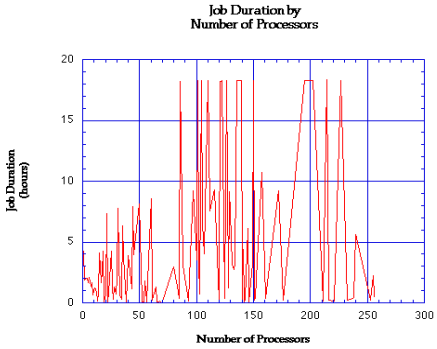

Fig. 7. Job Duration by Number of Processors Requested.

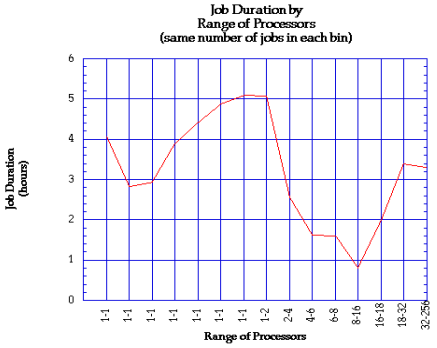

Fig. 8. Job Duration (binned) by Number of Processors Requested.

of number of processors requested. The shape is very erratic and correlations, if any, are difficult to perceive.

One problem with this presentation of data is that it is not weighted by the number of jobs requesting a given number of processors. As was mentioned above, over half the jobs for the CTC maturing workload are serial. One can compensate for this by creating bins with equal numbers of jobs, in order of increasing numbers of processors. The results for 15 bins is shown in Figure 8.

The first 7 bins are for serial jobs only. The data was processed in chronological order, so these bins reflect temporal changes in the average jobs duration of serial jobs from mid-September through early December. It is clear that after a brief drop-off in late Septmber to a low value of about 3 hours, the mean job duration increased steadily to its high of over 5 hours at the end of the period.

The next set of bins, from 2-4 processors through 8-16 processors, shows a steady decline in job durations. One suspects that jobs in this range are indicative of program development and debugging, with correspondingly lower run times. The last set of bins shows an increase in job durations with a leveling off for the very largest jobs. Jobs in

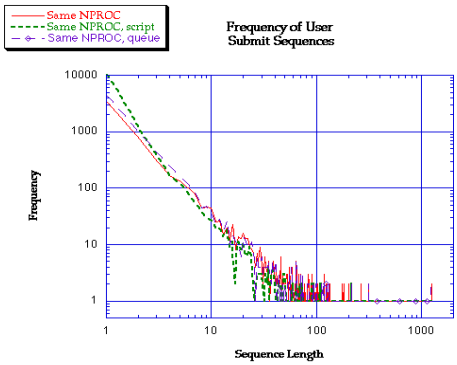

Fig. 9. Frequency of User Submit Sequence Lengths.

this range represent production use of the machine and account for the vast majority of cycles.

3.5 User Job Submittal Sequences

Another important measure of the workload is the degree to which users submit sequences of similar jobs. In our case, we define 3 different classes of "similar" jobs for a given user:

1. Same number of processors requested
2. Same number of processors requested and same batch script
3. Same number of processors requested and same LoadLeveler queue

The results appear in Figure 9. As one would expect, the least restrictive definition of similar jobs (definition 1) exhibits the fewest short sequences and the most long sequences. Definition 3 shows only a slight difference from the least restrictive, while the requirement that there be the same batch script generates far more sequences of length 1 and consistently fewer for sequences 5 or greater.

The most striking aspect of the data, however, is the virtually linear decrease up to sequences of about 50 in length, regardless of definition. Since the plots are log-log, it suggests a Zipf's distribution. Based on linear regression for the first 50 datapoints, the coefficient for the Zipf's distributions for each of the three user sequence definitions was -1.85, -2.1 and -2.0 respectively.

4 Policy and Procedures

Throughout most of the reporting period, CTC policies remained as they were in the beginning. One modification was made early – to encourage use of the machine while the load was still relatively modest, the limitation of 2 simultaneously running jobs per user was raised to 4 in August.

4.1 New Batch Class

As the wait times started their dramatic increase in early November, the CTC considered several policy changes. Two were implemented. On November 20, a new batch class for short-running jobs was created. This queue imposed a limit of a 15-minute run-time and a maximum of 8 processors that could be requested, and 8 thin nodes were reserved for this queue. The purpose of this queue was to provide faster turn-around for small program development and debugging jobs. The effect was pronounced. The mean wait time dropped from 965 minutes for the single 15-minute queue the previous week to an average of 280 minutes for both queues the following week. The median wait time dropped even more dramatically, from 147 minutes the previous week to 16 minutes the next. And from that time forward, wait times for the 15-minute queues have remained relatively low.

This decrease in wait time is due in large part to the new batch class. During the 2 weeks immediately following the activation of the new batch class, there were 367 jobs in the standard 15-minute queue with an average wait time of 309.4 minutes. As for the new class, there were 262 jobs submitted with an average wait time of 17.9 minutes, a factor of 17 less than for the standard queue. Thus these smaller jobs received much better service than if they had to contend with larger jobs in the standard queue.

4.2 Reduction in Number of Simultaneously Running Jobs

With the increase in demand for the SP2 and resulting contention for resources, there was a concern about equitable access to the machine. Thus it was decided to review the policy of a maximum of 4 simultaneously running jobs. An analysis of the data for the two weeks of maximum backlog (November 5 through November 18) showed that over 10,000 node-hours, or 12.4 percent of the total volume, came from jobs whose users already had at least 2 other jobs running at the same time . This number was deemed excessive, so on November 26 the number of simultaneously running jobs for a given user was reduced from 4 back down to 2.

4.3 Future Directions

Inefficiencies in the current LoadLeveler job scheduler are a major cause for the long wait times and relatively low ceiling of 50,000 node-hours. To improve this situation, the CTC is planning to incorporate the EASY scheduler [5] into its environment. This scheduler offers several advantages:

- It requires much less overhead to acquire nodes
- It is deterministic and can thus inform users when their job will run
- As the workload progresses, it will update the time when jobs will run, shortening the time to execution if possible
- It has the ability to backfill with smaller jobs

Work is underway to interface EASY with the LoadLeveler API, thus minimizing the impact on our users. At first, EASY will be given a portion of the nodes to manage jobs which require no special resources. However, as EASY is enhanced to provide full resource management, it will replace the LoadLeveler job scheduler completely.

5 Conclusions

During the period of time when batch trace records have been analyzed, the workload on the CTC IBM SP2 has undergone a significant change. Through mid-September, it can be characterized as one featuring program development and debugging with a certain amount of modest production work. From that time onward, however, production computing has become more prominent. The median job duration, weighted by node-hours generated, increased from 6 hours in July to a peak of 15 hours in November. The median number of processors used also increased.

The workload data shows an upper bound of about 50,000 user-node hours consumed per week, despite the theoretical maximum of over 70,000. An examination of average weekly backlog data shows an increase from 250 nodes in early October to over 2000 in early December. During that same period, average wait times for the 15-minute queue rose from less than 50 minutes to about 900 minutes. All this is evidence that the upper bound is driven by system limitations in servicing the workload, not in the workload itself.

During the period of maturing workload (mid-September through early December), we observed that the 128MB nodes are most heavily requested in proportion to their numbers. The single 2GB node is also well used, but the remaining nodes are relatively under-requested. We also observed a sizable serial workload, primarily a legacy from a prior IBM mainframe.

A look at job duration profiles showed that jobs requesting 2-16 processors had relatively shorter run times than those requesting more processors. This is likely due to the fact that these smaller jobs reflect program development, whereas jobs with more processors are doing production work. User submittal patterns were examined, and it was found that the frequency of similar job submittals followed a Zipf's distribution.

The paper detailed several policy changes made in response to the changing workload and backlog of work to be done. Adding a new 15-minute queue for small parallel jobs had a dramatic, positive effect on the wait times. There was also a discussion of plans to incorporate the Argonne EASY scheduler into the CTC environment, with the hopes that this will increase the throughput of the machine.

References

1. T. Agerwala, J.L. Martin, J.H. Mirza, D.C. Sadler, D.M. Dias and M. Snir, "SP2 System Architecture". *IBM Systems Journal*, Vol. 34, No. 2, 1995.
2. R. Cypher, A. Ho, S. Konstantinidou and P. Messina. "Architectural Requirements of Parallel Scientific Applications with Explicit Communication". In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, May, 1993. p. 2-13.
3. D.G. Feitelson and B. Nitzberg. "Job Characteristics of a Production Parallel Scientific Workload on the NASA Ames iPSC/860". *IPPS'95 Workshop on Job Scheduling Strategies for Parallel Processing*, April, 1995.
4. S.G. Hotovy, D.J. Schneider and T. O'Donnell. "Analysis of the Early Workload on the Cornell Theory Center IBM SP2". In *Proceedings of ACM SIGMETRICS Conference*, 1996 (to appear).
5. D.A. Lifka. "The ANL/IBM SP Scheduling System". *IPPS'95 Workshop on Job Scheduling Strategies for Parallel Processing*, April, 1995.

6. W. Pfeiffer, S. Hotovy, N.A. Nystrom, D. Rudy, T. Sterling and M. Straka. *JNNIE: The Joint NSF-NASA Initiative on Evaluation*. San Diego Supercomputer Center Technical Report GA-A22123, July, 1995.
7. M.E. Rosenkrantz, D.J. Schneider, R. Leibensberger, M. Shore and J. Zollweg. "Requirements of the Cornell Theory Center for Resource Management and Process Scheduling". *IPPS'95 Workshop on Job Scheduling Strategies for Parallel Processing*, April, 1995.
8. *IBM LoadLeveler Administration Guide*, IBM Document Number SH26-7220-02, October, 1994.