Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization

James Patton Jones[1] and Bill Nitzberg[1]

MRJ Technology Solutions
NASA Ames Research Center, M/S 258-6
Moffett Field, CA 94035-1000

nitzberg@nas.nasa.gov
Tel: (650) 604-4513

Abstract

The NAS facility has operated parallel supercomputers for the past 11 years, including the Intel iPSC/860, Intel Paragon, Thinking Machines CM-5, IBM SP-2, and Cray Origin 2000. Across this wide variety of machine architectures, across a span of 10 years, across a large number of different users, and through thousands of minor configuration and policy changes, the utilization of these machines shows three general trends: (1) scheduling using a naive FIFO first-fit policy results in 40-60% utilization, (2) switching to the more sophisticated dynamic backfilling scheduling algorithm improves utilization by about 15 percentage points (yielding about 70% utilization), and (3) reducing the maximum allowable job size further increases utilization. Most surprising is the consistency of these trends. Over the lifetime of the NAS parallel systems, we made hundreds, perhaps thousands, of small changes to hardware, software, and policy, yet, utilization was affected little. In particular, these results show that the goal of achieving near 100% utilization while supporting a real parallel supercomputing workload is unrealistic.

---

1. Work performed under NASA contract NAS2-14303, Moffett Field, CA 94035-1000

1.0 Introduction

The Numerical Aerospace Simulation (NAS) supercomputer facility, located at NASA Ames Research Center, serves in the role of pathfinder in high performance computing for NASA. In the late 1980s, we began exploring the use of highly parallel systems for supporting scientific and technical computing [Bai91]. Today, it is commonly accepted that "supercomputing" is synonyms with "parallel supercomputing".

Supercomputing means running "big" jobs or applications which cannot be run on small or average sized systems. "Big", of course, is a relative term; we generally consider a job "big" if it is using at least half of the available resources of a "big" system. (We leave the definition of "big system" to the reader.)

Traditional vector supercomputers (e.g., the Cray C90) are capable of sustaining nearly 100% utilization while supporting "big" jobs and running a varied workload [Car95]. Our experience has shown that this level of utilization is not attainable for running a supercomputing workload on a parallel supercomputer.

2.0 The NAS Parallel Supercomputing Workload

The NAS facility supports research and development in computational aerosciences. Hundreds of research projects are funded annually which use the parallel supercomputers at NAS to perform high-end scientific and technical computing. Over the past 11 years, the NAS parallel workload, priorities, and approach have been consistent.

The workload consists of a mix of:

- 100's of users; new users are constantly added
- scientific and technical computing of aerospace applications
- code development, debugging, scaling and performance analysis
- "production" runs of existing applications

Most of the applications run at NAS are statically balanced (applications which require a well-balanced load across all nodes). Statically balanced applications strive to give an equal amount of work to each processor. A single slow process in a statically load-balanced application can completely ruin the performance of the application, as other processes will have to wait for it. Another issue arises from message passing synchronization. Even if we overlay parallel jobs to avoid load balancing problems, tightly synchronized applications can incur an extra synchronization delay for messages because processes are not *gang-scheduled* (scheduled to run at the same time across all their assigned nodes). These restraints are consistent across typical parallel supercomputing workloads. (For further discussion of parallel supercomputing workloads, see [WLMFN96].)

At the same time, the NAS scheduling policy has consistently strived for (in order of priority):

1. Overnight turn-around for "big" jobs, and
2. Good machine utilization.

The first priority supports "supercomputing", the second supports efficient use of resources. NAS supports supercomputing by favoring supercomputer-sized jobs ("big" ones, typically those that cannot run on any other system within NASA) over smaller jobs. In general, the system configuration, user allocations, and scheduling policies are tuned so that big jobs get overnight turn-around.

In apparent contrast to the first priority is the second. Good machine utilization has historically meant 99% on traditional vector supercomputers, and the stakeholders (those who's money purchased the machines) have traditionally used utilization as a metric for success. As we show, parallel supercomputing does not achieve 99% utilization.

The system configuration and the mechanisms by which we let users run jobs, has also been consistent throughout the past 11 years. The systems are all space-shared (partitioned), and batch scheduled. Interactive use is permitted, but it must take place by allocating resources via the batch system, then using those resources interactively. This approach to using parallel computers has prevailed, despite the availability of good time sharing and gang-scheduling facilities on several systems for two reasons: consistency of timings and efficiency of execution. Analysis of algorithms, exact megaflop rates and scaling, is a major component of the NAS workload. Approaches other than strict, partitioned space sharing, don't support this. Furthermore, systems such as the Intel Paragon and Cray Origin 2000 suffer from an interference problem, in which it is possible for jobs to "overlap" in such a way as to slow each other down by far more than would be expected by the simple sharing of resources.

3.0 Analysis of NAS Parallel Supercomputer Utilization

In this section we describe the hardware configuration of five NAS parallel supercomputers and discuss batch job scheduling and utilization for each. (For a discussion of batch job scheduling requirements for large parallel supercomputer, like those at NAS, see [STT95].) During the analysis, several trends begin to emerge. These are discussed as they become apparent in the data presented.

3.1 Intel iPSC/860 (Jan. 1990 to Sept. 1994)

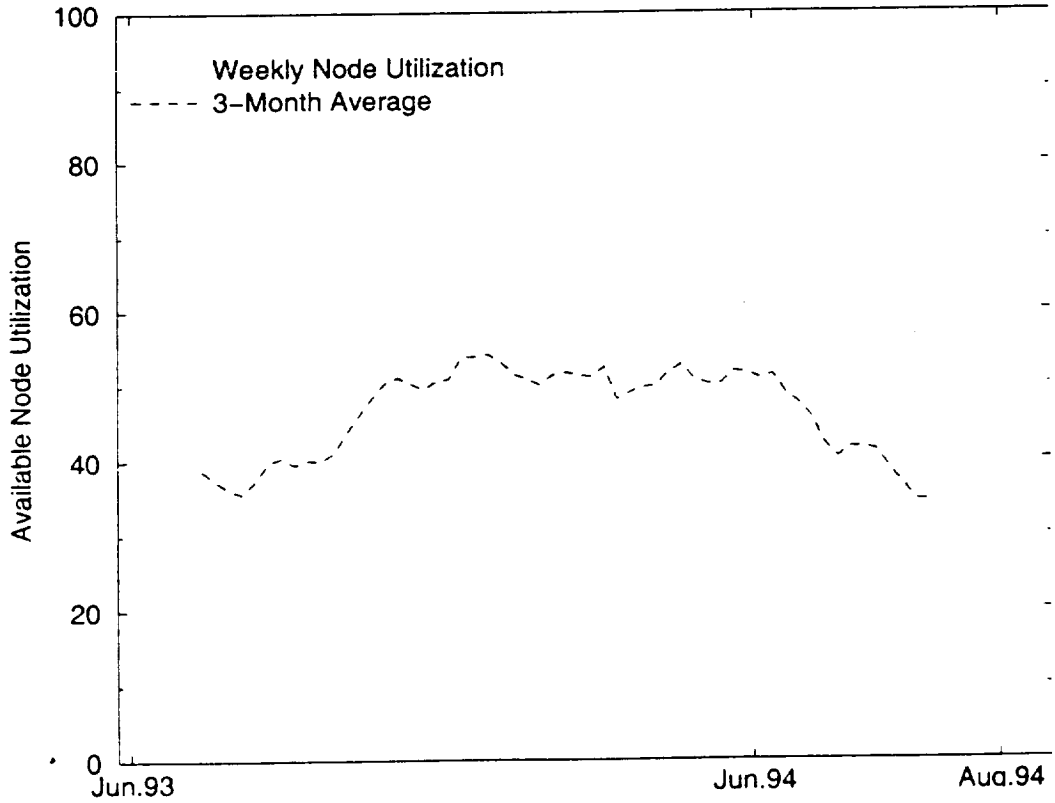
The Intel iPSC/860 at NAS had 128 compute nodes (each with a single 40 MHz i860 XR processor and eight megabytes of physical memory). The Network Queueing System (NQS, [Kin85]) was used as the batch system, implementing queue-level "first-in first-out first-fit" (FIFO-FF) scheduling with different size

priorities during the day (i.e. big jobs had priority at night, small jobs during the day).

The FIFO-First-Fit algorithm works as follows: batch jobs are evaluated in FIFO order in the queue, i.e. oldest job first. For each job, the batch system first checked if there were enough nodes available to run the job, and if so, then compared the job requirements (walltime and node count) to the current scheduling policy. If either of these two checks failed, the scheduler skipped to the next job. If both are successful, the scheduler ran the job and removed it from the list. This process continued until all the jobs were evaluated.

Scheduling on the iPSC/860 was relatively simple, as the system itself was very inflexible. The architecture divided the system into "partitions" each of a power-of-2 number of nodes. Batch jobs were then run in the smallest possible partition size. This made scheduling easier, but forced idle time when running medium sized jobs. For example, a 65-node job could only run in a 128-node partition. Since there was no time-sharing available, this forced the remaining 63 nodes to be left idle. Furthermore, there existed a system limit of a maximum of ten concurrent partitions. This limit also had the potential for forcing idle time, even when there was a backlog of work. For example, if the iPSC/860 was running ten 2-node jobs, the remaining 108 nodes would be idle. But given the typical job size in the NAS workload, the maximum partition limit was rarely exceeded. (The system ran out of nodes well before it allocated 10 partitions.)

Figure 1: iPSC/860 Utilization

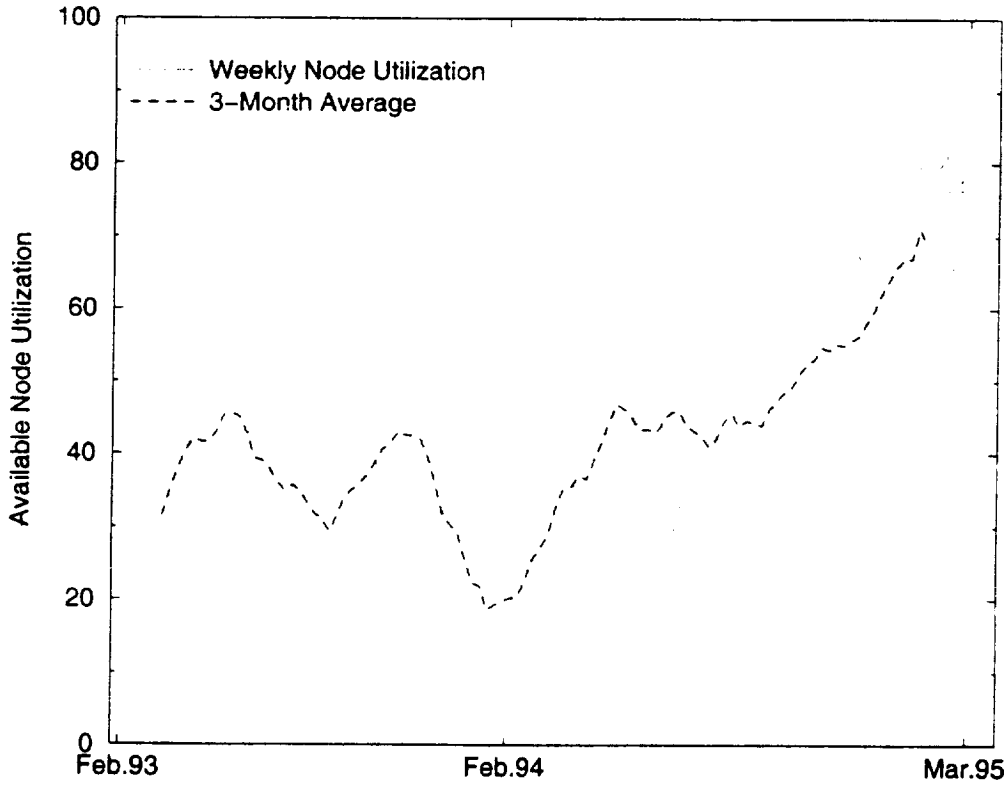


The iPSC/860 was fairly unreliable during the first two years at NAS. The first year the system was thoroughly investigated by NAS staff, during which time a variety of benchmarks were developed and run on the system. Figure 1 shows the node utilization starting in mid-1993. (Full accounting data for the first two years is unavailable.) At the time, the utilization shown was considered an impressive improvement over that of previous years, and is primarily attributable to two factors. The first being a significant increase in system stability. Secondly, in early 1993, users had begun to shift from application debugging to running their codes as "production" batch jobs. Notice that the utilization ranged between 40 and 60 percent, for most of the period shown.

3.2 TMC CM-5 (Jan. 1993 to Mar. 1995)

The Thinking Machines Corporation's (TMC) CM-5 at NAS had 128 compute nodes (each with one 33 MHz SPARC processor, four vector units, and 32 megabytes of physical memory). The CM-5 was scheduled using the Distributed Job Manager (DJM) which also implemented a size-priority FIFO-FF algorithm that was time-of-day sensitive.

Figure 2: CM-5 Utilization



Like the iPSC/860, the CM-5 architecture restricted all partitions to a power-of-2 number of nodes. However, the CM-5 further restricted the partition size to a minimum of 32-nodes, and the partition size could not be changed without a reboot of the entire system. During the day, the CM-5 was run with one 64-node

partition and two 32-node partitions. Each night, the system was reconfigured into a single 128-node partition to allow large jobs to run.

The CM-5 followed quite a different life-cycle compared to the iPSC/860. Initially only NAS staff had access for benchmarking and evaluation purposes, and to work to stabilize the system. But rather than taking several years like the iPSC/860, we had to wait only several weeks before putting "real" users on. Figure 2 shows how quickly the scientists put the CM-5 to work. Part of the reason for the short ramp-up was that most of the researchers were migrating to the CM-5 from the previous generation CM-200 (which had been previously upgraded from a CM-2) at NAS. Many of these users already had codes that ran well on this architecture.

Like the iPSC/860 much of the CM-5's final year's usage increase was due to users completing the debugging cycle, and moving to running production codes. Halfway through the second year of the system's stay at NAS, in an effort to increase the utilization of the machine, space sharing was relaxed on the small partitions during the day to allow two jobs to timeshare within a given partition. Doing so resulted in a gradual increase of utilization, however, it so resulted in a 20 percent slowdown in both timeshared applications.

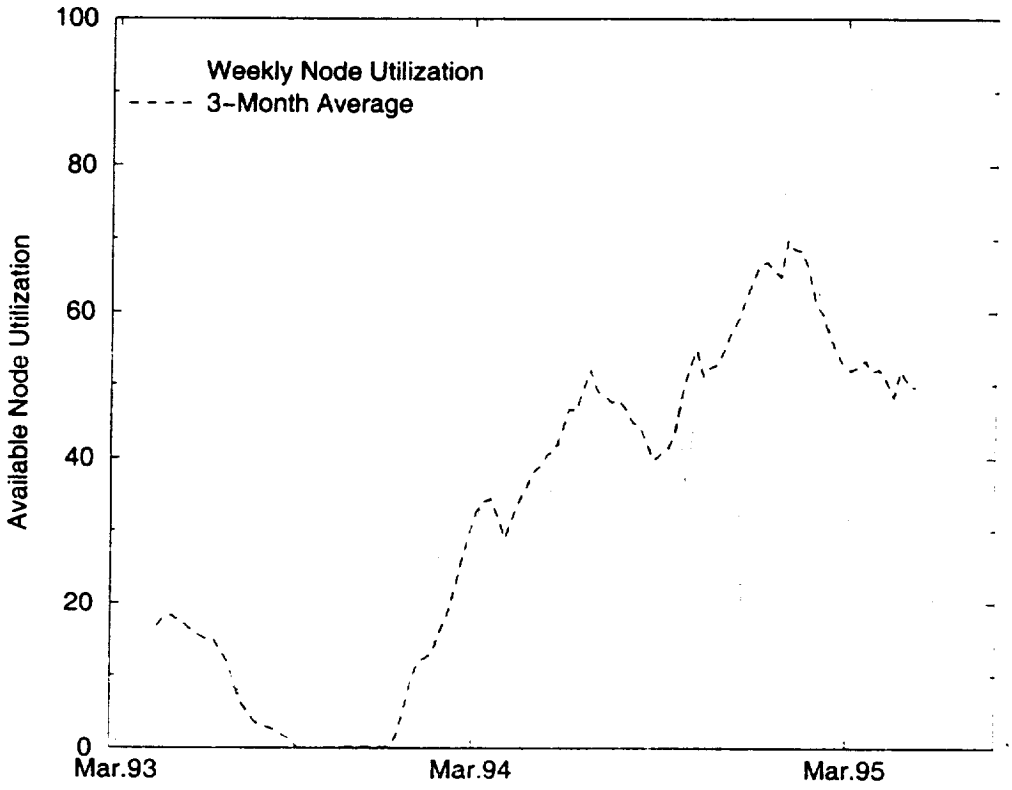
3.3 Intel Paragon XP/S-15 (Feb. 1993 to July 1995)

The Intel Paragon XP/S-15 at NAS consisted of 208 compute nodes (each with two 50 MHz i860 XP processors and 32 megabytes of physical memory). Using NQS, we implemented queue-level FIFO-FF scheduling with different size priorities, like on the iPSC/860. Scheduling the Paragon, however, was more difficult than the previous systems because power-of-2 job sizes were no longer required. The resulting wide variety of job sizes decreased the scheduling efficiency.

The Paragon, like the CM-5, had a relatively short shake-out period before we started adding users onto the system. These were primarily users from the iPSC/860 who wanted to try out the new system. Once on, many chose to return to the iPSC/860 until the Paragon stabilized. Unfortunately, the system was never completely stable.

The utilization shown in Figure 3 for the first half of 1993 is based on UNIX SAR (system activity report) and load average data. Some data for 1993 was lost (thus the apparent zero usage). Following this, the MACS accounting software was installed, enabling more accurate utilization tracking. This is also the time when the remaining iPSC/860 users began to migrate over to the Paragon in order to continue their work. (Compare the iPSC/860 and the Paragon utilization graphs in Figure 6 to more clearly see the drop in the older system corresponding to an increase in the newer system.)

Figure 3: Paragon Utilization



The periodic dips in the Paragon utilization correspond to regular frequent operating system upgrades and testing. During these times, though the system was available to users, many opted off the machine to avoid the frustrations of trying to use a system in flux. From the utilization graph, we see that the FIFO-FF algorithm maintained the utilization between and 40 and 60 percent for most of the "production" lifetime of the system.

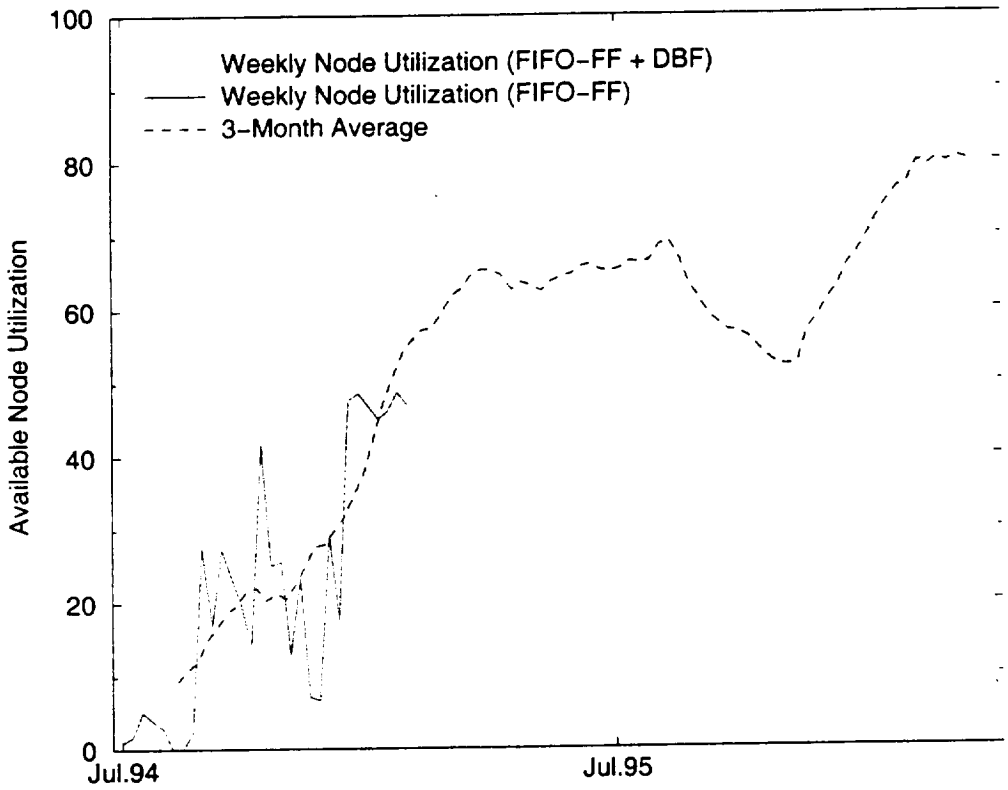
3.4 IBM SP-2 (July 1994 to Sept. 1997)

The NAS IBM SP-2 was a 160-node system with 150 compute nodes. Each node was an IBM RS6000/590 workstation powered with a single 66.7 MHz POWER2 processor and at least 128 megabytes of physical memory. (Six of the nodes had 512 megabytes of memory). The system arrived with IBM's LoadLeveler batch system, which provided simple FIFO scheduling.

Based on what we had learned with previous parallel systems, we predicted that a FIFO-FF scheduling algorithm would result in a system node utilization of around 50 percent. However, the Loadleveler batch system used a simple FIFO algorithm which was achieving roughly 25 percent utilization. After six months, we replaced LoadLeveler with the Portable Batch System (PBS) utilizing a FIFO-FF algorithm [Hen95]. System utilization immediately doubled (see Fig-

ure 4), averaging 50 percent. This level of utilization continued the entire time we used the FIFO-FF scheduler.

Figure 4: SP-2 Utilization



As soon as we installed PBS, we began implementing a dynamic-backfilling (DBF) algorithm [THJ98, FW98]. In the algorithm, the scheduler finds the highest priority job which should be scheduled next. The backfilling algorithm is designed to efficiently drain the system, and to reserve nodes for the top-ranked job. The First-Fit algorithm could continually take away nodes from a waiting large job to run a smaller job. Draining the system is an expensive operation as a large number of nodes may need to be kept idle to ensure that a particular job can run. For example, suppose we want to run a 128-node job, but there are only 127 nodes available, and there is a 5-hour job running on the last node. We have to keep 127 nodes idle for 5 hours to guarantee that this job will run. While this simplistic approach works, it is obvious that it does not lead to the best system utilization possible. A better solution is to have the scheduler recognize that we will not be able run our job for 5 hours, but we can use the 127 nodes to run any jobs that can complete in 5 hours, using a *backfilling* method. *Static-Backfilling* fixes the starting time of the high-priority job at the earliest time possible (i.e., the earliest time when the required nodes will be available). In our previous example, this was 5 hours. A *Dynamic-Backfilling* algorithm, rather than fixing the time at the earliest time that it can, will instead determine the most appropriate time to start the job within a starting window. The earliest time possible may in some cases not lead to the best result. Using our previous example, let's

assume that we had also a 125-node job (for 5 hours 30 minutes) queued. Using static-backfilling, we could not run this job as we only have 5 hours to backfill. But with dynamic-backfilling, we should recognize that by shifting the starting time by 30 minutes, we will be able to fill 125 nodes, significantly increasing resource utilization. The Dynamic-Backfilling algorithm attempts to continually balance both the need to run high-priority jobs, and the need to maintain as many nodes as possible in-use.

Approximately two months after installing PBS, its FIFO-FF scheduler module was replaced with the DBF scheduler module. (This DBF scheduling module is now included in the PBS distribution.) System utilization again jumped, this time roughly 20 percentage points, to 70 percent. Over time, as users began to run fewer debugging jobs, and started scaling up the problem size of their codes, the utilization slowly crept up to 80 percent. The system continued at this level until it was allocated to the NASA Metacenter project, which substantially changed the usage, scheduling model and workload flow of the system. (Discussion of the Metacenter project and its scheduling experiments is included in [Jon96, Jon97].)

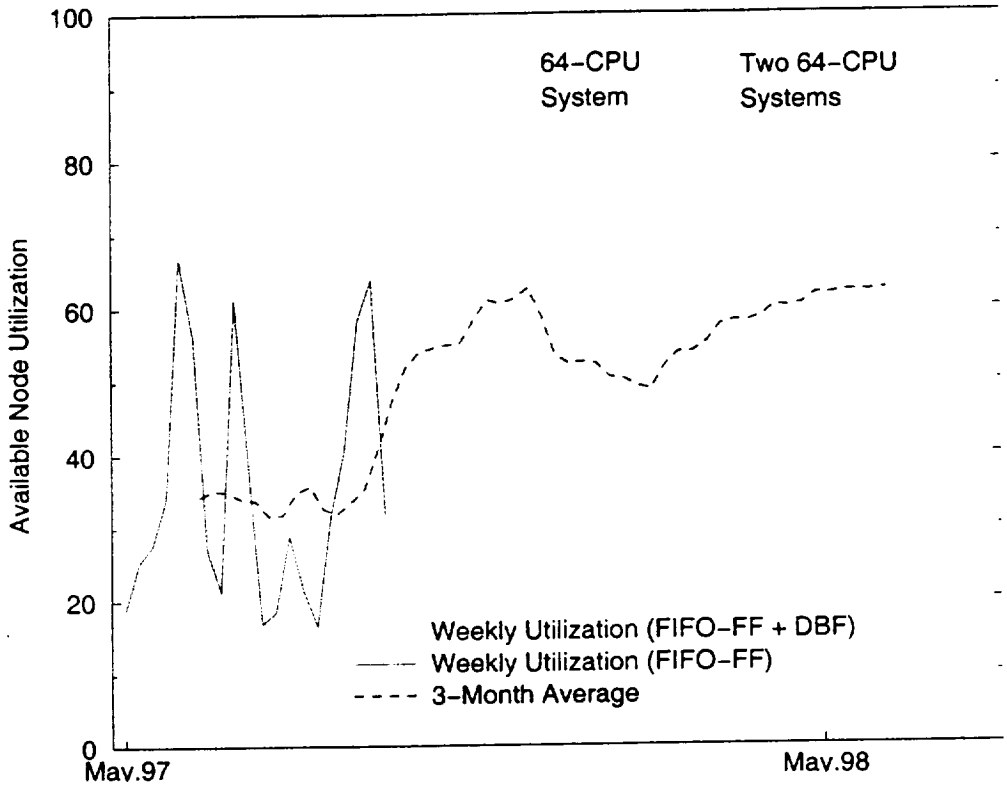
3.5 Cray Origin2000 (Jan. 1997 -)

In January 1997, NAS received its first 32-processor SGI Origin2000. (System larger than 32-processors receive the "Cray" label.). The Origin2000 is a distributed shared memory (DSM) system, with each building block being a "node board". Each node board in an Origin2000 possess two MIPS RISC R10000 64-bit processors, and a configurable amount of memory (512 MB on the NAS systems). Each node is attached to a modified hypercube network.

One of the most useful features of the Origin2000 is the single system image. Users can utilize the system as a large SMP rather than having to be concerned with the distributed nature of the architecture. So when the system first arrived, we decided to see if we could schedule it like a true timeshared symmetrical multiprocessor (SMP). We installed PBS with a FIFO-FF SMP scheduler that we had been running on several Cray J90s. Even with its single system image, trying to schedule this distributed memory system quickly proved problematic, as the interference between jobs resulted in severe performance degradation and varied runtimes. We quickly turned to another solution. We switched to software partitioning of the system, where we assigned sets of processors to specific partitions. Each partition was assigned an SGI NODEMASK which identified which nodes belonged to that partition. (A NODEMASK is similar to a "processor set", except it is node-based rather than CPU-based. While not a perfect solution, the NODEMASK capability proved quite functional, even though it was only advisory to the kernel. Some of this functionality will be made available in the SGI MISER kernel scheduler.)

In March 1997, we doubled the size of our Origin2000 system, creating a 64-processor parallel supercomputer running under a single system image. Figure 5 shows the utilization starting with the installation this system..

Figure 5: Origin2000 Utilization

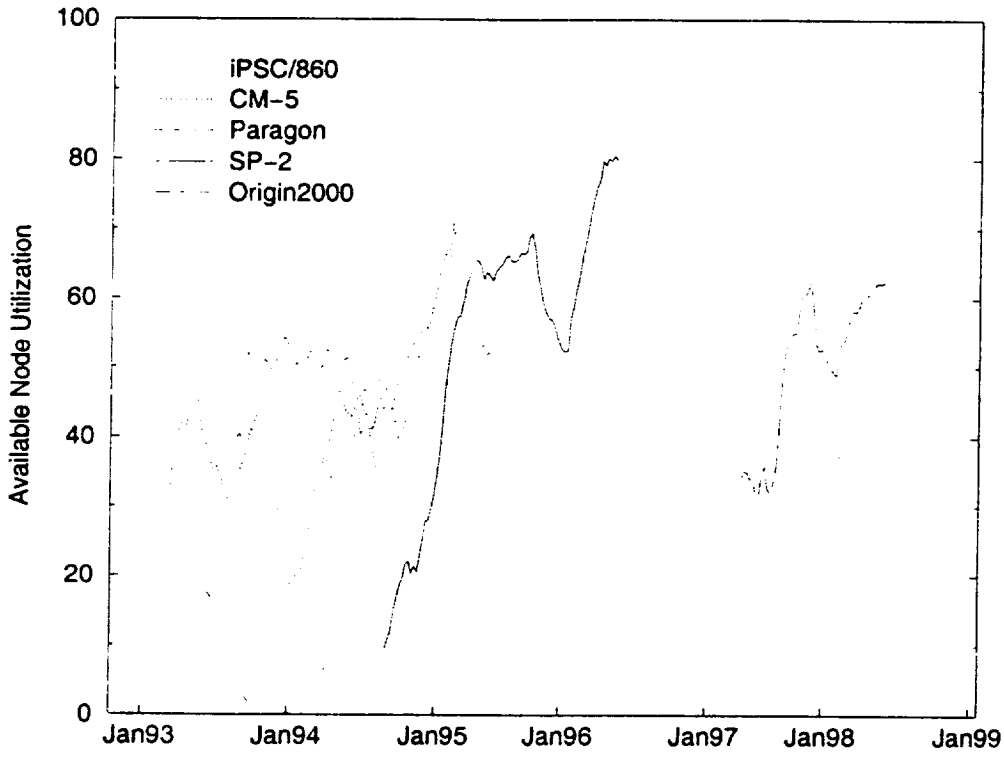


We scheduled each of the partitions within the Origin2000 as we did on the previous parallel systems. Figure 5 shows the system utilization that resulted: a rough average of 35 percent. From our previous experience, we predicted adding dynamic backfilling to the scheduling algorithm would add another 20 percentage points to the utilization. Our prediction was again borne out: average system utilization increased from about 35 percent to about 55 percent. Nearly a year later NAS purchased a second 64-processor Origin2000 system. Rather than being run as a separate system, it was configured to share the workload of the first system.

Another trend we have noticed is that if you reduce the size definition of "big jobs" in relation to the maximum number of nodes available, utilization will increase. We predicted that by adding this second 64-processor system to the compute pool (thereby doubling the number of processors available for computation) but maintaining the maximum job size at 64-processors, utilization should increase. We were surprised at how smoothly the utilization curve in Figure 5 transitioned between doubling the resources, without increasing the maximum job size.

[Note to reviewers: Figure 5 ends at the point of installation of two additional 128-processor Origin2000 systems. Given that the full user base was not given immediate access, there is insufficient data to make a fair comparison between these new system and those already reported. However, by mid-November, these two system were merged into the first 256-processor Origin2000, and the full NAS parallel system user base was given access. By the due date for the final draft of this paper, there will be sufficient data to present, with a 12-week floating average, the utilization of that system as well. Preliminary data indicates that previously mentioned trends will be borne out with this system as well.]

Figure 6: Comparison of Parallel Supercomputer Utilization



In order to make comparison of the various systems utilization graphs easier, a composite graph is included below. Figure 6 shows the lifetime utilization of each of the five MPP systems, along a single timeline. It is useful to compare the arrival and decommission of the systems with each other. From such a composite graph, it is easier to see that the iPSC/860 utilization began dropping as the Paragon started rising.

4.0 Caveats and Conclusions

First, a caveat. There are hundreds of variables which contribute to effective scheduling of a parallel supercomputer. We have ignored all but one—the scheduling algorithm—in our analysis. Further, we make some broad, sweeping gener-

alizations, which are heavily dependent on the user workload of the system. Although experience has shown that the NAS facility is a prototypical supercomputing center, one should not totally discount the possibility that the unique features of this facility contribute to these results.

Data gathered over 11 years operating parallel supercomputers (including the Intel iPSC/860, Intel Paragon, Thinking Machines CM-5, IBM SP-2, and Cray Origin 2000) shows three distinct trends:

- scheduling using a naive FIFO first-fit policy results in 40-60% utilization,
- switching to the more sophisticated dynamic backfilling scheduling algorithm improves utilization by about 15 percentage points (yielding about 70% utilization), and
- reducing the maximum allowable job size increases utilization.

Most surprising is the consistency of these trends. Over the lifetime of the NAS parallel systems, we made hundreds, perhaps thousands, of small changes to hardware, software, and policy. Yet, utilization was affected little. In particular, these results show that the goal of achieving 100% utilization while supporting a real parallel supercomputing workload is currently unrealistic...and it doesn't matter what system you use, or who your users are, or exactly the method used for partitioning resources.

5.0 Future Work

The "goodness" of a scheduling algorithm and policy is hard to quantify. Although utilization is the best metric available, it is still inadequate. Utilization does not take into account properties of the workload which, even if optimally scheduled, would not give 100% utilization. Such workload parameters as job-size mix, arrival rate of jobs, efficiency of applications, etc., are ignored. A better metric is needed. The utilization we report is simply the amount of time processors are assigned out of the total time processors are available (i.e., we only ignore system down time). We would like to refine utilization to be based not on the total uptime of a system, but on the optimal scheduling of the given workload.

Seven years of job accounting data for five different parallel supercomputers is a lot of data; we have only scratched the surface. We would like to analyze "big" job turn-around times in a similar fashion as we have done for utilization. Further, we would like to investigate correlations between system stability (crashes), user load, turn-around time, workload characteristics, utilization, and, if possible, system culture (e.g., night time vs. day time, conference deadlines, etc.).

6.0 References

[Bai91] David H. Bailey, *Experiences with Parallel Computers at NASA/Ames* **NAS Technical Report RNR-91-007**, NAS Facility, NASA Ames Research Center, February 1991.

[Car95] Nick Cardo, *Batch Scheduling: A Fresh Approach*, in **Proceedings of Cray's User Group Conference**, March 1995.

[FW98] Dror Feitelson and A. Mu'alem Weil, *Utilization and Predictability in Scheduling the IBM SP2 With Backfilling*, in **Proceedings of 12th International Parallel Processing Symposium.**, pp. 542-546, Apr 1998.

[Hen95] Robert Henderson, *Job Scheduling Under the Portable Batch System*, in **Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing**, Santa Barbara, CA, April 24-29 1995.

[Jon96] James Patton Jones, *The NASA SP2 Metacenter*, in **Proceedings of the Computational Aerosciences Workshop**, High Performance Computing and Communication Program, NASA Ames Research Center, August 1996.

[Jon97] James Patton Jones, *Implementation of the NASA Metacenter: Phase 1 Report*, **NAS Technical Report NAS-97-027**, NAS Facility, NASA Ames Research Center, October 1997.

[Kin85] B. Kingsbury, *The Network Queuing System*, Sterling Software, Palo Alto, 1985.

[STT95] Bill Saphir, Leigh Ann Tanner and Bernard Traversat, *Job Management Requirements for NAS Parallel Systems and Clusters*, in **Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing**, Santa Barbara, CA, April 24-29 1995.

[THJ98] Bernard Traversat, Ed Hook and James Patton Jones. *A Dynamic-Backfilling Algorithm for Efficient Space-Sharing Scheduling on an IBM SP2*, **NAS Technical Report NAS-98-101**, NAS Facility, NASA Ames Research Center, November 1998.

[WLMFN96] K. Windisch, Virginia Lo, R. Moore, Dror Feitelson, and Bill Nitzberg, *A Comparison of Workload Traces From Two Production Parallel Machines*, in **Proceedings of 6th Synp. Frontiers of Massively Parallel Computing**, pp. 319-326, Oct 1996.