Source of Acquisition
NASA Goddard Space Flight Center

A QUEUE SIMULATION TOOL FOR A HIGH PERFORMANCE SCIENTIFIC COMPUTING CENTER

Carrie Spear, Computer Sciences Corporation
James McGalliard, FEDSIM

Abstract:

The NASA Center for Computational Sciences (NCCS) at the Goddard Space Flight Center provides high performance highly parallel processors, mass storage, and supporting infrastructure to a community of computational Earth and space scientists. Long running (days) and highly parallel (hundreds of CPUs) jobs are common in the workload. NCCS management structures batch queues and allocates resources to optimize system use and prioritize workloads. NCCS technical staff use a locally developed discrete event simulation tool to model the impacts of evolving workloads, potential system upgrades, alternative queue structures and resource allocation policies.

1. INTRODUCTION

This paper describes a queue simulation tool in use at the NASA Center for Computational Sciences (NCCS), a high performance computer facility at the Goddard Space Flight Center in Greenbelt, Maryland. The tool simulates the behavior of batch job queues, the batch job scheduler, workloads and the systems they run on and predicts performance metrics of interest such as expansion factors, wait times, system utilization, and throughput. Using the tool, systems staff balance the conflicting objectives of low expansion, low wait time, high utilization and high throughput in an environment of high latent demand.

The paper is structured as follows:

- Section 1 describes the NCCS environment, workloads, systems, batch job queue structures and performance metrics.
- Section 2 discusses alternative job scheduling algorithms of interest to the NCCS and high performance computer (HPC) centers generally.
- Section 3 describes the queue simulation tool, including its design, validation, and example tool inputs.
- Section 4 provides insights into system and workload behavior gained using the tool, and compares our results with prior studies.
- Section 5 has a summary and future plans.

1.1 NCCS ENVIRONMENT

Goddard is a major center for NASA's Science Mission Directorate and is home to the nation's largest community of Earth scientists and engineers. Goddard's missions include expansion of knowledge of the Earth and its environment, the solar system, and the universe through observations from space. The Hubble Space Telescope Control Center is on the Goddard campus, and Goddard is a design center for Earth-observing satellites and other spacecraft. Goddard is also the home of the NCCS.

The NCCS is a supercomputing facility that provides Goddard's science community with HPCs, mass storage, network infrastructure, software, and support services. About 600 scientists use the NCCS to increase their understanding of the Earth and space through computational modeling and processing of space-borne observations. NCCS systems are targeted to the specialized needs of Earth and space scientists and NASA's exploration initiative.

NCCS performance management was the subject of a 2003 CMG paper [Glassbrook].

1.2 NCCS WORKLOADS

The largest NCCS supercomputer workloads are mathematical models of the Earth's atmosphere, oceans, and climate. Another large workload is data assimilation, which processes Earth-observing satellite data and other sparse climate data and generates models of the global climate that are the best fit of current data.

Examples of other workloads include the following:

- 3D Modeling of High Energy Emission from Rotation-Powered Pulsars
- 3D Simulations of Accretion to a Star with Magnetic Field
- Assimilation of Satellite Observations of Clouds to Improve Forecast Skill
- Gravity Wave Simulations
- Global Magnetohydrodynamic Simulations of the Solar Wind in Three Dimensions

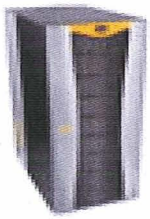
And many others.

1.3 CURRENT NCCS SYSTEMS

NCCS supports these computational science workloads using a suite of high performance systems that currently include:

- Palm/Explore, an SGI Altix 3700 BX2 with a total of 1,132 CPUs

PALM/EXPLORE



palm:
SGI Altix 3700 BX2
128 CPUs
272 GB memory

explore:
SGI Altix 3700 BX2
1 512 CPU, 2 256 CPUs
1 1024 GB memory, 2 512 GB memory

- Discover, a Linux Cluster with 1,536 CPUs
- Courant, an SGI Origin 3800 with 128 CPUs
- Dirac, an SGI Origin 3800 with 64 CPUs, used as a file server
- Disk drives from Data Direct Networks and other vendors
- Tape robots from Sun/StorageTek

This paper focuses on the Palm/Explore production machine.

1.4 JOB SCHEDULING ON PARALLEL SYSTEMS

As is typical of high performance systems, many NCCS workloads simulate the behavior of their object of inquiry by representing it in a multi-dimensional grid. Global climate models can be mapped onto current-generation highly parallel systems by dividing the atmosphere into groups of three-dimensional cells, assigning each group to a processor node, simulating climate behavior within that group on that node, and periodically synchronizing across nodes. Users specify the number of processing nodes that the application uses when they submit the job.

The dispatching software of traditional systems optimize processor utilization by assigning a new job to a processor at certain times, such as the start of an input/output operation or the expiration of a time slice. Current high performance systems, where there may be upwards of 1,000 or 10,000 processors in the complex, often do not perform as well under this job control method. Swapping a job out on one processor may leave 60 or 120 other processors idle. Instead, processors are allocated to a parallel job for the duration of the job and are not reassigned until the job completes.

The ability of a parallel job to use the processing resources it receives depends on the characteristics of that job. Amdahl's law analyzes a job's ability to use multiple processors depending on the proportion of parallel vs. serial code that it executes. In an environment like the NCCS, systems staff can provide advice and training on the optimal use of parallel processor allocations, but the primary responsibility for exploiting parallelism remains with the application programmer.

On the other hand, the system administrators and batch job scheduling software can manage parallel processing resources to optimize the assignment of jobs to CPUs. Ideally, few processing nodes sit idle waiting for work if there are jobs waiting in the queue. The objective is to allocate jobs onto idle processors consistent with the users' workload priorities, resource allocations and performance needs.

1.5 NCCS WORKLOAD PARALLELISM

As noted above, NCCS batch jobs are allocated to specific processors dedicated to those jobs

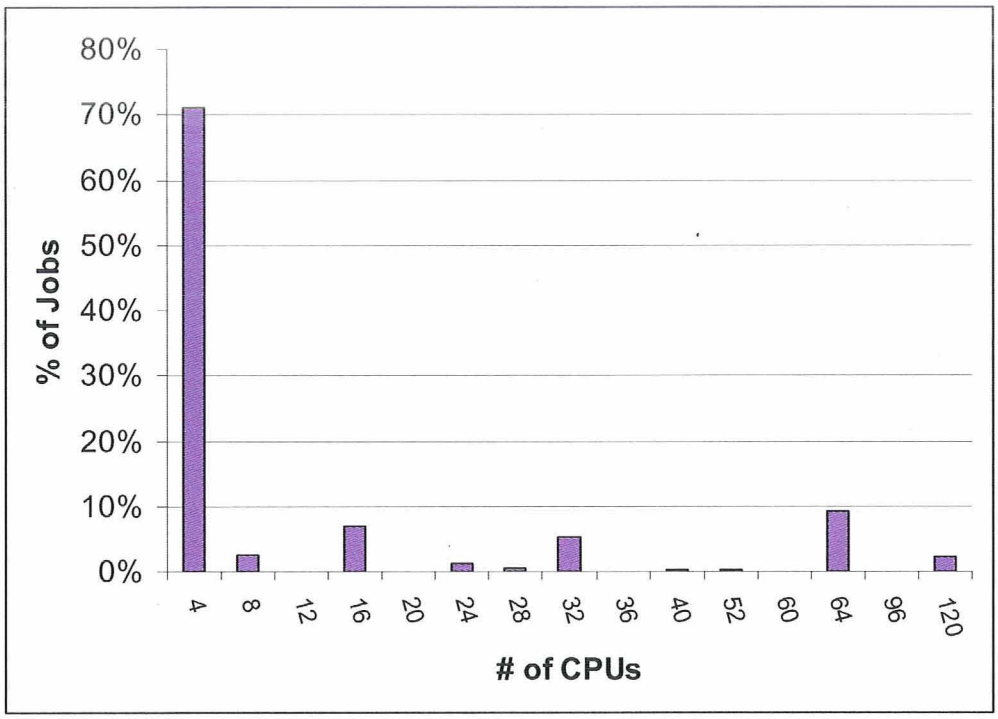


Exhibit 1 – Total Jobs by CPU for All Sectors for April 2007

Small jobs (4 or fewer CPUs) are by far the most numerous…

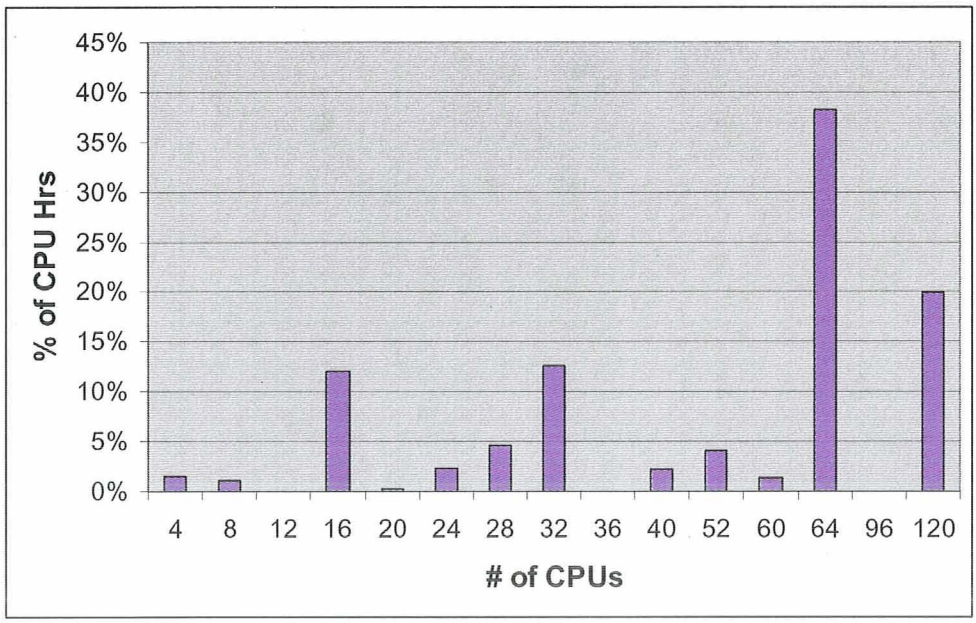


Exhibit 2 – Total CPU Hours by CPUs for April 2007

but 64- and 120-processor jobs consume the majority of the processor resources for this period.

until they complete. Jobs vary in the number of processors they use and the length of time the job runs. Exhibits 1 and 2 show the Palm/Explore workload profile for April, 2007. The charts show that, although highly parallel jobs are less numerous than small (4 CPUs or less) workloads, they consume the majority of processing resources.

1.6 NCCS QUEUE STRUCTURE

NCCS uses a system of workload queues to provide optimal performance for our workloads and systems. System administrators structure the queues along functional lines, based on historical patterns, knowledge of user requirements and management policies. The current queue structure is depicted in Exhibit 3 and is taken from an NCCS web site. (Note: Queue and user names throughout this paper have been modified due to system security security policy.)

Batch Queues on the SGI Altix 3000

Read the documentation below the table to familiarize yourself with the restrictions on some of the queues below. All queues except checkout and prepost will run on e1, e2, and e3. The default number of CPUs for any job is 2, and the default wall clock time is 5 minutes.

Special Queues on the SGI Altix 3000 System

	Up to 1 Hours	**Up to 3 Hours**	**From 1 to 12 Hours**	**Longer than 12 Hours**
Up to 16 processors	archive max 2 proc/job and 2 hour maximum	prepost-(5), palm	allpurp_small-(5), Maximum 18 processors/job	
Up to 32 processors	checkout-(5), palm			
From 2 to 254 processors			allpurp-(5) 16 to 254 processors background (1), 4 hour maximum	
More than 254 processors				allpurp_hi-(6) high_priority-(7)

Numbers in parentheses designate priority, with 7 being the highest priority and 1 being the lowest.

Exhibit 3 – Current NCCS Queue Structure on Palm/Explore

checkout

Routed specifically to the system's front end, palm
Time constraint of 1 hour maximum per job
No more than 4 jobs in this queue can be run by the same user at the same time.

archive

This queue is to be used for the purpose of moving data (data archival jobs, staging jobs, no MPI/OpenMP processing).
Only 2 jobs allowed per user at one time to run.
Job size is limited to 2 processors.
There are only 10 processors in total set aside for this queue.
Jobs in this queue will run on the backend systems (not palm, but rather e1, e2 or e3).
If users want to use this queue they need to specify the queue name in their qsub parameters **"-q archive"** on the command line or **"#PBS -q archive"** in the job script itself.

prepost

This queue is to be used for pre/post processing work (not production runs).
Jobs now limited to no more than 16 processors
If users want to use this queue they need to specify the queue name in their qsub parameters **"-q prepost"** on the command line or **"#PBS -q prepost"** in the job script itself.
No more than 6 jobs in this queue can be run by the same user at the same time.

allpurp_small

No more than 5 jobs in this queue can be run by the same user at the same time.
Job size is limited to 18 processors.

Minimum wallclock time has been lowered to 1 hour by default (so the queue allows 1-12 hours walltime).

allpurp

No more than 8 jobs in this queue can be run by the same user at the same time.

Allows jobs sized between 16 processors and 254 processors.

Exhibit 3, cont'd. – Current NCCS Queue Structure on Palm/Explore

background

Designed to have the lowest priority on the system and is targeted for two groups:
NCCS staff
Users who have used up all their allocated hours on the system
This queue is only turned on if there isn't any work waiting in the high_priority, allpurp_hi or allpurp queues.
Note that users may still run jobs in the background queue even if they have a current allocation; use of this queue will not count against that allocation amount.

allpurp_hi

Designated for users who have a demonstrated need for either very large or very long jobs
Must be approved by NCCS staff, the NCCS Director, or NASA HQ
Total number of processors available (and maximum number of processors per job) in this queue is 510

urgent

Reserved for use on jobs designated by NCCS staff, the NCCS Director, or NASA Headquarters as needing priority within the overall workload
Total number of processors available (and maximum number of processors per job) in this queue is 510

Exhibit 3, cont'd. – Current NCCS Queue Structure on Palm/Explore

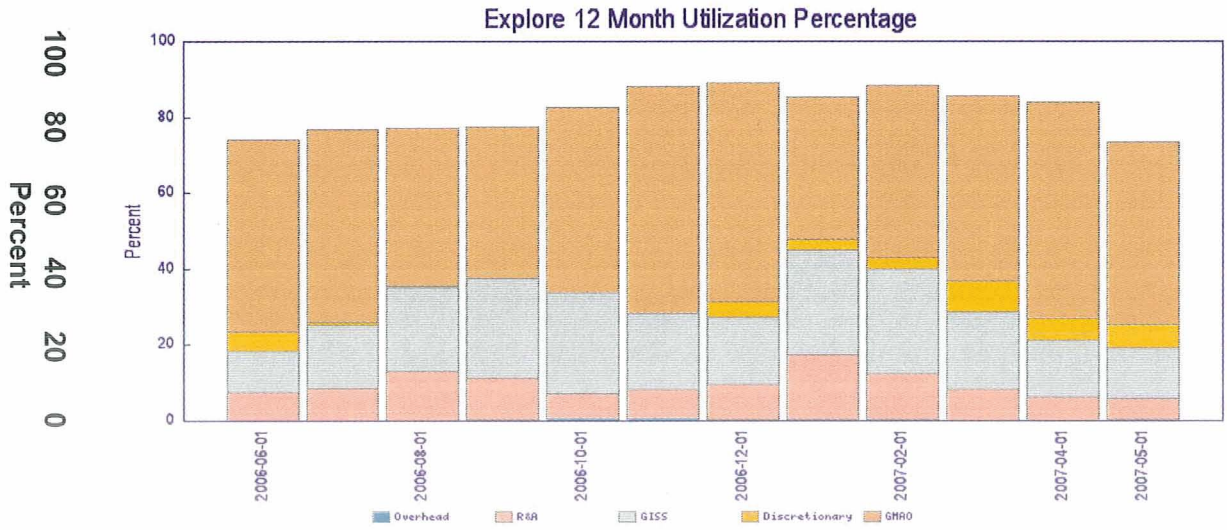


Exhibit 4 – Palm/Explore Utilization, 1 Year

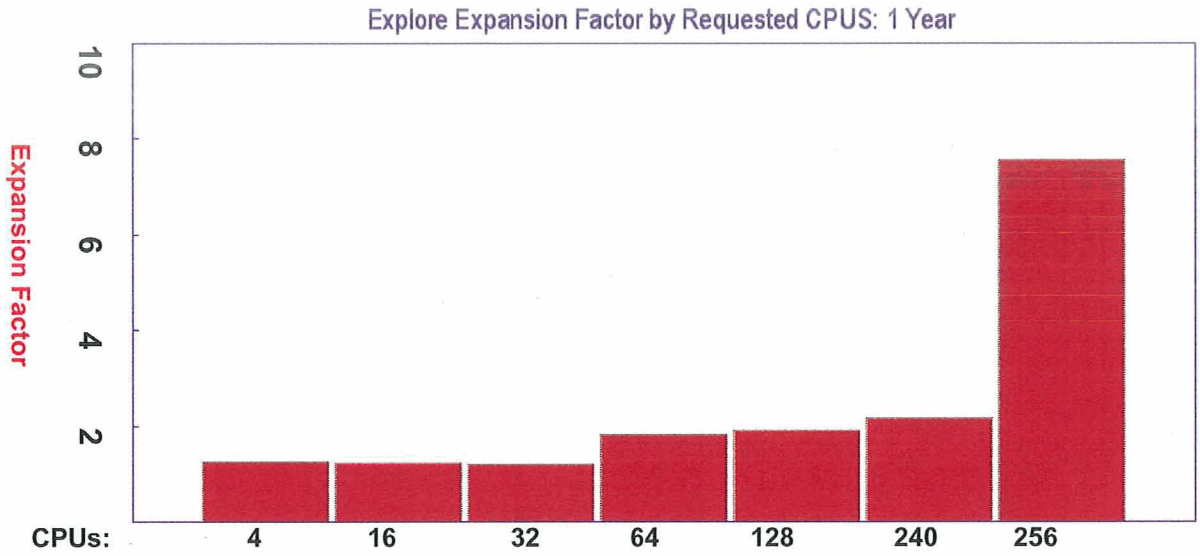


Exhibit 5 – Palm/Explore Expansion Factor, 1 Year

1.7 PERFORMANCE METRICS

Many commercial computer operations manage performance using service level agreements based on response time percentiles, reliability, throughput, or hardware resource utilization, but high performance data centers are somewhat different. Batch typically dominates HPC data center workloads (like NCCS), and has turnaround times that vary from seconds to days or even weeks. In an environment of CPU-bound workloads running on dedicated nodes, a key performance metric for system administrators is the expansion factor – the percent increase in job completion time due to queue waiting vs. time spent running on the allocated nodes.

Expansion Factor = (Run Time + Queue Time)/Run Time

The NCCS also measures processor utilization and divides this in two parts: (part 1) the percent of available nodes allocated to some job over time, and (part 2) the utilization of allocated nodes by running jobs. Utilizations over 90% in terms of allocation (part 1) are typical, and depend on the job scheduling algorithm, queue structure, and job mix. Part 1 utilization is the focus of the simulation and this paper. Processor utilizations within allocations (part 2) below 20% for running jobs are also typical. These low numbers are due to Amdahl's law (proportion of serial vs. parallel code) and other factors. Part 2 utilization is outside the scope of this paper.

The NCCS queue simulation tool estimates both the expansion factor and processor utilization at the allocation (part 1) level. Exhibits 4 and 5, above, show recent Palm/Explore utilization and expansion.

2. PARALLEL JOB SCHEDULING ALGORITHMS

NCCS uses the Portable Batch Scheduler (PBS) from Altair to manage the Palm/Explore batch workload (there are competing products that offer comparable functionality). When a user submits a job (*qsub-*) to a queue managed by PBS, the operating system (SUSE Linux in this case) turns over management of that job to PBS. Linux will not swap a PBS job during its execution. PBS controls most of the CPUs on the system.

Parallel job scheduling generally works as follows:

- The system administrators structure queues with priorities, job time and processor count limits, and user or other functional queue target workloads. (Section 1.6 shows the current queues on Palm/Explore.)

- Users estimate job run time and specify the number of processors the job requires. Run time estimates are required to implement backfilling, which is discussed in the next section. NCCS uses backfill.

- The job scheduling system allocates jobs to processors based on the number of CPUs required, estimated time required, priority, queue, and the other jobs currently executing or awaiting execution on that system.

Other considerations also impact job scheduling. At the NCCS, users receive annual allocations of processor time. Once they exceed their allocation, their jobs will no longer run. Other factors include access to non-CPU resources, such as mass data storage and compiler and library versions.

2.1 BACKFILL AND RELAXED BACKFILL

In parallel systems, swapping the job running on a single processor would result in wait states on all the other allocated CPUs. In practice, it makes more sense to dedicate all of the processors to a particular job until the job completes. (Alternative methods, such as gang scheduling [Feitelson] [Frachtenberg], exist and are useful in some environments, but are beyond the scope of this paper and are not relevant to the NCCS.) With dedicated processor allocations for the life of a job, optimization of system utilization becomes an issue of intelligently assigning jobs to groups of CPUs in such a way that utilization and throughput are high and wait time is low. Many HPC sites use backfill to improve utilization, and there has been extensive discussion of this method in the literature (see, e.g., [Ward], [Feitelson], [Shmueli], [Hovestadt]).

Exhibit 6 is a simple example of parallel job scheduling without backfill. In the diagram, the vertical axis represents multiple processors, the horizontal access represents time, and each box represents a job. Each job occupies a certain number of processors for a certain amount of time. In the example, job J1 is running and the next highest-priority job, J2, waits until J1 completes. Lower priority jobs J3 and J4 wait until higher priority J2 is finished, even though there is unused capacity on the system that J1 does not need.

In Exhibit 7, Strict Backfill has been added to the job scheduling algorithm. ("Backfill" and "Strict Backfill" are used interchangeably in the literature.) Because J3 can use the idle processors that J1 does not need and will complete before J1, backfill allows lower priority J3 to jump forward in the queue and run to completion before J2 starts. J4 must wait, however.

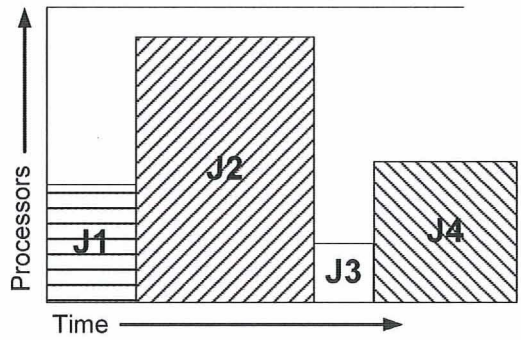


Exhibit 6 – No Backfill

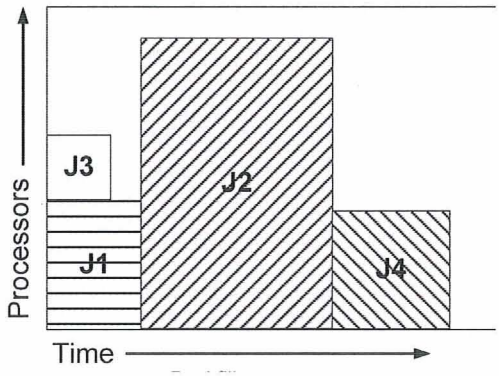


Exhibit 7 – Strict Backfill

In *strict* backfill, no lower priority job can delay any higher priority job. With *relaxed* backfill, some small delays to the higher priority jobs can be tolerated if the global utilization and throughput of the system can be improved thereby. Exhibit 8 illustrates this situation. Relaxed backfill allows both jobs J3 and J4 to jump ahead of higher priority J2, even though the execution time of J4 exceeds J1 and causes J2 to be delayed slightly.

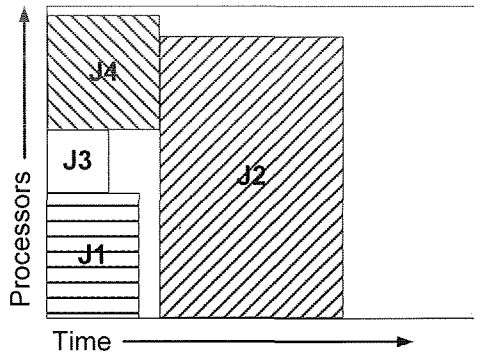


Exhibit 8 – Relaxed Backfill

The delay tolerable by a higher priority job is a tunable parameter in the job scheduler.

There are many variations on backfill [Ward], [Shmueli].

2.2 OTHER SCHEDULING CONSIDERATIONS

Backfill aside, other job scheduling approaches and considerations include the following:

- Gang Scheduling. In gang scheduling, all processors in the complex swap in new jobs at a set time quantum. Multiple jobs may reside in local memory for each node and can run in their turn. Although difficult to implement and not often used [Feitelson], gang scheduling has some of the same benefit as backfill, allowing short jobs to finish quickly and avoid excessive blocking by long jobs of higher priority.

- Starvation. When small, short jobs can jump ahead of large, long jobs in the queue due to backfill or other strategies, there can be a danger that the long job never runs – this is called starvation. Relaxed backfill avoids this problem by limiting the delay a high priority job must tolerate, and there are other strategies for avoiding starvation, as well.

- Priority Policies. First Come First Served (FCFS), Shortest Job First (SJF), and other priority disciplines are possible. When job length (e.g., SJF) is a factor, users must provide run-time estimates at job submission. User estimates are typically higher than actual run times. FCFS results in low utilization and backfill is often used with FCFS to remedy this.

- Non-Batch Workloads. Interactive sessions, the operating system and other overhead workloads run on separate nodes apart from those under control of PBS for batch. Less than 5% of the nodes are needed for these workloads at the NCCS.

- Conservative and Aggressive Backfill. [Srinivasan] states, "In conservative backfill, every job is given a reservation when it enters the systems. A smaller job is moved forward in the queue as long as it does not delay any previously queued job. In aggressive backfilling, only the job at the head of the queue has a reservation."

There is an extensive literature on parallel job scheduling alternatives.

3. QUEUE SIMULATION TOOL

NCCS uses a simulation tool developed at the center to model the behavior of queues managing workloads on its highly parallel systems. Written in C, the tool accepts input in three ways: (1) interactively, (2) using delimited flat files (to define computer environments and processing workloads), or (3) using actual system log files (for the workload definition only).

CPU Name	CPU Count
e1	512
e2	256
e3	256
palm	128

Queue Name	Urgent
allpurp	50
specx	70
checkout	50
prepost	50
test_ix	50
back	40
allpurp_hi	60
urgent	70
allpurp_small	50
mero	60
train	40
archive	50

Exhibit 9 – Sample Computer File

The computer file names processors and their CPU counts and queues and their priorities, as illustrated in Exhibit 9. The workload file is

formatted for compatibility with system log files and includes columns for the actual wait and run times experienced. Simulated wait and run times can be compared to actual results in the log file. Exhibit 10 is an excerpt from a workload log file. (All times are in seconds. "Submission Time" is Unix epochal seconds.)

191880.palm	ebp_ops	palm	2	1164931160	1	55	10800	prepost
191882.palm	ebp_ops	palm	4	1164931344	1	38	28800	prepost
191870.palm	barry	palm	8	1164929986	402	1023	10800	prepost
191883.palm	ebp_ops	palm	2	1164931381	1	50	3600	prepost
191837.palm	lossone	palm	8	1164927466	1	4547	5400	allpurp_small
191886.palm	ebp_ops	palm	4	1164931674	1	365	10800	prepost
191783.palm	rlang	e1	64	1164920408	4751	6882	7200	allpurp
191891.palm	ebp_ops	palm	2	1164932038	2	68	18000	prepost
191867.palm	c0xxg	palm	2	1164929806	2	2325	28800	allpurp_small
191887.palm	ebp_ops	palm	4	1164931950	1	343	7200	prepost
191893.palm	xoan	palm	16	1164932221	73	6	28800	prepost

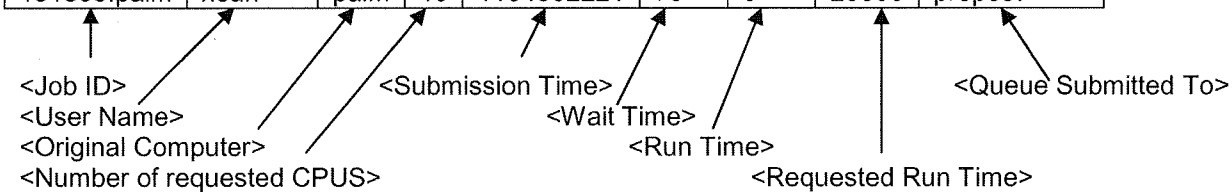


Exhibit 10 – Workload Log File Excerpt

The simulation engine reads in the processor, queue, and workload specifications and builds up workload queue loads and job sequences from these inputs. The QueueRunning simulation module advances through simulated wall-clock time, increments job status, selects new running jobs from the queues when others complete, and backfills appropriately when that option is enabled. Each simulation run takes only a few seconds, so testing many alternative scenarios is easy.

The NCCS validated the tool using both synthetic data with known expected results and natural NCCS workload log data from the Palm/Explore system. The log files provide the submission time, run time and other specifics of production workloads and the simulator uses this production workload trace to model workload behavior under actual and prospective system configurations. NCCS compared input log file wait times to simulated wait times to validate the tool.

4. RESULTS

Exhibit 11 provides the results from several simulation runs. As with the input files, all times are in seconds.

4.1 CORRELATION WITH PREVIOUS WORK

NCCS experience and use of the simulation tool correlates with work published previously.

- Backfilling helps system utilization and throughput by fitting small jobs into the holes left by larger jobs [Ward]. NCCS experience and simulation results support this finding.
- FCFS is the most common job sequencing algorithm used with backfill. FCFS without backfill results in low system utilization [Srinivasan]. NCCS results support this finding.
- Most cluster and other supercomputer facilities use both backfilling and job prioritization [Feitelson]. NCCS uses backfill with job prioritization (see Exhibit 3). Although prioritization can help utilization and wait times if short jobs have higher priorities, prioritization with backfill provides even better results.
- Users generally overestimate job run times significantly [Srinivasan]. Several studies suggest that this inaccuracy helps performance with FCFS and conservative

backfill, because early job completions open holes in the system that backfill can exploit. Inaccuracy is less helpful with relaxed backfill, as large holes are less helpful when backfill is relaxed. NCCS experience supports the finding that users overestimate their run times.

- Relaxed backfill hurts larger jobs that are bumped backward in the queue to accommodate smaller jobs, but averaged across all jobs (small jobs are far more numerous), relaxed backfill reduces the expansion factor dramatically. NCCS historical and simulation data both show that expansion factors increase for jobs with the highest parallelism (not necessarily the longest running jobs). NCCS management and users are willing to accept this tradeoff in the interest of faster response to smaller jobs and higher system utilization.

- [Srinivasan] Although methods like backfill are widely popular on supercomputers, the effectiveness of alternative scheduling strategies depends on the job mix. Understanding one's own workload is important. At the NCCS, management may authorize high priority treatment for a particular workload, and in some cases could dedicate an entire partition or system to a single workload. Storm tracking during hurricane season is an example. Such decisions significantly impact lower priority workloads.

- Several studies used system logs borrowed from external sources or synthetic data. We used our own system logs.

Workload	No Backfill	Strict Backfill	Relaxed Backfill	Remove 128 CPUs	Remove 64 CPUs	Add 128 CPUs
Total elapsed time	1,301,061	1,301,061	1,301,061	1,382,013	1,313,067	1,301,061
Total run time	41,779,774	41,779,774	41,779,774	41,779,774	41,779,774	41,779,774
Average run time	2,785	2,785	2,785	2,785	2,785	2,785
Max run time	43,337	43,337	43,337	43,337	43,337	43,337
Min run time	1	1	1	1	1	1
Average wait time	4,076	190	190	54,934	14,564	1,128
Max wait time	52,340	77,165	77,165	1,266,060	145,741	22,174
Min wait time	0	0	0	0	0	0
Average usage	82.78	86.12	82.78	87.68	86.85	74.50
Total Expansion Factor	2.46	1.10	1.07	20.72	6.23	1.41
(0 to 32) Expansion Factor	2.97	1.02	1.01	26.59	7.96	1.54
(33 to 64) Expansion Factor	1.23	1.17	1.11	6.64	2.02	1.06
(65 to 120) Expansion Factor	1.36	1.81	1.87	6.16	2.15	1.15
(121+) Expansion Factor	1.23	3.01	1.44	3.59	1.65	1.15

EXHIBIT 11 – SOME SIMULATION RESULTS

4.2 RESULTS SPECIFIC TO NCCS

Experimentation with this simulation tool yielded the following results specific to our environment.

- Relatively modest (5-10%) reductions in system capacity caused dramatic increases in wait times. Run times are not affected much because jobs run on dedicated processors once dispatched. Palm/Explore is now running very near saturation. (One can also deduce this from the consistent processor utilization over 90%.) A simulated 5% capacity reduction with no change in the

workload results in wait times of almost one month for some jobs, and an average wait time of 15 hours.

- Adding backfill to the scheduling logic improves system performance more than a 10% increase in processor capacity. It's a bargain.

5. SUMMARY AND FUTURE PLANS

NCCS performance management includes system design and architecture, management of service contracts to incentivize performance, competitive system acquisitions with benchmark tests, training and consulting with users on parallel application programming and parallel APIs (e.g., MPI and OpenMP), use of optimizing compilers and libraries, and hardware system tuning [Glassbrook]. NCCS analysis and optimization of queue structures and job scheduling algorithms is assisted using the queue simulation tool described in this paper. For example, a recent user inquiry about the expected impact of a proposed configuration change was addressed using this tool.

NCCS user appetites for system performance are constantly increasing. Climate researchers and others can do better science when machines with better performance permit more detailed models (e.g., smaller cell sizes, longer simulated time periods, more variations in model ensembles, additional system dynamics details). This tool is available to system administrators to help squeeze more performance out of existing platforms and plan for future upgrades.

Planned future enhancements to the tool include the following:

- Analysis of users' historical workload patterns may yield information applicable to the scheduler's decisions. For example, if a user estimates a 12-hour run time but their jobs complete in 10 minutes on average, then the simulation tool could detect that pattern and forward it to the PBS scheduler with a recommended adjustment to the user's estimation.

- Running the tool in parallel with the PBS scheduler in real time may yield other helpful information about scheduling decisions that would enhance system performance.

- The tool may be made available as open source, so that a community of users can enhance the simulator further.

BIBLIOGRAPHY

[Feitelson] Feitelson, Dror G., and others. "Parallel Job Scheduling – A Status Report." 10th Workshop on Job Scheduling Strategies for Parallel Processing, New York, 2004.

[Fractenberg] Frachtenberg, Eitan, and others. "Parallel Job Scheduling Under Dynamic Workloads." JSSPP 2003, Springer-Verlag, Berlin, 2003.

[Glassbrook] Glassbrook, Richard and McGalliard, James. "Performance Management at an Earth Science Supercomputer Center." CMG 2003.

[Hovestadt] "Scheduling in HPC Resource Management Systems: Queuing vs. Planning." JSSPP 2003, Springer-Verlag, Berlin, 2003.

[Shmueli] Shmueli, Edi and Feitelson, Dror G. "Backfilling With Lookahead to Optimize the Performance of Parallel Job Scheduling." JSSPP 2003, Springer-Verlag, Berlin, 2003.

[Srinivasan] Srinivasan, Srividya and others. "Characterization of Backfilling Strategies for Parallel Job Scheduling." Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02), 2002.

[Ward] Ward, William and others. "Scheduling Jobs on Parallel Systems Using a Relaxed Backfill Strategy". Lecture Notes in Computer Science, Springer-Verlag, 2002.