April 19, 2000. For submission to The 6th Workshop on Job Scheduling Strategies for Parallel Processing

System Utilization Benchmark on the Cray T3E and IBM SP

Adrian Wong, Leonid Oliker, William Kramer, Teresa Kaltz and David Bailey

National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
One Cyclotron Rd
Berkeley, CA 94720, USA

Obtaining maximum utilization of parallel systems continues to be an active area of research and development. This article outlines a new benchmark, called the *Effective System Performance* (ESP) test, designed to provide a utilization metric that is transferable between systems and illuminate the effects of various scheduling parameters. Results with discussion are presented for the Cray T3E and IBM SP systems together with insights obtained from simulation.

1 Introduction

The overall value of a high performance computing system depends not only on the raw computational speed but also on system management. System characteristics such as job scheduling efficiency, reboot and recovery times and the level of process management are important factors in the overall usability and effectiveness of the system. Common performance metrics such as the LINPACK and NAS Parallel Benchmarks [1, 2] are useful for measuring sustained computational performance, but give little or no insight into system-level efficiency issues.

In this article, we describe a new benchmark, the *Effective System Performance* (ESP) benchmark, which measures system utilization [3]. Our primary motivation in developing this benchmark is to aid the evaluation of high performance systems. Additionally, it will be used to monitor the impact of configuration changes and software upgrades in existing systems. We also hope that this benchmark will provide a focal point for future research and development activities in the high performance computing community. The emphasis in this work is on scheduling and resource management and should be viewed as complementary to performance benchmarks, such as NAS.

The ESP test extends the idea of a throughput benchmark with additional constraints that encapsulate day-to-day operation. It yields an efficiency measurement based on the ratio of the actual elapsed time relative to the theoretical minimum time assuming perfect efficiency. This ratio is independent of the computational rate and is also relatively independent of the number of processors used, thus permitting comparisons between platforms.

The test was run on the Cray T3E and the IBM SP at NERSC. The T3E consists of 512 Alpha EV56 processors at 450 MHz with an aggregate peak of 614 GFlop/s. The SP consists of 512 Power3 processors at 200 MHz with an aggregate peak of 410 GFlop/s. The SP exhibits higher sustained performance for most applications, however, the T3E has better scalability for tightly-coupled parallel applications. With two systems of the same size but different scheduling characteristics, NERSC is in a unique position to implement and validate the ESP test.

2 Utilization and Scheduling

In this work, we are primarily concerned with parallel applications that require a static number of processors (or partition size). Scheduling several parallel applications concurrently is particularly problematic since time and partition constraints must be satisfied while simultaneously the system utilization should be at a maximum. Utilization, in this sense, is defined as the fraction of busy processors to available processors integrated over time. In day-to-day operation, other constraints and requirements create a situation more complex and subtle. For example, a scheduler may use a *best-fit-first* (BFF) strategy but at a cost of starvation of larger partition jobs. While this may reduce the elapsed time and achieve higher utilization compared to a *first-come-first-serve* (FCFS) strategy it does not address the issues of turnaround, fairness and productivity.

One indirect measurement of utilization is the elapsed time required to process a workload consisting of a number of jobs. If the time for each job is constant, irrespective of whether it is run dedicated or concurrently with other jobs, then the variation in the elapsed time for the workload depends only on the scheduling strategy and system overhead. For a given workload with jobs of varying partition sizes and elapsed times, a utilization efficiency, E, can be defined as,

$$E = \frac{\sum_i p_i t_i}{PT} \tag{1}$$

where p_i and t_i are the partition size and elapsed time, respectively, for the i-th job, T is the observed time for the workload and P is the number of available processors. In the ideal limit of perfect packing with no overhead, the efficiency, E, approaches unity. Here the numerator and denominator are in units of CPU-hours (or CPU-seconds), a unit of that is useful for discussing parallel resources. Note, there is a distinction between perfect packing and optimal scheduling. Some of the possible parameters include; preemption, backfilling, variable deadlines, multiple resource requirements, scheduling dependencies and job priorities. Not surprisingly this problem is NP-hard and many heuristic techniques have been developed to approximate a good solution. Examples of recent work for non-preemptive, preemptive and multiple resource schemes include [4, 5, 6, 7].

Particular scheduling strategies depend on the availability of certain key system functionality. As shown in Table I, the systems examined in this work differ considerably in available functionality. *Checkpoint/restart* is the ability to save to disk and subsequently restore a running job. System-initiated checkpointing of long-running jobs affords flexibility in scheduling and maintenance. Without checkpointing, one is faced with the undesirable choice between waiting to idle the machine or the premature termination of jobs. *Swapping* and *gang-scheduling* is the parallel analog of time-slicing on single processor machines but usually with a course granularity. This allows oversubscription (multiple processes per processor), increasing the apparent number of processors available and allowing greater flexibility in scheduling. *Job migration/compaction* refers to the capability to move a running job to a different set of processors. This aids in defragmentation and scheduling large partitions. *Priority preemption* is the ability to launch a higher priority job with the resulting effect of swapping out lower priority jobs. This is useful to improve turnaround and prevent starvation of large partition jobs. *Backfill* is the insertion of smaller, shorter jobs ahead of earlier jobs in the queue to fill empty slots. More sophisticated backfill algorithms exploit *a priori* run time information such that jobs have a guaranteed launch time but may be promoted ahead to fill empty slots. Systems with *disjoint partitions* do not require contiguous (in a topological sense) parallel partitions on the interconnect and, therefore, fragmentation is not an issue.

Table I: System Functionality on T3E and SP

Function	T3E	SP
Checkpoint/restart	X	
Swapping/Gang-scheduling	X	
Job migration/compaction	X	
Priority preemption	X	
Backfill	X	X
Disjoint partitions		X

The first four functions mentioned, namely, checkpoint/restart, swapping, migration and preemption, all depend on the fundamental kernel operation of preemptively changing process images between run and idle states and moving them to memory and disk. Unfortunately, this is either not implemented in stock operating systems or is not exposed to the global scheduling/queuing software.

The Cray T3E at NERSC has demonstrated high utilization due to system management and the availability of checkpoint/restart and priority preemption [8, 9].Scheduling on the T3E comprises two interacting subsystems; a batch queue (NQS) and the *global resource manager* (GRM). NQS schedules at a coarse grain and launches jobs from queues based on site policies. Multiple NQS queues for different partition sizes are used. The number of released jobs for each queue may be limited by the NQS configuration. Therefore, the profile of jobs launched may be adjusted according to the time of the day. For example, at night, larger partition jobs are preferentially launched by limiting small partition jobs. Dynamic management of currently launched jobs is implemented by GRM using swapping, priority preemption and compaction. Two priority rankings are recognized by GRM; normal and prime. The prime rank is used to preemptively insert jobs to run immediately with the effect of swapping out running jobs. Within each priority rank, there maybe more processors subscribed than physically available, in which case, GRM can either gang-schedule between jobs or hold jobs pending a suitable partition. GRM uses a BFF strategy to launch held jobs. To prevent starvation of larger jobs, NQS must be appropriately configured and prime rankings judiciously assigned to larger jobs.

Loadleveller is used on the SP for scheduling batch jobs. In general, each job is assigned dedicated processors and runs to completion without swapping. Different job *classes* with varying requirements and priorities may be defined within Loadleveller. Loadleveller uses an overall FCFS strategy with backfill to launch jobs from the queue. The ordering of jobs considered for launch maybe adjusted using system and/or user assigned priorities. Efficient backfill requires *a priori* estimated run times.

3 Benchmark Design

A throughput benchmark was the initial starting point for the design of the ESP test. It consists of a set of jobs of varying partition sizes and times with the objective of obtaining the shortest elapsed run time. By reporting the utilization efficiency, E, instead of the absolute time, the ESP test is independent of the computational rate. A theoretical minimum elapsed time of 4 hours (or 4×512 CPU-hours) on the T3E was chosen for the benchmark. This choice was a compromise between a longer simulation more representative of actual production and a shorter time more amenable to benchmarking.

The turnaround of larger partition jobs has always been a concern. In order to encapsulate this problem, the test includes two jobs with partition sizes equal to the number of available processors (*full configuration* jobs). The test stipulates that upon submission, the full configuration jobs must be run before any further jobs are launched. The first full configuration job can only be submitted after 10% of the theoretical minimum time has elapsed such that it is non-trivial to schedule. Similarly, the second full configuration job must complete within 90% of the test and is not simply the last job to be launched. The requirement to run these two full configuration jobs is a difficult test for a scheduler but it is, nonetheless, a common scenario in production environments.

Outages, both scheduled and unscheduled, are common in these systems and the time to shutdown and reboot has a significant impact on utilization over the lifetime of a system. The ESP test includes a shutdown with reboot which is required to start immediately after the completion of the first full configuration job. In practice, the shutdown and reboot cycle is difficult to implement without manual intervention during the test. If the shutdown/reboot time, S, is known in advance then the operation need not be performed. The utilization efficiency with shutdown/reboot is simply,

$$E = \frac{\sum_i p_i t_i}{P(T + S)} \tag{2}$$

The jobs are grouped into three blocks and the order of submission is determined from a reproducible pseudo-random sequence. The total number of CPUs requested in the first block is at least twice the available processors and the number of CPUs in the second block at least equal to the available processors. The remaining jobs constitute the third block. The first block is submitted at the start with the second and third blocks submitted 10 and 20 minutes thereafter, respectively. This structure was designed to forestall artificially configured queues specific to this test and, at the same time, provide sufficient queued work to allow flexibility in scheduling. No manual intervention is allowed once the test is initiated.

We consider it important that the job mix be representative of the user workload. Accounting records for three months from the T3E were distilled into a workload profile. User jobs were sorted into classes defined by run time and partition size. Table II shows the workload profile expressed as a percentage of the available CPU-hours. Using the 4 hour

Table II: Workload Profile on the T3E at NERSC (%)

Size (procs)	8	16	32	64	128	256
Time(s)						
300- 900	0.5	0.7	1.9	4.5	0.5	1.3
1000-3600	0.5	1.4	3.4	10.2	2.8	4.0
3700-9600	2.0	4.2	7.6	35.1	17.3	2.4

elapsed time (2048 CPU-hours) for the test, Table II allows one to calculate the amount of CPU-hours for each class. For example, the class of jobs with a partition size of 64 and taking 1000-3600 seconds, represents 209 CPU-hours (= 209/64 hours). It is important to note that while the profile reflects the application and problem-size constraints of users (for example, a minimum of aggregate memory), there is also a secondary effect of the scheduling policy enforced on the system. That is, the users will naturally adapt their pattern of usage to maximize their productivity for a given queue structure.

The applications in the job mix originate from our user community and are used in production computing. Furthermore, the job mix profile was designed to span the diverse scientific areas of research amongst our users. Attention was also paid to diversify computational characteristics such as the amount of disk I/O and memory usage. For each class, an application and problem set was selected to satisfy the time and partition size constraints. The number of instances (Count) of each application/problem was adjusted such that aggregate CPU-hours reflected the workload profile. Table III lists the final job mix for the ESP benchmark with the elapsed times for each job on the T3E and SP.

4 Results and Discussion

Two test runs were completed on the T3E. In both cases, a separate queue was created for full configuration jobs and was marked to preempt running jobs. The full configuration jobs can thus be launched immediately on submission independent of the queue of general jobs. Process migration/compaction was also enabled for both runs. In the first run, labeled *Swap*, the system was oversubscribed by two and gang-scheduled with a time-slice of 20 minutes. A single NQS queue was used for the general job mix. In the second run, labeled *NoSwap*, the system was not oversubscribed. Each job ran uninterrupted until completion. Six queues for different maximum partition sizes; 256, 128, 64, 32, 16, with decreasing priority were used.

On the SP, two classes (queues) were created in Loadleveller; a general class for all jobs and a special high priority class for the full configuration jobs. It is not possible to

Table III: ESP Application Job Mix

Application	Discipline	Size	Count	Time(s)	
				T3E	SP
gfft	Large FFT	512	2	30.5	255.6
md	Biology	8	4	1208.0	1144.9
md		24	3	602.7	583.3
nqclarge	Chemistry	8	2	8788.0	5274.9
nqclarge		16	5	5879.6	2870.8
paratec	Material Science	256	1	746.9	1371.0
qcdsmall	Nuclear Physics	128	1	1155.0	503.3
qcdsmall		256	1	591.0	342.4
scf	Chemistry	32	7	3461.1	1136.2
scf		64	10	1751.9	646.4
scfdirect	Chemistry	64	7	5768.9	1811.7
scfdirect		81	2	4578.0	1589.1
superlu	Linear Algebra	8	15	288.3	361.2
tlbebig	Fusion	16	2	2684.5	2058.8
tlbebig		32	6	1358.3	1027.0
tlbebig		49	5	912.9	729.4
tlbebig		64	8	685.8	568.7
tlbebig		128	1	350.0	350.7

selectively backfill with Loadleveller. Our preliminary runs showed that backfill would defer launching of the full configuration job until the end of the test. This would clearly violate the intent of the test. Backfill was implicitly disabled by assigning large wallclock times (several times greater than the complete test) to all jobs. Thus Loadleveller was reduced to a strictly FCFS strategy.

The results of the ESP test for the T3E and the SP are summarized in Table IV. Two efficiency measurements, with and without the shutdown/reboot time factored in, are reported. Figures I, II and III show the details of the three runs where the instantaneous utilization is plotted against time and the time axis has been rescaled by the theoretical minimum time. Additionally, the start time for each job is indicated by an impulse where the height equals the partition size.

The obvious point is the significantly higher utilization efficiency of the T3E compared to the SP. This is due to the lack of a suitable mechanism to immediately launch full configuration jobs on the SP. On submission of the full configuration jobs, a considerable amount of time was spent waiting for running jobs to complete. This is evident in Figure III which shows two large regions where the instantaneous utilization drops very low. Without this drawback, it is likely the utilization on the SP would be comparable to the T3E. The time lag to run preferential jobs is indicative of the difficulty in changing modes of operation on the SP. This is important for sites that routinely change system characteristics, for example, between interactive and batch or between small and large partitions. The most desirable remedy would be to either checkpoint or dynamically swap out running jobs. It is noteworthy that the SP completed the test in less time due to its higher computational rate.

As seen in Figure I, the BFF mechanism on the T3E deferred large partition jobs (≥ 128) until the end. Consequently, at the end of the test there were large gaps that could not be filled by small jobs. On the SP, a FCFS strategy was indirectly enforced which can be seen illustrated in Figure III where the distribution of job start times is unrelated to partition size. It is evident from Figures I and II that a significant loss of efficiency on the T3E is incurred at the tail end of the test. In an operational setting, however, there are usually more jobs to launch. That is, the fact the ESP test is finite poses a problem since we are interested in a continual utilization given a hypothetical infinite number of queued jobs. Suggested solutions to this dilemma have proven to be awkward and require manual intervention. How the test should be terminated and post-analyzed will be reexamined in the next design of the ESP test.

The distribution of start times is qualitatively similar between the Swap and NoSwap runs on the T3E although the queue set up was different. In the second run, we deliberately assigned increasingly higher priorities to larger partition queues in an attempt to mitigate starvation. However, shortly after the start, it is unlikely that a large pool of idle processors would become coincidently available. In this scenario, the pattern of job submission reverts back to BFF and the queue set up has little impact. On the other hand, there is considerable difference in efficiency between the two T3E runs. This is attributed to the overhead of

Table IV: ESP Results

	T3E		SP
	Swap	*NoSwap*	
Available processors	512	512	512
Jobmix work (CPU-seconds)	7437860	7437860	3715861
Elapsed Time (seconds)	20736	17327	14999
Shutdown/reboot (seconds)	2100	2100	5400
Efficiency	0.64	0.75	0.36
Efficiency (w/o reboot)	0.70	0.84	0.48

Table V: Observed and Simulation Results on the T3E

	Efficiency
Observed (NoSwap)	0.84
Simulation	
+ w/o preemption	0.49
+ with preemption	0.84
+ gang-scheduling	0.86

swapping which is significant when the oversubscribed processes cannot simultaneously fit in memory and process images must be written to disk.

To aid the evaluation of the performance and sensitivity of the ESP test, a simulator was developed to predict the runtime of this workload using various scheduling schemes. Several simple scheduling algorithms, such as FCFS and BFF were implemented together with the option of using backfill schemes and priority preemption. The simulator was used in the definition of the ESP test in order to ensure that the test did not exhibit pathological behavior. We have also used the simulator to estimate results of the ESP test on various system configurations and compared them to observed results.

Table V compares observed efficiency against various simulator efficiencies without shutdown/reboot. The simulated efficiencies are somewhat optimistic since they do not account for system overhead such as I/O contention, swapping and processor fragmentation. A gang-scheduled simulation of the ESP test with a 1000 second time-slice and oversubscription of two achieved a utilization efficiency of 0.86. However, this is overly optimistic as the observed efficiency of 0.70 from the Swap run indicate that there is a significant overhead incurred with gang-scheduling. Furthermore, this is only a slight increase compared to the preemption simulation with an efficiency of 0.84 without the use of gang-scheduling. The most noticeable result obtained from the simulations has been the assessment of the preemption functionality. The simulated efficiency of the ESP test using a BFF algorithm increased from 0.49 to 0.84 when preemption was employed. These results indicate that preemption can have a very significant impact on system utilization in a real operational setting which agrees with the conclusion from the SP run.

5 Conclusion

In this work we described a new utilization benchmark that we successfully ran on two parallel computer systems. It has provided quantitative data on utilization and scheduling efficacy of these resources and useful insights on how to manage these systems. The most important conclusion is that certain system functionality; including checkpoint/restart, swapping and migration, are critical for efficient scheduling strategies. Unfortunately, the scheduling of parallel systems has received little attention from operating system kernel developers wherein such functionality must originate.

Future work will also include testing other parallel platforms with different scheduling and system functionality. We are interested in quantifying the effects of different schedulers with Loadleveller and particularly interested in utilizing the backfill capability. We also plan to introduce variations on the ESP test, for example, providing *a priori* runtimes to schedulers that may exploit such information. The scalability of the test at larger number of processors and the difficulty with the tail end of the test will have to addressed. Finally we intend to conduct a theoretical analysis of the ESP test in the context of multiple resource requirements. At some point, we hope to make the test freely available. We hope that this test will be of use to other sites and spur both industry and research to improve system utilization.

6 Acknowledgments

This work was supported by the Office of Computational and Technology Research, Division of Mathematical, Information and Computational Sciences of the U.S. Department of Energy, under contract DE-AC03-76SF00098. The authors would like to acknowledge the assistance of the system administrators on the T3E and SP; Tina Butler, Nicholas Cardo and Jacqueline Scoggins, in setting up the ESP test. We would like to thank the developers of the applications for allowing their software to be included in this test, including; Environmental Molecular Science Laboratory (NWChem), George Vahala (tlbe), Gregory Kilcup (qcd), Xiaoye Li (SuperLU) and Andrew Canning (paratec).

References

[1] Jack Dongarra. Performance of various computers using standard linear algebra software in a fortran environment. Available at *http://www.netlib.org/benchmarks/performance.ps*.

[2] David H. Bailey. The NAS parallel benchmarks. *International Journal of Supercomputer Applications*, 5:66, 1991.

[3] Adrian Wong, Leonid Oliker, William Kramer, Teresa Kaltz, and David Bailey. Evaluating system effectiveness in high performance computing systems. Available at *http://www.nersc.gov/ dhbailey*.

[4] D. Talby and D. G. Feitelson. Supporting priorities and improving utilization of the IBM SP2 scheduler using slack-based backfilling. In *13th International Parallel Processing Symposium*, page 513. IEEE, 1999.

[5] Dmitry Zotkin and Peter J. Keleher. Job-length estimation and performance in backfilling schedulers. In *8th High Performance Distributed Computing Conference*. IEEE, 1999.

[6] D. G. Feitelson and M. A. Jette. Improved utilization and responsiveness with gang scheduling. In D. G. Feitelson and L. Rudolph, editors, *Workshop in Job Scheduling Strategies for Parallel Processing*, page 238. IEEE, Springer-Verlag, 1997.

[7] W. Leinberg, G. Karypis, and V. Kumar. Job scheduling in the presence of mulitple resource requirements. TR 99-025, 1999. Computer Science Department, University of Minnesota.

[8] Horst Simon, William Kramer, and Robert Lucas. Building the teraflops/petabytes production supercomputing center. In *Fifth International Euro-Par Conference Proceeding*, page 61, August 1999.

[9] Jay Blakeborough and Mike Welcome. T3E scheduling update. In *41st Cray User Group Conference Proceedings*, May 1999.

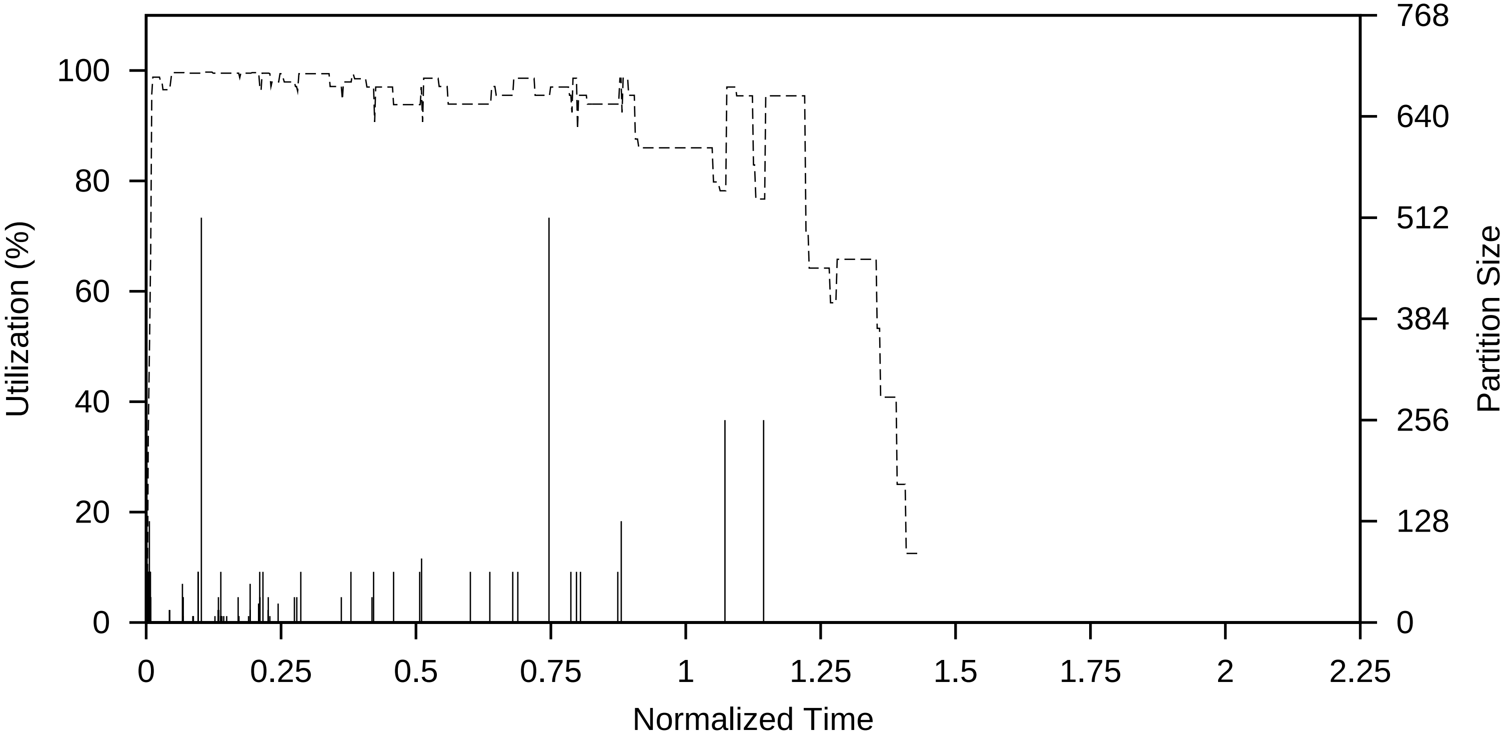

Figure I: T3E Swap : Utilization Chronology

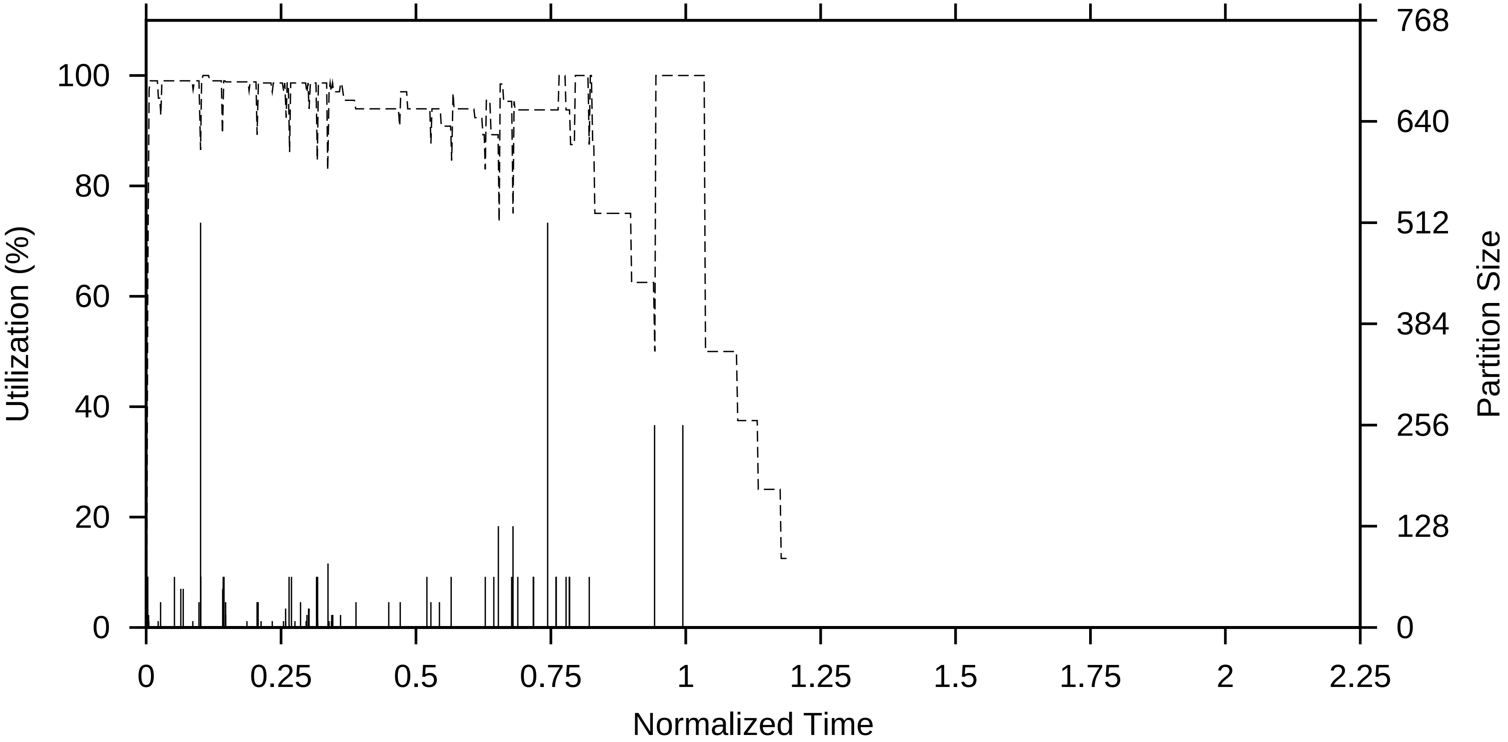

Figure 11: T3E NoSwap : Utilization Chronology

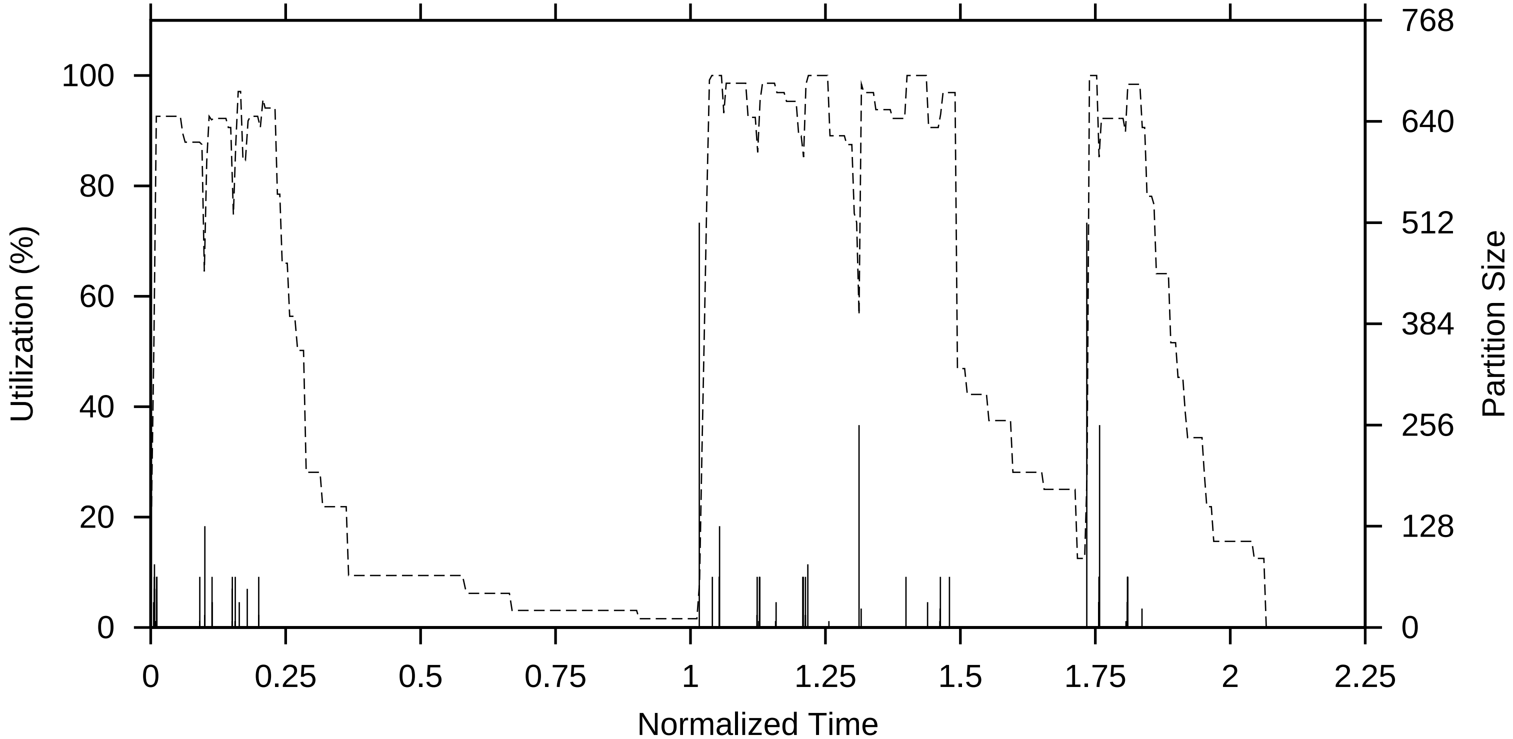

Figure III: SP : Utilization Chronology