ESP: A System Utilization Benchmark

Adrian T. Wong, Leonid Oliker, William T. C. Kramer, Teresa L. Kaltz and David H. Bailey
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
One Cyclotron Road, Berkeley, CA 94720, USA
Correspondence: David H. Bailey, `dhbailey@lbl.gov`

Abstract

This article describes a new benchmark, called the Effective System Performance (ESP) test, which is designed to measure system-level performance, including such factors as job scheduling efficiency, handling of large jobs and shutdown-reboot times. In particular, this test can be used to study the effects of various scheduling policies and parameters. We present here some results that we have obtained so far on the Cray T3E and IBM SP systems, together with insights obtained from simulations.

1. Introduction

The overall performance value of a high performance computing system depends not only on its raw computational speed but also on system management effectiveness, including job scheduling efficiency, reboot and recovery times and the level of process management. Common performance metrics such as the LINPACK and NAS Parallel Benchmarks [3, 1] are useful for measuring sustained computational performance for individual jobs, 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 and effectiveness [9]. Our primary motivation in developing this benchmark is to aid the evaluation of high performance systems. We plan to use it to monitor the impact of configuration changes and software upgrades in existing systems. But we also hope that this benchmark will provide a focal point for future research and development activities in the high performance computing community, possibly leading to significantly improved system-level efficiency in future production systems.

The ESP test extends the idea of a throughput benchmark with additional features that mimic day-to-day supercomputer center 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.

2. Scheduling and Utilization

Scheduling a set of parallel applications of different partition sizes is quite a challenge, since time and partition constraints must be satisfied, while at the same time maximizing the system utilization. In this setting, utilization is the fraction of busy processors to available processors integrated over time. In day-to-day operation, other constraints and requirements create a more complex situation. For example, a scheduler may use a best-fit-first (BFF) strategy, which is satisfactory for a workload of small- and medium-sized jobs, but which often starves large-partition jobs.

0-7803-9802-5/2000/$10.00

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

Table 1: System Functionality on T3E and SP

One measure of system utilization is the elapsed time required to process a given workload suite. If the time for each job is constant, regardless of whether it is run alone 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

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

where p_i and t_i are the partition size and best-case 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 is unity. Here the numerator and denominator are in units of CPU-seconds. The problem of optimal packing is NP-complete, but several effective heuristic techniques have been developed. Examples of recent work in non-preemptive, preemptive and multiple resource schemes include [8, 10, 4, 6].

Particular scheduling strategies depend on the availability of certain key system functionalities. As shown in Table 1, the currently available T3E and SP systems at NERSC differ considerably in this regard. Here *checkpoint/restart* is the ability to save to disk and subsequently restore a running job. Without checkpointing, a system administrator is often faced with the unpleasant choice of allowing a large portion of the system to be idle, or prematurely terminating one or more running jobs. *Swapping* and *gang-scheduling* are the parallel analogues of time-slicing on single processor machines, but with a relatively coarse granularity. They 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 pre-emption* is the ability to launch a higher priority job with the resulting effect of swapping out, suspending or terminating 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.

Checkpoint/restart, swapping, migration and pre-emption all depend on the fundamental kernel operation of pre-emptively changing process images between run and idle states and moving them to memory and disk. Unfortunately, few vendors have implemented this feature, or at least they have not made it available to the global scheduling/queuing software.

The Cray T3E at NERSC has recently demonstrated over 95% utilization for an extended time period [7, 2]. Scheduling on the T3E involves 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.

Loadleveler software is used on the IBM 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 first-come-first-served (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. ESP Benchmark Design

The throughput workload that is currently used in the ESP test 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. The ESP test runs roughly four hours on 512 CPUs of the NERSC T3E. This time length was a compromise between a longer simulation that is more representative of actual usage and a shorter time that is more suitable to routine benchmarking.

The throughput of large-partition jobs is an ongoing concern at a large supercomputer center such as NERSC, since without this focus, the rationale for acquiring and operating a large tightly-coupled computer system is weakened. Thus the ESP test includes two “full configuration jobs”, with partition sizes equal to the number of available processors. The run rules for the ESP test specify 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 capability environments.

Large systems typically require a great deal of system adminstration to maintain and improve their operation. These activities often require a system outage, either scheduled or unscheduled, and the time required for shutting down and restarting the system can significantly impact the overall system utilization. For this reason, the ESP test includes a shutdown-reboot cycle, which is required to start immediately after the completion of the first full configuration job. In practice, a shutdown-reboot cycle is difficult to implement as part of a test run without manual intervention during the test. So if the shutdown/reboot time S is known in advance then the operation need not be performed, and the utilization efficiency can be computed as

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

It has been pointed out to us that this design of a shutdown-reboot immediately following the full-configuration job tends to ignore the need to restart some applications (or the advantages of checkpointing). However, our test otherwise favors systems with these features.

Size (CPUs)	2–15	16–31	32–63	64–127	128–255	256+
0–3600 sec.	0.7	1.1	3.7	6.6	0.9	4.2
3600–7200 sec.	0.3	1.0	1.6	8.1	2.4	1.1
7200–14400 sec.	0.8	3.7	5.1	22.5	8.5	1.0
14400+ sec.	1.2	0.5	2.5	12.6	8.8	1.4

Table 2: Workload Profile on the NERSC T3E (Percent)

Size (nodes)	0–31	32–63	64–127	128–256
0–3600 sec.	6.9	9.0	0.5	0.3
3600–9600 sec.	2.2	3.0	0.8	0.8
9600–10800 sec.	3.0	3.1	2.7	1.7
10800–14400 sec.	16.8	22.8	21.9	4.5

Table 3: Workload Profile on the NERSC SP (Percent)

The jobs in the ESP suite 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 permitted once the test has been initiated.

We consider it important that the ESP test be representative of the user workload, so we designed the distribution of job sizes and run times in the ESP suite to roughly match the distribution of jobs running on NERSC production systems (except that the ESP run times were scaled down so that the test could be run in a more reasonable elapsed time). Tables 2 and 3 show some of this data, each set taken from a recent month's accounting records.

The applications in the ESP 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 4 lists the final job mix for the ESP benchmark with the elapsed times for each job on the T3E and SP.

If desired, the acutal codes used in the ESP test can be varied in several ways. First, a different site workload could be used, so the ESP test would compare different systems on this job mix. Secondly, a general suite of codes (such as well-known benchmark codes) could be used, with the objective of being easily portable to all systems. Thirdly, the test could be run with job mixes derived from different workloads and/or disciplines, permitting comparison of systems across disciplines. In part, the job mix described here achieves this last objective, since it is derived from a rather diverse set of applications and

				Run Times	
Application	Discipline	Size	Count	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

Table 4: ESP Application Job Mix

numerous, but there is no reason that codes representing other applications and/or discplines could not be used as well.

4. Results

Two test runs were completed on the T3E. In both cases, a separate queue was created for full configuration 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 *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 5. Two efficiency measurements, with and without the shutdown/reboot time factored in, are reported. Figures 1, 2 and 3 show the details of the three runs where the instantaneous utilization is plotted against time and the

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

Table 5: ESP Results

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.

These results show that the T3E has a significantly higher utilization efficiency than the SP. This is mainly due to the lack of an effective 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 3 which shows two large regions where the instantaneous utilization drops to a very low value. Without this drawback, it is likely that the utilization on the SP would be roughly 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 best remedy would be to either checkpoint or dynamically swap out running jobs.

Nonetheless, it is noteworthy that the SP system, in spite of its lower system-level efficiency, completed the test in less time due to its higher computational rate.

As seen in Figure 1, the BFF mechanism on the T3E deferred large partition jobs ($\geq$ 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 3 where the distribution of job start times is unrelated to partition size. It is evident from Figures 1 and 2 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 re-examined 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 swapping which is significant when the oversubscribed processes cannot simultaneously fit in memory and process images must be written to disk.

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

Table 6: Observed and Simulation Results on the T3E

5. Simulations

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 6 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.

6. Conclusions and Future Plans

We have described a new system utilization benchmark, which we have successfully run on two highly parallel production supercomputers. This test has provided quantitative data on the utilization and scheduling efficiency of these systems, as well as useful insights on how to better manage them. The most important conclusion is that certain system functionalities, such as checkpoint/restart, swapping and migration, are critical for the highly efficient operation of large systems.

We have summarized here the results that we have obtained so far. We have however reached agreement with some other research centers and vendors to run this test on their large systems, including systems with different architectures, scheduling systems and operating priorities. In particular, we have established plans to run the ESP test on a clustered SGI Origin system at NCSA, which has a total of 512 CPUs, and on a more tightly coupled SGI Origin system in the NAS facility at NASA Ames Research Center, which has also has a total of 512 CPUs. In addition, we have now reached an agreement with Compaq to run the ESP test on a system with 512 Alpha EV6/7 CPUs and a Quadrix interconnect. These results will be presented at the SC2000 meeting in November.

We also plan to introduce some variations on the ESP test, for example a variation with a job mix that provides *a priori* runtimes to schedulers, so that they may exploit this information in achieving higher utilization. The scalability of the test at larger number of processors and the difficulty with the tail end of the test also needs to be addressed. Further, we intend to conduct a theoretical analysis of the ESP test in the context of multiple resource requirements. Finally, we plan to make package the test in a freely available software archive, together with facilities for simple installation and execution. In this way we hope that this test will be of use to other sites and will help spur both industry and research to improve system utilization on future systems.

7. 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). We also wish to thank the referees of the earlier draft for some very useful comments on the manuscript.

References

[1] David H. Bailey, et. al, "The NAS Parallel Benchmarks", *Intl. Journal of Supercomputer Applications*, vol. 5 (1991), pg. 66.

[2] Jay Blakeborough and Mike Welcome, "T3E Scheduling Update", *41st Cray User Group Conference Proc.*, 1999.

[3] Jack Dongarra, "Performance of Various Computers Using Standard Linear Algebra Software in a Fortran Environment". Available from `http://www.netlib.org/benchmarks/performance.ps`.

[4] D. G. Feitelson and M. A. Jette, "Improved Utilization and Responsiveness with Gang Scheduling", *Workshop in Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph, ed., Springer-Verlag/IEEE, 1997, pg. 238.

[5] Morris A. Jette, "Performance Characteristics of Gang Scheduling in Muliprogrammed Environments", *Proc. of Supercomputing '97*, IEEE, 1997.

[6] W. Leinberg, G. Karypis and V. Kumar, "Job Scheduling in the presence of Mulitple Resource Requirements", CS Dept. TR 99-925, University of Minnesota, 1999.

[7] Horst Simon and William Kramer and Robert Lucas, "Building the Teraflops/Petabytes Production Supercomputing Center", *Fifth International Euro-Par Conference Proc.*, 1999, pg. 61.

[8] D. Talby and D. G. Feitelson, "Supporting Priorities and Improving Utilization of the IBM SP2 Scheduler using Slack-Based Backfilling", *13th International Parallel Processing Symposium*, IEEE, 1999, pg. 513.

[9] Adrian Wong, Leonid Oliker, William Kramer, Teresa Kaltz and David Bailey, "Evaluating System Effectiveness in High Performance Computing Systems", 1999. Available from `http://www.nersc.gov/~dhbailey`.

[10] Dmitry Zotkin and Peter J. Keleher, "Job-Length Estimation and Performance in Backfilling Schedulers", *8th High Performance Distributed Computing Conference*, IEEE, 1999.

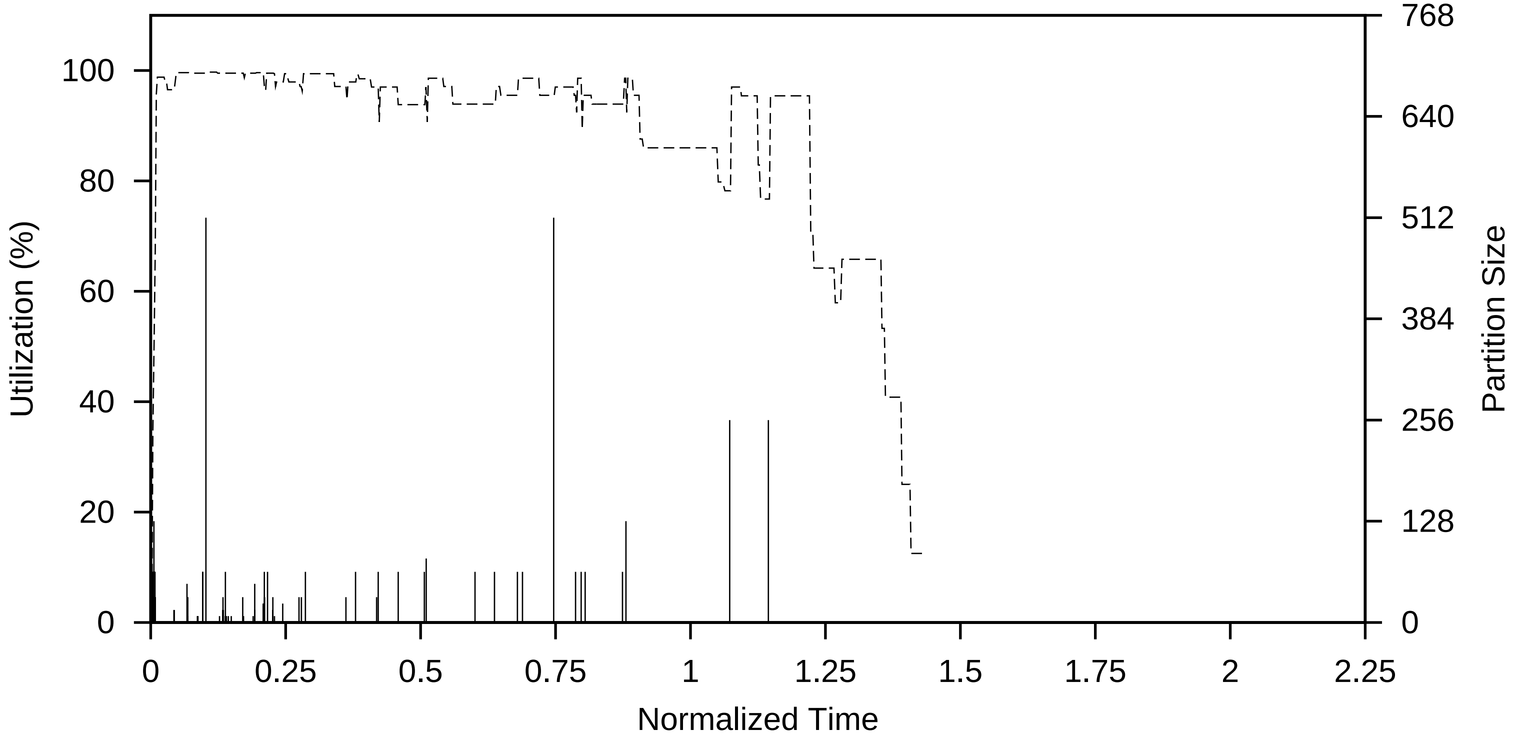

Figure 1: T3E Swap : Utilization Chronology

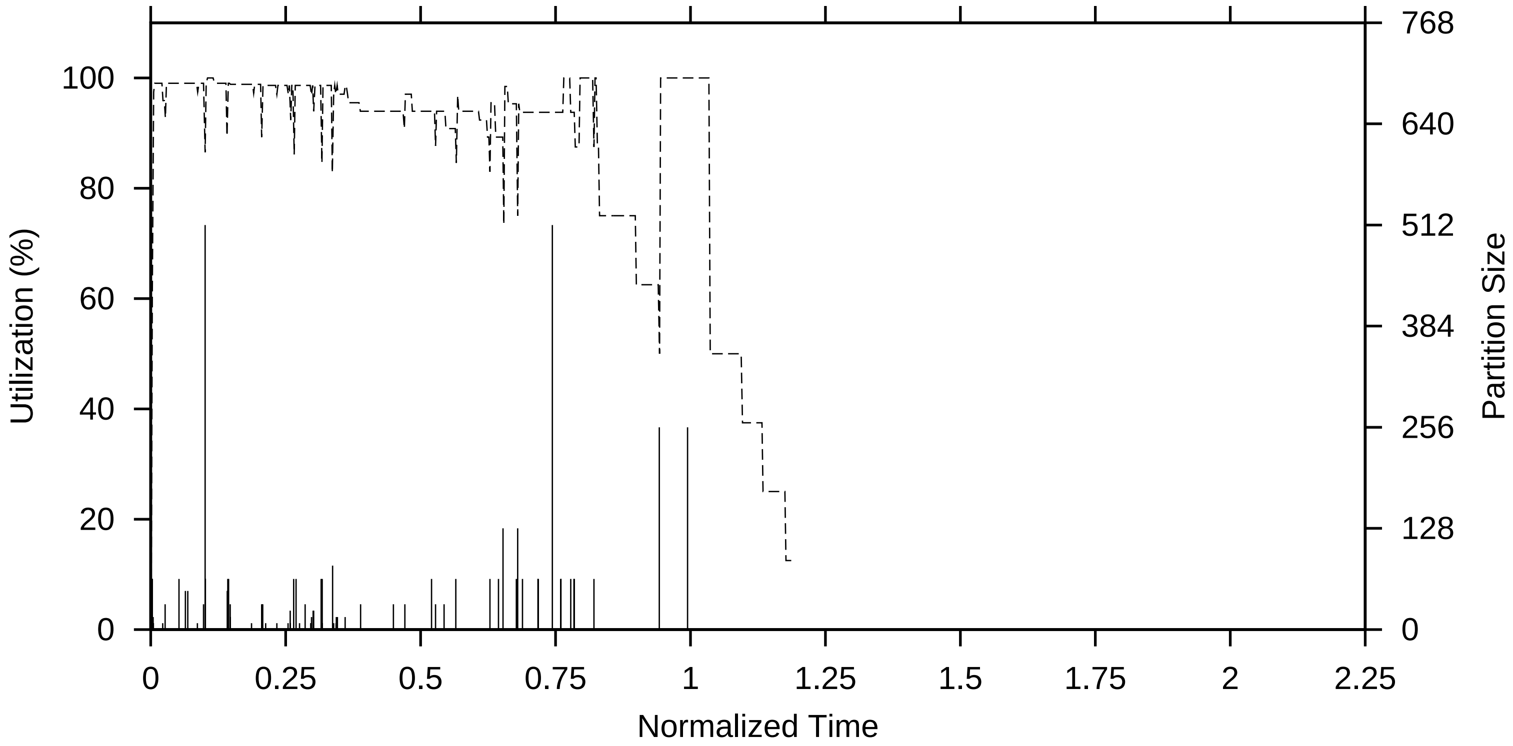

Figure 2: T3E NoSwap : Utilization Chronology

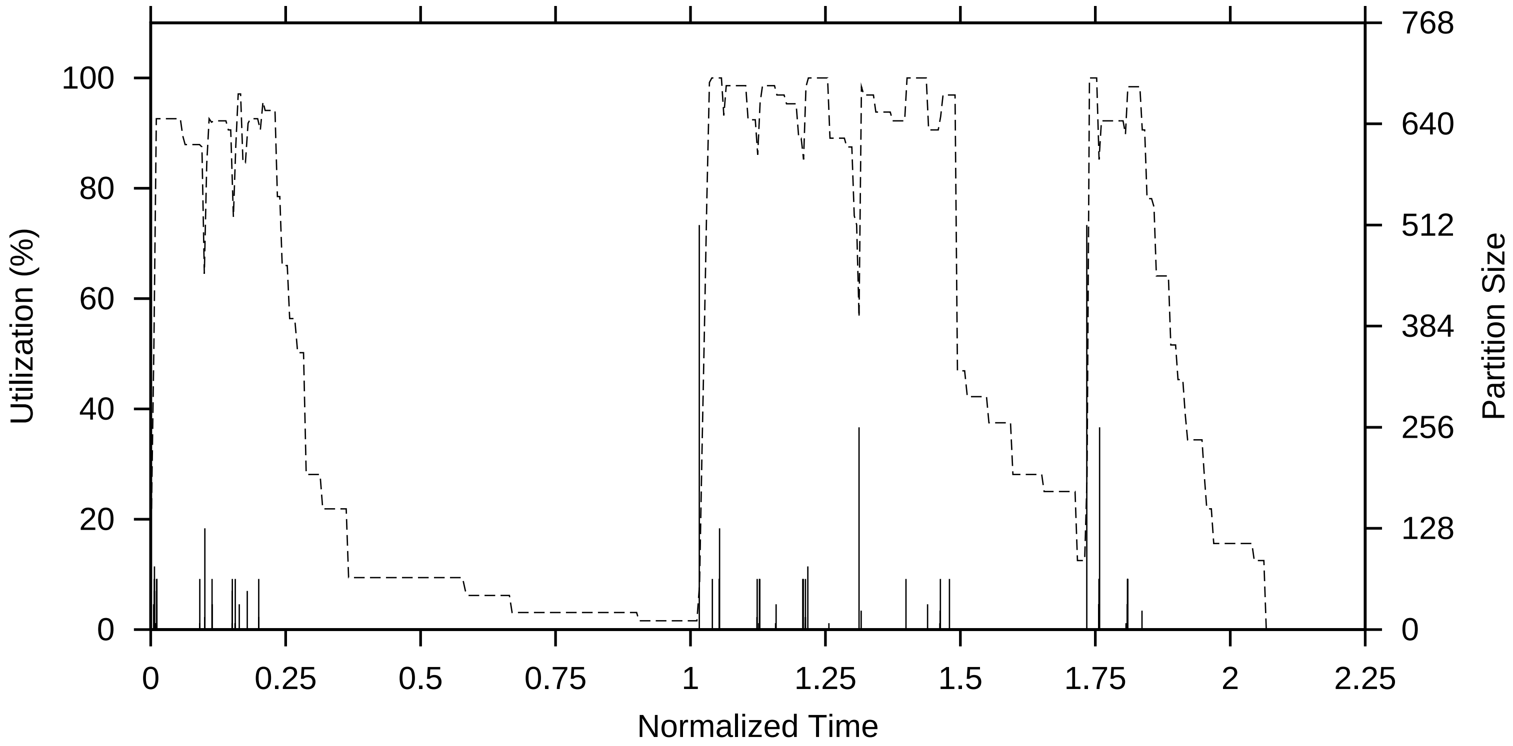

Figure 3: SP : Utilization Chronology