Redbooks Paper

Sheeba Prakash
Barry Spielberg

Effective System Performance Suite on IBM @server pSeries

This Redpaper highlights the improvement in Effective System Performance achieved on the IBM @server pSeries® 655 system. By executing the latest version of ESP suite and using appropriate LoadLeveler® settings, superior performance ratios for ESP (94%) were achieved.

The authors of this Redpaper

This Redpaper was produced by a team of specialists from around the world.

Sheeba Prakash is a staff Engineer/Scientist at the IBM @server pSeries and HPC Benchmark Center. She joined the HPC benchmarking team in 2000 and works on performance analysis of scientific and technical applications for pSeries and xSeries® servers. She holds a master's degree in Traffic Engineering and Transportation Planning and a Master of Science in Computer Science from the University of Houston.

Barry Spielberg is a certified AIX® System Administrator working in the pSeries Benchmark Center. He manages large AIX clusters used for commercial and technical application benchmarking.

Introduction

Effective System Performance (ESP) test is a National Energy Research Scientific Computing Center (NERSC)-designed benchmark suite that provides a metric that focuses on operating system attributes for production-oriented parallel systems. These attributes include parallel launch time, job scheduling, preemptive job launch, and system reboot time. The metric is processor speed independent with low contention for shared resources such as the file system. The mechanics of the test is similar to that of a throughput measurement, but is not a throughput test.

The overall value of a High Performance Computing system depends not only on the raw computational speed, but also on the efficiency of the system management. LINPACK and NAS parallel benchmarks are useful for measuring sustained computational performance, but they give little or no insight into system-level efficiency issues. ESP is designed to give a measure of the effectiveness of a massively parallel system and can aid in the evaluation of High Performance Computing systems.

In order to schedule a set of parallel applications of different partition sizes, it must satisfy the time and partition constraints of the jobs, and at the same time, maximize the system utilization. We consider utilization as the fraction of busy processors to available processors integrated overtime. There are various scheduling strategies, including FCFS, BFF, Backfill, and Gang. First Come First Served (FCFS) means scheduling the job that comes first into queue. The Best Fit First (BFF) schedules the first job that meets the resource requirements. The Backfill scheduler looks at the available resources versus the requested resources by jobs and allocates them in a way that maximizes system utilization. The Gang scheduler helps jobs share time slices by swapping between jobs.

LoadLeveler supports Default (FCFS), Backfill, and Gang schedulers. More details about each of these schedulers is in “LoadLeveler schedulers” on page 13.

Benchmark details

Measuring the elapsed time to process a given workload suite is one measure of system utilization. If the time for each job is constant, regardless of whether it is run alone or concurrently with other jobs, the variation in the elapsed time for the work load depends only on the scheduling strategy and system overhead. Given a workload with jobs of varying partition sizes and elapsed times, then utilization efficiency E can be calculated by:

$$E = \sum p_i t_i / PT$$

Where:

- T = Observed time for the workload
- P = Number of available processors
- p_i = Partition size of ith job
- t_i = Best case elapsed time for ith job

No manual intervention is allowed after the test starts. The order of job submission is determined from a reproducible pseudo-random sequence. With this, an artificially configured queue is obtained specific to this test. At the same time, it provides sufficient queued work to allow flexibility in scheduling. The job sizes and run times in the ESP suite were designed to roughly match the distribution of jobs running on NERSC production systems.

The test could be applied to any system size, with a minimum of a 64-way system. In this study, we tested on 64-way, 128-way, and, 256-way pSeries 655 systems.

The execution of benchmarks consists of the scheduler submitting a fixed number of parallel jobs into the system. The run times of each of these individual jobs are known ahead of the launch. From the known target times of the individual jobs, a hypothetical absolute minimum time (AMT) is computed as work/system size. The elapsed time (ET) from the submission of the first job to the end of the last job is measured. The ESP ratio is measured as the ratio of AMT to ET. In an ideal case, with no overhead and perfect packing of jobs to processors and zero shutdown/reboot time, the efficiency would be unity (one).

We performed three ESP tests:

1. Throughput test ESP1 measures the elapsed time of a batch of equal priority jobs. A set of 228 jobs are submitted to the queue at one-second intervals. The time from the submission of the first job to the end of the run of the last job is measured. This gives the elapsed time (ET1) for ESP1.
2. Multimode test ESP2 measures the elapsed time of a batch of jobs that includes the submission of higher-priority jobs into the queue. Two high-priority jobs are submitted after 40 minutes from the submission of first job into queue and another after two hours. The time from the submission of the first job to the end of the run of the last job is measured. This gives the elapsed time (ET2) for ESP2.
3. The third measurement, ESP3, is the system reboot time. This is the time required for the system to shut down, come back up, and function again. This elapsed time is ET3.

The ESP ratios are calculated as:

```
ESP1 = AMT1/ET1
ESP2 = AMT2/ET2
```

The final ESP ratio is calculated as:

```
ESP = (AMT1+AMT2)/(ET1+ET2+ET3)
```

Results and discussion

This section describes the details of our runs.

Calculation of absolute minimum time

ESP tests are batch tests consisting of 230 jobs of 14 job types, each of which has a target run time. The absolute minimum time (AMT) for throughput and multimode tests are computed, as shown in Table 1.

Table 1 AMT calculation

				For 128-way system			For 256-way system		
Job type	**Fractional size**	**Number of copies**	**Target run time**	**Job CPU count**	**Job work**	**Total work**	**Job CPU count**	**Job work**	**Total work**
A	0.03125	75	257	4	1028	77100	8	2056	154200
B	0.0625	9	341	8	2728	24552	16	5456	49104
C	0.5000	3	536	64	34304	102912	128	68608	205824
D	0.25	3	601	32	19232	57696	64	38464	115392
E	0.50	3	312	64	19968	59904	128	39936	119808

				For 128-way system			For 256-way system		
Job type	**Fractional size**	**Number of copies**	**Target run time**	**Job CPU count**	**Job work**	**Total work**	**Job CPU count**	**Job work**	**Total work**
F	0.0625	9	1846	8	14768	132912	16	29536	265824
G	0.125	6	1321	16	21136	126816	32	42272	253632
H	0.1582	6	1078	20	21560	129360	40	43120	258720
I	0.03125	24	1438	4	5752	138048	8	11504	276096
J	0.0625	24	715	8	5720	137280	16	11440	274560
K	0.0957	15	495	12	5940	89100	24	11880	178200
L	0.125	36	369	16	5904	212544	32	11808	425088
M	0.25	15	192	32	6144	92160	64	12288	184320
Z	1.0	2	100	128	12800	25600	256	25600	51200
	Total	230			176984	1405984		353968	2811968
AMT1 (sec)							10784		10784
AMT1 (hrs)							3.0		3.0
AMT2 (sec)							10984		10984
AMT2 (hrs)							3.05		3.05

The absolute minimum time is computed as the total work divided by the system size**.** As illustrated in Table 1 on page 3, we see that AMT is the same, regardless of the system size:

```
AMT1= 10784 seconds
AMT2 =10984 seconds
```

Benchmark execution

Throughput, multimode, and shutdown/reboot time were measured separately for the following system configuration.

System components

Our system was a cluster of 32 pSeries 655 1.1 GHz nodes, dual plane SP2® switch configuration, as shown inTable 2.

Table 2 System components

System	**pSeries 655**
Processor	POWER4™, 1.1 GHz
Number of processors	8
Memory	32 GB
Large pages	16 GB

System	pSeries 655
Kernel	32 bit
Operating system	AIX 5L™ V5.1, Maintenance Level 3, plus fixes
Compilers	C Compiler, xlc 6.0
LoadLeveler	3.1
File system	GPFS

Program analysis

In this section, we analyze some of the data we collected.

Variation in the run time of individual jobs (ESP1)

Table 3 shows the variations in individual jobs from the execution of ESP1, 256 way run. As shown in the table, we observed no significant variation in the run times of the individual jobs.

Table 3 Variation in real time of individual jobs

			Percentage variation in real time	
Job type	**Fractional size**	**Number of copies**	**Below target time (%)**	**Above target time (%)**
A	0.03125	75	0	2
B	0.0625	9	1	3
C	0.5	3	3.5	0
D	0.25	3	2	0.5
E	0.5	3	0.5	2
F	0.0625	9	4	12
G	0.125	6	2	2
H	0.1582	6	2	4
I	0.03125	24	1	0
J	0.0625	24	2	7
K	0.0957	15	5	4
L	0.125	36	4	3.5
M	0.25	15	7	2

Results from MPI profiling

We performed further program analysis using the message passing interface (MPI) trace library. Table 4 on page 6 summarizes the results.

Table 4 Communication-intensive application

# Procs	Comm time	Elapsed time	CPU time	% Communication
256	918.8	989.9	939.2	**92.8**
128	556.9	592.9	564.9	**93.9**
32	272.4	298.1	282.4	**92.1**
16	1041.2	1154.9	1043.3	**90.2**
8	556.24	995.8	978.4	**55.9**

The ESP suite contains an application, pchksum, that does exclusive OR (XOR) operations repetitively. The MPI trace library, developed by Bob Walkup of IBM® Research and distributed by ACTC IBM, was used in the analysis of the code in pchksum. As we can see from Table 4, the application is communication intensive, with more than 90% of its elapsed time spent in communication when run with multiple nodes. Therefore, the performance of pchksum depends to a great extent on the communication subsystem. The test results for the 256-way, 128-way, and 64-way systems indicate that the number of nodes does not affect the percent of ESP time in communication.

However, for large systems and those runs with system size such that the job types are not multiples of eight (size of the node), the performance of the communication subsystem might have an impact on ESP. But this impact might not be significant because of the way ESP is designed. The user has the option to specify target time, which is passed on as an argument to the executable. The application then runs for that specific period of time.

System utilization

The log files generated during ESP runs were further analyzed to get the CPU utilization during the ESP execution time. Figure 1 shows system utilization over time.

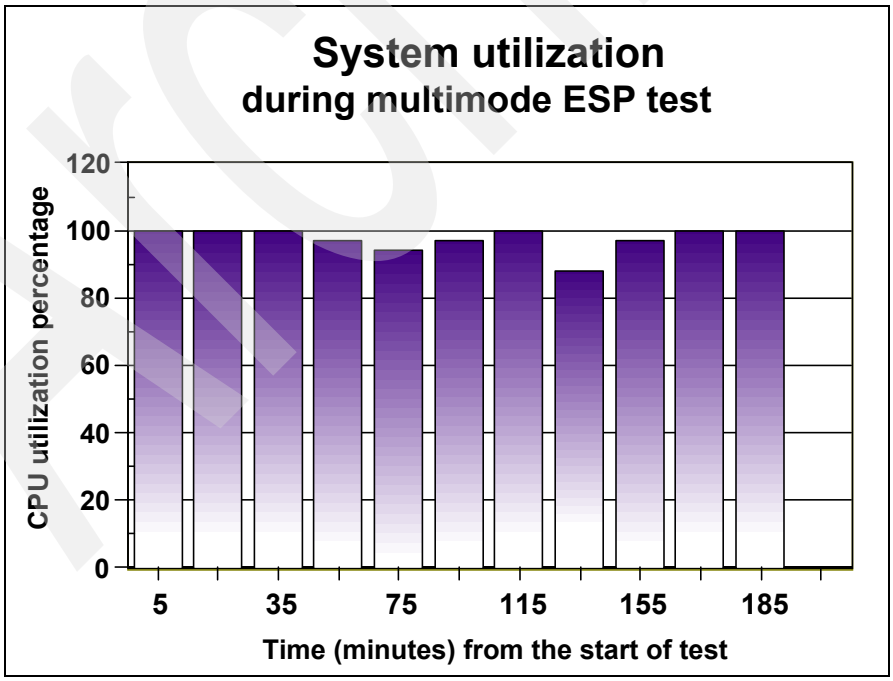

Figure 1 CPU utilization during ESP test

We observed that more than 95% CPU utilization could be attained during the execution of the ESP test when the runs were done with appropriate LoadLeveler settings. Table 9 on page 14 gives the time and system utilization in tabular form. That data corresponds to the 256-way ESP2 run. “Run script” on page 13 also provides a sample run script with LoadLeveler settings.

Preemption (ESP2)

Checkpoint/restart, swapping, migrating, and preemption all depend on the kernel operating with preemption by changing process images between run and idle states and moving them to memory and disk. The preemption feature is supported by the Gang scheduler, which is used in the multimode test case (ESP2) run. As you recall, in this run, the higher-priority jobs are submitted into the queue after 40 minutes from the submission of the first jobs and the second priority job after two hours. Immediately after the submission of these jobs, they get scheduled, preempting the other jobs. This includes the running jobs, which all stop and go into a preempted stage. After the execution of the higher priority jobs completes, these preempted jobs get rescheduled and run to completion. See Table 5.

Table 5 Snapshots of queue during preemption

Time (minutes)	Time stamp	Run jobs	PEs busy	Jobs in queue	Preempted jobs
First instance of preemption					
25	15243	24	256	150	0
35	15855	24	256	130	0
39	16160	24	256	118	0
40	16160	Z job comes into queue.			
	16164	Z job gets scheduled and starts running.			
40m 30s	16191	1	256	118	24
	16221	1	256	118	24
	16252	1	256	118	24
	16262	Z job finishes.			
43	16282	24	256	118	0
45	16466	24	256	112	0
55	17077	18	248	89	0
Second instance of preemption					
115	20576	8	256	41	0
120	20942	8	256	38	0
120	20942	Z job comes into queue.			
	20946	Z job gets scheduled and starts running.			
2h 0m 30s	20972	1	256	38	
	21003	1	256	38	8
	21033	1	256	38	8
	21044	Z job finishes.			

Time (minutes)	Time stamp	Run jobs	PEs busy	Jobs in queue	Preempted jobs
122	21064	8	256	37	0
125	21185	8	248	35	0

Note: A Z job has a higher priority.

Table 5 on page 7 shows the snapshots of scheduler status during preemption. As we can see from the table, there is negligible overhead due to preemption during ESP testing. It took about four seconds for the running jobs to stop and go into preempted stage while the priority job got scheduled.

Shutdown/reboot time (ESP3)

The shutdown/reboot time varies with the system configuration. This includes interfaces such as Ethernet/switch interface, switch configuration, and I/O configuration (number of disks mounted on the system and disk size). Another factor to be considered is the weight to be given for the shutdown/reboot time. This also is dependent on the production environment as to how often the system gets shut down. If we consider that the system gets shut down once in 14 days, the weighting factor for system reboot time can be decreased by a factor of 14.

Results from test runs

Test runs that were done to arrive at the optimum LoadLeveler options are shown in Table 6. Table 10 on page 15 shows the results of more runs.

Table 6 Results from test runs

Test	# Procs	Type of scheduler	LoadLeveler options	ESP ratio
Thru	256	Gang	Not-shared, WCL=36000, US	0.82
Thru	256	Gang	Not-shared, US	0.82
Thru	256	Back	Shared, WCL=1.2 run time+100, US	0.84
Thru	256	Gang	Shared, blocking= unlimited, US	0.93
Thru	256	Gang	Shared, blocking=8, US	0.943
Thru	256	Gang	Shared, blocking=8, IP	0.95
Multi	256	Gang	Shared, blocking=8, IP	0.94
Multi	256	Gang	Shared, blocking=8, US	0.95

Table notes:

- Thru = throughput test.
- Multi = multimode test.
- WC L= wall clock limit.
- In all tests, the environment variable MP_SHARED_MEMORY is set to yes.

From Table 6, you can see LoadLeveler settings using the options of *node_usage=shared*, *blocking=8*, and *network.mpi=csss, shared, ip* gives the best ESP performance ratio for throughput measurement. LoadLeveler settings using the options of *node_usage=shared*, *blocking=8*, and *network.mpi=csss, shared, us* worked best for multimode testing. There was

no significant difference in the results of both throughput and multimode measurements when IP or US interface was used with the these LoadLeveler settings.

Based on the above results, optimum LoadLeveler settings are *node_usage=shared*, *blocking=8*, and *network.mpi=csss, shared, us/ip* for both the throughput and multimode ESP measurements using the Gang scheduler. An Effective System Performance of 94 to 95% was obtained with the these settings. For a sample run script with the these LoadLeveler settings, see “Run script” on page 13.

We observed (in tests #1 and #2 in Table 10 on page 15) that when node_usage=not_shared is used with the Gang scheduler, the scheduler tends to behave in an FCFS manner. These tests correspond to the wall_clock_limit set to the very high value of 10 hours (test 1) and when wall_clock_limit was not specified in the run scripts (test 2). The log files from the runs indicate that the E job, which is a large 128-way job and is 14th in queue, delays the scheduling of rest of the jobs that are behind it in queue. Here, setting the user priority of large jobs to a higher value would help in getting the large jobs to get scheduled first, and thus eliminate the scenario of idle CPUs. In addition, when the job types are not multiples of the size of node (in our case 8), it results in idle CPUs, and thus lower system utilization.

Another observation is regarding the jobs that were scheduled with the Backfill scheduler for which the wall_clock_limit setting was set to 20% more than the target run time plus 100 seconds. Here (see tests #6 and #9 in Table 10 on page 15), after the large 128-way E job came into queue, the next job to be scheduled was the 64-way M job. It had a minimum wall_clock_limit, although there were other jobs that required less CPUs ahead of it in queue, but with higher wall_clock_limit settings.

During multimode testing, we observed that when Z jobs with higher priority came into the queue, LoadLeveler determined the time T_{begin} at which previous running jobs would finish and free up enough nodes to run Z. Then, it backfilled that spare time with other jobs in the queue by looking at their CPU requirements and wall_clock_limit to see if they would finish before T_{begin}.

Scalability

We did runs for 64-way, 128-way, and 256-way system sizes. Good scalability was observed for the ESP ratios of 64-way, 128-way, and 256-way runs, producing an average of ESP ratio 0.94. This is shown in Table 7 and in Figure 2 on page 10.

Table 7 Results of 64-way, 128-way, and 256-way runs

Number of procs	Test	ESP ratio
64	Throughput	0.935
128	Throughput	0.943
256	Throughput	0.943

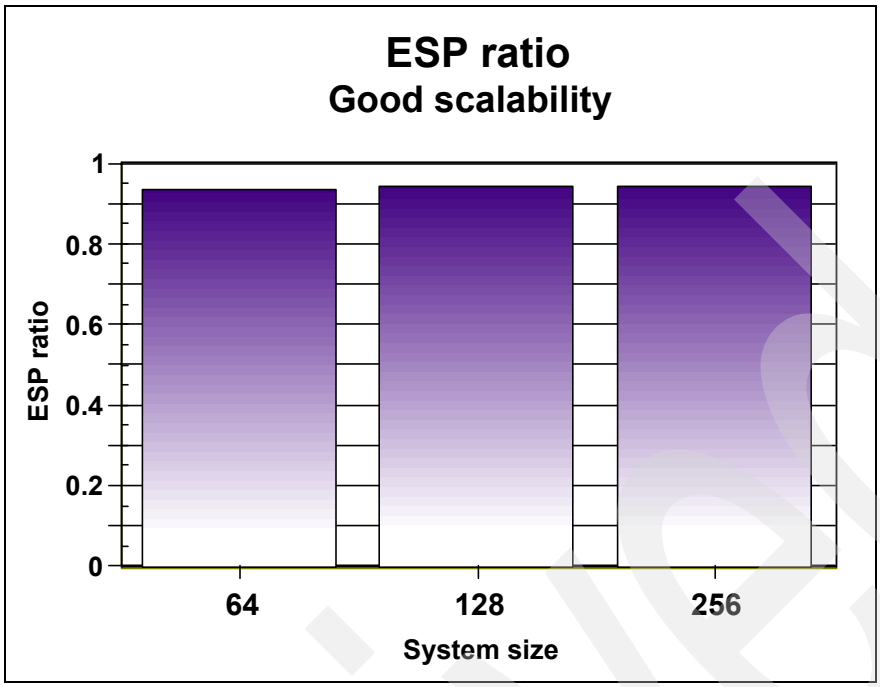

Figure 2 Scalability of ESP

Results

Table 8 summarizes the results for the 256-way runs for which we have maximum ESP ratios.

Table 8 Results for 256-way runs

Type of scheduler	Test	System size	Elapsed time (ET) seconds	ESP ratio
Gang	Throughput (ESP1)	256	11441	**0.943**
Gang	Multimode (ESP2)	256	11529	**0.95**

LoadLeveler settings

We used the following LoadLeveler settings:

- node_usage=shared
- blocking=8
- network.mpi=csss
- shared
- us
- environment MP_SHARED_MEMORY=yes

"Run script" on page 13 provides the corresponding run script.

Final ESP ratio considering system reboot time

The shutdown for a single p655 node was measured to be about 13 minutes on a 16 p655 nodes cluster. We observed that 16 nodes took about 15 minutes to shut down and reboot. The shutdown/reboot time was measured from issuing the command **`shut down -fr`** to the time system was up and the SP switch was up. The measurements were done on the cluster with p655 nodes with the configuration given in Table 2 on page 4.

If the system is shut down in parallel, 32 nodes could take about 15 minutes to reboot.

System shutdown and reboot time (ET3) equals 900 seconds.

The final ESP ratio is measured from ESP1, ESP2, and ET3 as follows:

1. If the system reboot time is given equal weight as the throughput and multimode tests:

   ESP = (AMT1+AMT2)/(ET1 + ET2+ET3) = (10784+10984)/(11441+11529+900) **= 0.91**

2. If the system reboot time is given less weight by a factor of seven than the throughput and multimode tests, assuming that the system gets shut down once a week so that its weight is reduced by that factor:

   ESP = (AMT1+AMT2)/(ET1 + ET2+ET3/7) = (10784+10984)/(11441+11529+900/7) = **0.94**

3. If the system reboot time is given less weight by a factor of 14 than the throughput and multimode tests, assuming that the system gets shut down once a week:

   ESP = (AMT1+AMT2)/(ET1 + ET2+ET3/14) = (10784+10984)/(11441+11529+900/14) = **0.945**

Conclusions

This study helped improve the Effective System Performance. The final ESP1 and ESP2 ratios suggest a 94 to 95% system effectiveness for a 256-way pSeries 655 system. A final ESP ratio is still above 90% even when shutdown/reboot time is also considered with equal weight along with throughput and multimode tests. A 94% ESP ratio was calculated when shutdown/reboot time is given less weight, depending on the frequency of shutdown.

The study helped to identify optimum LoadLeveler settings to be used during the throughput and multimode tests of ESP for maximizing system effectiveness. The ESP measurements tested the various features of LoadLeveler, including preemption, the effects of altering LoadLeveler settings, and the effectiveness in scheduling parallel jobs. Therefore, this could be used as a testing methodology for schedulers.

Further work in this area could include

- Studying the ESP scaling on very large pSeries clusters
- Studying a different scheduler, such as PBS, on pSeries for a comparison of schedulers
- Extending the study to Linux clusters using PBS

The ESP suite can be downloaded from the NERSC Web site:

http://www.nersc.gov/aboutnersc/esp.html

References

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

- *Workload Management with LoadLeveler*, SG24-6038
- Adrian Wong, Leonid Oliker, William Kramer, Teresa Kaltz, and David Bailey, “System Utilization Benchmark on the Cray T3E and IBM SP,” National Energy Research Scientific Computing Center (NERSC), Sixth Workshop on Job Strategies for Parallel Processing, April 19, 2000

- Adrian Wong, Leonid Oliker, William Kramer, Teresa Kaltz, and David Bailey, “Evaluating System Effectiveness in High Performance Computing Systems,” National Energy Research Scientific Computing Center (NERSC), LBNL Technical Report #44542, 1999
- Adrian T. Wong, Leonid Oliker, William T.C. Kramer, Teresa Kaltz, and David Bailey, “ESP: A System Utilization Benchmark,” National Energy Research Scientific Computing Center (NERSC), IEEE, 2000
- NERSC Web site

  http://www.nersc.gov/aboutnersc/esp.html

Acknowledgments

We are grateful to Christine Kostek and Sharon Selzo, Benchmark Center, Poughkeepsie, for the project management and system time for the successful completion of this project.

We also express our gratitude to Enci Zhong, Waiman Chan, and Sebastian Athiyunthan, LoadLeveler Development team, Poughkeepsie, for their support in dealing with issues related to schedulers and preemption.

We are grateful to our management, especially Clarisse Taaffe-Hedglin, pSeries and HPC Benchmark Center, for providing us with the resources and opportunity to work on this benchmark.

We are grateful to NERSC for making the ESP software available to the public, which made it possible for us to access the latest ESP benchmark suite and to execute the same.

Appendix A

In this appendix, we provide additional details about the schedulers and our runs.

LoadLeveler schedulers

The LoadLeveler schedulers are:

Default scheduler, LL_DEFAULT

For serial jobs, LoadLeveler simply looks at the jobs at the top of the queue. If there is a node available that satisfies its requirements, the job will run.

For parallel jobs, the Default scheduler attempts to accumulate sufficient resources to run the job. If there are not currently enough nodes available to run the job, the available nodes are reserved by LoadLeveler for this job until enough nodes become available to run it.

Disadvantages

As more and more nodes are reserved for a parallel job, the system as a whole becomes less utilized. Often parallel jobs do not accumulate enough nodes within the hold time (NEGOTIATOR_PARALLEL_HOLD) limit to allow them to run.

Backfill

The Backfill scheduler provides more flexibility in scheduling. It lets the user specify the wall_clock_limit setting. Given a start time and the maximum elapsed time, LoadLeveler can calculate the latest end time for the job, and thus knows the node availability when that job finishes.

Apart from scheduling parallel jobs more efficiently, it aids multiple user space tasks per adapter and supports multiple tasks per node. Neither of these features are supported by the Default scheduler.

Gang

When jobs are submitted with the Default or Backfill schedulers, a parallel job would continue to execute uninterrupted until completion. The Gang scheduler allows two or more jobs to be simultaneously active on a node and for execution to swap between the jobs at regular intervals. The CSS software has been enhanced so that a process can relinquish a user space window, but remain active within the system. This window can be used by another job.

Run script

A sample run script with the identified optimum LoadLeveler settings for ESP testing is given in Example 1. We explain this run “System utilization” on page 6.

Example 1 Sample run script

```
# @ job_type         = parallel
# @ job_name         = A_008_000
# @ output           = /gpfs/fs2/sheeba/esp2/logs/A_008_000.out
# @ error            = /gpfs/fs2/sheeba/esp2/logs/A_008_000.out
# @ notification     = never
# @ node_usage       = shared
# @ class            = Y_Class
# @ wall_clock_limit = 36000,36000
# @ blocking         = 8
### @ blocking = unlimited, when the job size is not a multiple of 8 ###
```

```
# @ total_tasks      = 8
# @ network.MPI = csss,shared,us
# @ environment      = MP_SHARED_MEMORY=yes
#
# @ queue
ESP=/gpfs/fs2/sheeba/esp2
t0=`$ESP/Epoch`
echo `$ESP/Epoch` " S " A_008_000 "  Seq# 192" >> $ESP/LOG
/usr/bin/time $ESP/bin/pchksum -v -t  257

echo `$ESP/Epoch` " F " A_008_000 "  Seq# 192" >>
$ESP/LOG
```

Detailed results of runs

Table 9 shows a chart of the CPU utilization from the start of the submission of the jobs. This is also explained in “System utilization” on page 6.

Table 9 Utilization by time

Time (minutes)	PEs	% Utilization
5	256	100
15	256	100
25	256	100
35	256	100
45	256	100
55	248	97
65	256	100
75	240	94
85	248	97
95	248	97
105	256	100
115	256	100
125	248	97
135	224	88
145	240	94
155	248	97
165	232	91
175	256	100
185	256	100
192	0	0

We discuss Table 10 in “Results from test runs” on page 8.

Table 10 Test details

#	Test	# Procs	Scheduler	LoadLeveler options	US/IP interface	Elapsed time, ET (sec)	ESP ratio (AMT/ET)	Comments
1	Thru	256	Gang	Not-shared, wall clock=36000 sec	US	13151	0.82	Wall clock set to high value
2	Thru	256	Gang	Not shared	US	13178	0.82	Wall clock time not specified
3	Thru	256	Gang	Shared, blocking=8	US	11441	0.943	
4	Thru	256	Gang	Shared, blocking=unlimited	US	11609	0.93	
5	Thru	256	Gang	Shared, blocking=8, @network.mpi=csss, shared, ip	IP	11303	0.95	
6	Thru	256	Back	Shared, blocking=8, wall clock=20% + 100	US	12772	0.84	
7	Multi	256	Gang	Shared, blocking=8, @network.mpi=csss, shared, ip	IP	11668	0.94	
8	Multi	256	Gang	Shared, blocking=8	US	11529	0.95	
9	Multi	256	Back	Blocking=8, shared, wall_clock_limit=20 %+100	US	12902	0.85	
10	Thru	128	Gang	Blocking=unlimited, shared	US	11539	0.935	
11	Thru	128	Gang	Not_shared	US	13290	0.81	
12	Thru	128	Gang	Blocking=8 for jobs that are multiples of 8, unlimited for others	US	11437	0.943	
13	Thru	64	Gang	Blocking=unlimited, shared	US	11791	0.915	
14	Thru	160	Back	Shared, blocking=8; blocking=unlimted for rest, wall_clock_limit=360 00 sec	IP	13902	0.78	

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an Internet note to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	IBM®	pSeries®
AIX 5L™	ibm.com®	Redbooks(logo) ™
@server™	LoadLeveler®	SP2®
@server™	POWER4™	xSeries®

Other company, product, and service names may be trademarks or service marks of others.