



Redbooks Paper

Balaji V. Atyam
Raj Panda
Angelo Rossi
Carlos P. Sosa

Single Processor Performance of Scientific Applications on AIX and Linux on POWER Systems

Abstract

In an earlier report, we presented the performance of several scientific applications on AIX® and SUSE LINUX for the IBM @server® pSeries® systems. In this report, we expand our original study by including performance analysis on the most recently announced POWER4™+ family of IBM @server pSeries systems. In our latest study, we show that the pSeries running either AIX or Linux® represent a valuable combination to carry out scientific calculations in several fields. We also look at various compilers' performance and provide information about the current status of Linux on POWER™ series.

Introduction

In our previous study we mentioned that Linux is becoming a popular alternative to proprietary systems like Windows® NT or UNIX® at the low end, and continues to increase its presence on high-end systems [1]. As a result, most of the major hardware and software vendors have announced plans to support Linux and are playing an active role in Linux's evolution—working on additional features more suited to enterprise users, in particular with enhanced scalability as the top priority.

Linux on 64-bit POWER for IBM® Systems (PPC64) has become one of the important IBM eServer™ Linux strategies [2]. The AIX pSeries servers are known for industry-proven performance, reliability and scalability. Linux on POWER can leverage the enterprise-level advantages of pSeries, but this market depends on the eServer enablement team to facilitate its maturation. The progression of ISV software to Linux/POWER is aided by the arrival of high-performance IBM VisualAge® for C/C++ and IBM XL Fortran compilers [3].

In this study we analyze the performance of a number of scientific applications, mainly in Life Sciences, on AIX and SUSE Linux for POWER with the latest IBM compilers. Also a comparison study is made between IBM and GNU compilers to demonstrate the performance advantage of IBM compilers on pSeries systems. In the next section, the different features of the pSeries systems used in this report are summarized. In subsequent sections, brief descriptions of applications and performance comparisons are presented.

Hardware

In our previous study, we used 1.0 GHz p630 POWER4 and 1.1 GHz p655 POWER4 systems [1]. In the current study we extend the number of systems tested by including a 1.7 GHz p655 POWER4+, a 1.2 GHz p630 POWER4+, and a 1.45 GHz p615 POWER4+ [4-6].

The p655 pSeries Server is the latest AIX and Linux server from IBM, with the POWER4+ Multi Chip Module (MCM) at the core of this latest architecture [4-6]. Many of the systems that we tested here are 4-way, but an 8-way system was also used for some of the tests. The building blocks for the systems utilized here are an 8-way MCM running at 1.1 GHz and a 4-way MCM running at 1.7 GHz. One key difference between the p630 and p615 class systems and the p655 class systems is that the packaging in the former set uses a Single Chip Module (SCM) [5]. A full description of the POWER4 architecture is beyond the scope of this work. See numbers 4-7 in "References" on page 22 for further details.

Software

We have configured the benchmark systems with the following software (OS and compilers) [3]:

- ▶ AIX 5.1_ML4, IBM XL Fortran for AIX v8.1.0.4, IBM Visual Age C++ for AIX 6.0.0.3
- ▶ SUSE Linux Enterprise v8.0 + Service Pack 1 for PowerPC®, IBM XL Fortran for Linux v8.1, IBM Visual Age C++ for Linux v6.0
- ▶ SUSE Linux Enterprise v8.0 + Service Pack 1 for PowerPC, GNU compilers (gcc and g77) v3.3.1

Benchmarks

All the benchmarks used in this study are selected from the areas of Life Sciences, Oil and Gas, and Particle Physics. Table 1 summarizes the different applications chosen for this report [1].

Table 1 List of applications used in this study

Area	Application	Version
Life Sciences	GAMESS	02/16/2002(R1)
	GROMACS	3.1.4
	AMBER	7
	FTDOCK	2.0
	AutoDock	3.0
	BLAST	2.2.6
	FASTA	3.4t22
	CLUSTAL W	1.83
	Simwalk2	2.80
	HMMER	2.3.1
Oil and Gas	DMO	
Particle Physics	MILC	6

It should be noted that this set of applications does not represent all the currently available applications for Linux on POWER, but were chosen at a time when these applications were available to us when we started this study. We are

currently porting applications as they become available, and additional publications are planned to will be submitted in the near future. There are also efforts to port applications in the commercial area [2].

Results and discussion

In this section we discuss the results.

GAMESS

The General Atomic and Molecular Electronic Structure System (GAMESS) is a general *ab initio* quantum chemistry package [8]. GAMESS is maintained by the members of the Gordon research group at Iowa State University. Briefly, GAMESS can compute SCF wave functions such as RHF, ROHF, UHF, GVB and MCSCF. Correlation corrections to these SCF wave functions include Configuration Interaction, second order perturbation theory, and Density Functional Theory approximation. Analytic gradients are available for automatic geometry optimization, transition state searches, or reaction path following. Computation of the energy Hessian permits prediction of vibrational frequencies. These methodologies are well documented in the literature and it is beyond the scope of this study to describe each method [8].

GAMESS is a floating point intensive application written in Fortran and C. From a performance point of view this type of code relies on linear algebra constructs. So it is not uncommon to find extensive use of “fundamental loops,” for example, DAXPY, DDOT, and matrix-multiplication as well. The extent of the use of these operations is code and functionality dependent. Also, I/O can be a major component of certain options in the program. However, in this study we make no attempt to look at I/O performance.

We have considered two benchmarks, both based on the Restricted Hartree-Fock approximation (RHF) [8]; RHF/6-31G(d) frequency calculation on Adamantane ($C_{10}H_{16}$) T_d symmetry and RHF/6-31G(d) frequency calculation on Pentenal (C_5H_8O) C_1 symmetry, provided by Gordon research group at Iowa State University. Adamantane has 182 basis functions and Pentenal has 106 basis functions.

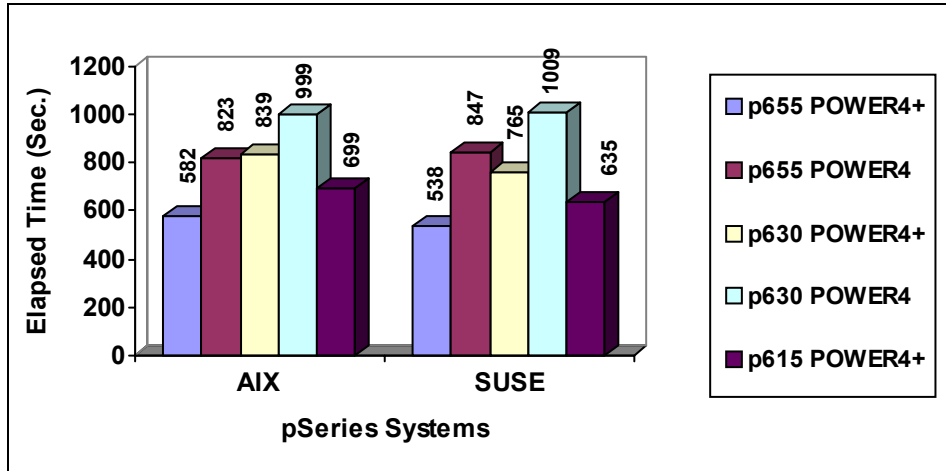


Figure 1 GAMESS Adamantane RHF/6-31G(d) frequency benchmark

In general, when looking at the performance of the different applications we focus our attention on the following: Performance differences of the same run with the operating systems (OS) on different platforms, on the same platform with different OS, and on the same platform with different compilers, where applicable.

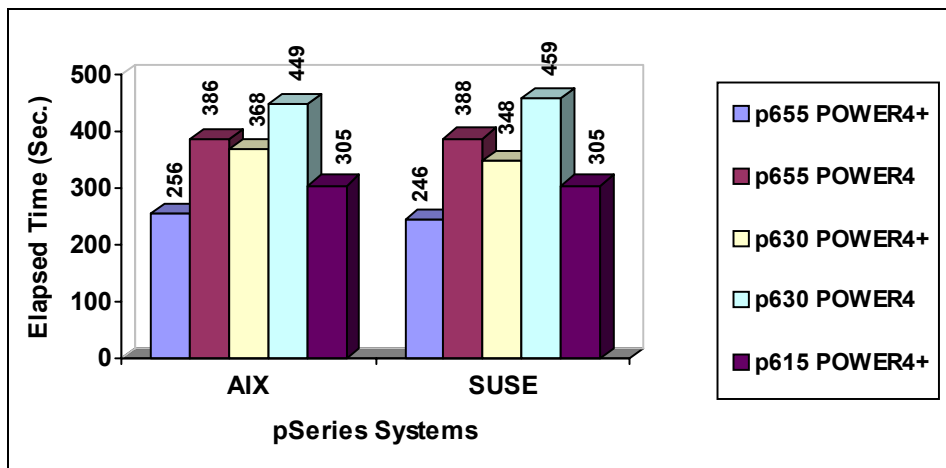


Figure 2 GAMESS Pentenal RHF/6-31G(d) frequency benchmark

A comparison of the performance difference for Adamantane and Pentenal when going from p655 (or p630) POWER4 to p655 (or p630) POWER4+ is consistent with the speedup expected by simply extrapolating using the clock speed ratio for both machines. A minimal improvement in the speedups is seen for Pentenal

when compared to Adamantane. This may be attributed to the fact that Pentenal is a smaller system than Adamantane and may fit better in cache (mainly L2 cache).

Unlike in our previous study, this time the present benchmarks carried out on the eServer Linux PPC64 tend to be faster by 0 to 9 percent than the same applications running under AIX. The faster timings on pSeries Linux PPC64 may be attributed to more mature compilers. The consistency of the results between the two benchmarks should not be surprising since these are similar runs and also because the benchmarks exercise the same sections of the code.

GROMACS

Groningen Machine for Chemical Simulation (GROMACS) is a molecular mechanics and molecular dynamics application [9] and is primarily designed to simulate biochemical molecules based on Newton's equations of motion. GROMACS makes use of fast algorithms when computing nonbonded interactions that usually dominate simulations. Many groups are also using it for research on non-biological systems such as polymers [10].

We have taken three cases to analyze the performance of this application:

- ▶ *dppc*: A phospholipid membrane, consisting of 1024 dipalmitoylphosphatidylcholine (DPPC) lipids in a bilayer configuration with 23 water molecules per lipid, for a total of 121,856 atoms. It was simulated with a twin-range group-based cut-off of 1.8 nm for electrostatics and 1.0 nm for van der Waals interactions. The long-range Coulomb forces between 1.0 nm and 1.8 nm were updated every tenth integration step during the neighbor list generation. This simulation was performed for 100 iterations.

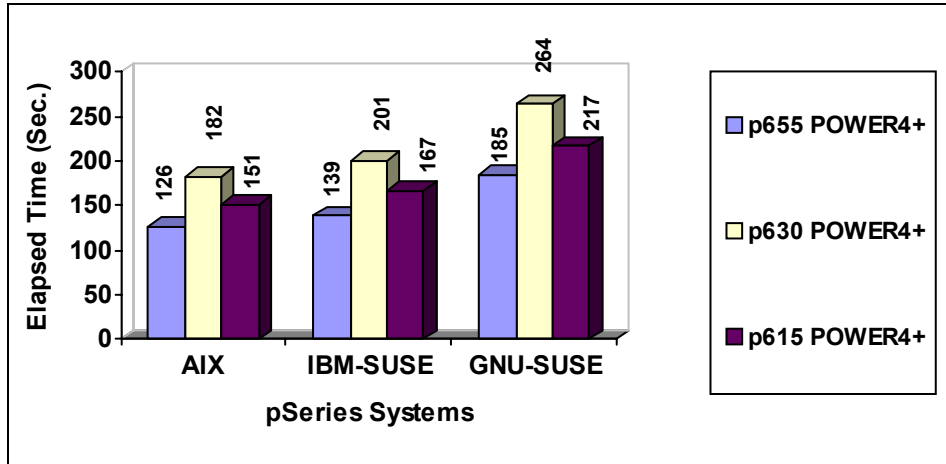


Figure 3 *dppc GROMACS benchmark*¹

- *Villin*: The system was simulated with 3000 water molecules in a truncated octahedron unit cell (slightly less than 10,000 atoms in total), using a group-based cut-off for both electrostatic and Lennard-Jones interactions at 0.8 nm. This simulation was performed for 5000 iterations.

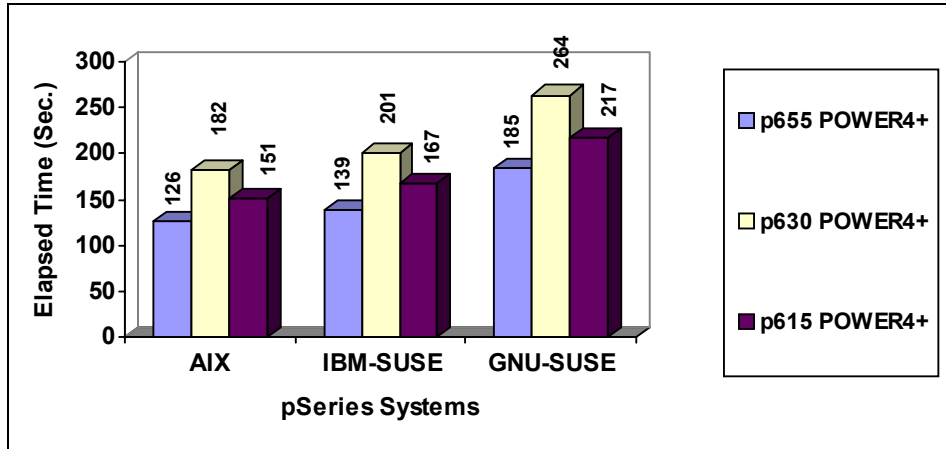


Figure 4 *Villin GROMACS benchmark*¹ on page 7

- *Poly-CH2*: A 6000-unit polyethylene molecule modeled with anisotropic united atoms. It means the bonded forces act on the position of the carbon, while the Lennard-Jones interaction site is a united atom displaced to the center of the CH2 group. Although the anisotropic interaction sites increase

¹ GNU-SUSE refers to the application compiled with the GNU compilers [11] on the same OS.

the particle number to 12,000 they are easily implemented as dummy atoms, keeping the computational cost very close to that of a 6000 particle system. This simulation was also performed for 5000 iterations.

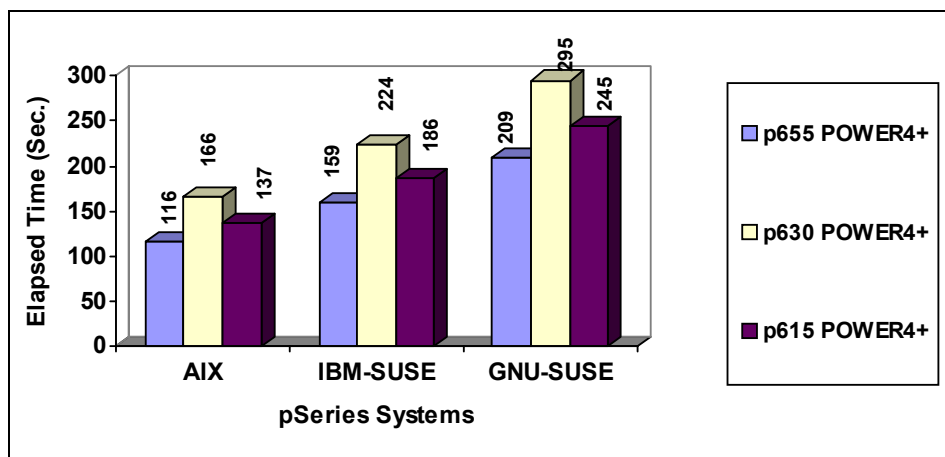


Figure 5 Poly-CH² GROMACS benchmark¹ on page 7

The dppc, villin, and Poly-CH₂ benchmark numbers for GROMACS are depicted in Figure 3 on page 7, Figure 4 on page 7, and Figure 5, respectively. We have presented three types of numbers: AIX with IBM Fortran and C compilers, SUSE Linux with IBM Fortran and C compilers, and SUSE Linux with GNU g77 and gcc compilers (GNU-SUSE) on IBM pSeries systems. In this case we do not include the numbers from our previous study because they were performed with an older version of GROMACS (Version 3.1.2). This is to simplify the performance analysis.

The results that we have found with the latest GROMACS version are consistent with what we found in our previous study. In general we found that for these examples, GROMACS tends to be faster when running under AIX on POWER. On p655 POWER4+ and p630 POWER4+ GROMACS tend to be faster on AIX for dppc, villin, and poly-CH₂ by about 10 percent, 10 percent, and 37 percent, respectively.

Similarly as before, our results for these examples indicate that IBM compilers are superior over the GNU compilers for Linux. This result is not surprising since the IBM compilers are optimized for the pSeries. On p655 POWER4+ and p630 POWER4+, Linux with IBM compilers performed better than Linux with GNU compilers on average by 32 percent, 32 percent, and 31 percent for dppc, villin, and Poly-CH₂, respectively.

AMBER

Assisted Model Building with Energy Refinement (AMBER) is a flexible suite of programs for performing molecular mechanics and molecular dynamics calculations based on force fields [12]. Sander is the primary program used for molecular dynamics simulations and is the only program considered in our current study. Sander carries out energy minimization, molecular dynamics and NMR refinements. AMBER is floating point intensive Fortran code. The version used in this study corresponds to AMBER7 for IBM systems [13]. In this study we extend our previous work by running the jac and gb_mb benchmarks on pSeries POWER4+ systems [1].

- ▶ *Jac*: This is a joint Amber-CHARMM benchmark. It considers a protein dhfr (dihydrofolate reductase) in an explicit water bath with cubic periodic boundary conditions. Details of system size and simulation conditions are: 23,558 atoms, cubic periodic box, 62.23 Å dimension, 9Å nonbond cutoff with 2Å buffer, that is, list with 11Å cutoff, 1 fs time step, 1000 steps, NVE ensemble (constant energy, constant volume), bonds to hydrogen constrained (SHAKE), The particle mesh Ewald (PME) method was used for calculating the Lennard-Jones (LJ) and electrostatic interactions with 64x64x64 grid, equilibration temperature was 300 K.

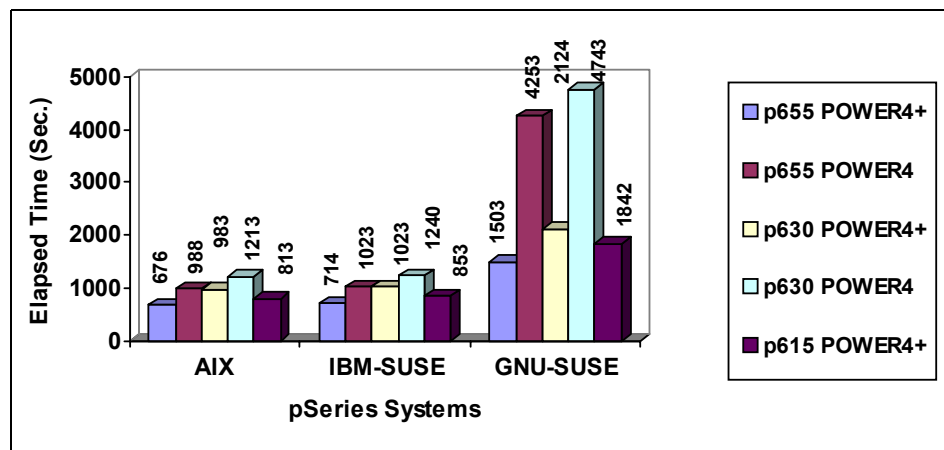


Figure 6 AMBER jac benchmark¹ on page 7

- ▶ *gb_mb*: Generalized Born myoglobin simulation. Myoglobin is a relatively simple oxygen-binding protein found in almost all mammals, primarily in muscle tissue. This protein has 2492 atoms (153 residues), and is run with a 20 Å. cutoff and a salt concentration of 0.2M, with nrespa=4 (long range forces computed every four steps).

For this particular application we see a different performance behavior for the two examples selected here. In the case of the jac benchmark, the

performance of the Linux version on both systems is within 2–6 percent of the AIX version with IBM compilers. However, the GNU compiler shows slowdowns of 122 percent, 330 percent, 116 percent, and 291 percent for p655 POWER4+, p655 POWER4, p630 POWER4+, and p630 POWER4, respectively. To improve the performance for this case with the GNU compiler additional optimization might be required. On the other hand, a comparison between AMBER compiled with IBM versus the GNU compiler for gb_mb shows slowdowns that oscillate between 7–17 percent.

It should be pointed out that the gb_mb benchmark makes extensive use of the mathematical MASS library [14]. This library is not available on Linux on POWER, yet. The large discrepancy between AIX and SUSE on all the machines using binaries built with IBM compilers may be attributed to the lack of MASS libraries [14].

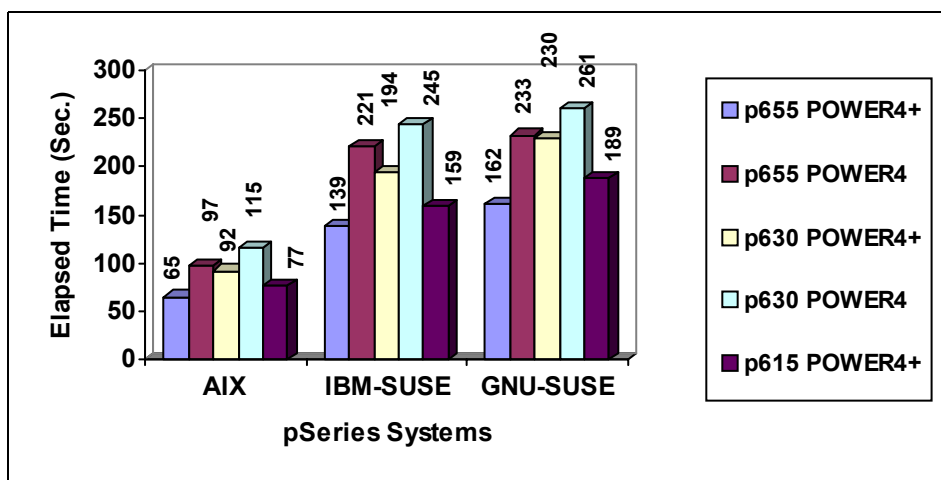


Figure 7 AMBER gb_mb benchmark¹ on page 7

One can notice a large performance difference between AIX and Linux, unlike jac benchmark above. This may be due to one of the routines egb.f in the application making extensive use of MASS libraries [14] that are currently not available on SUSE Linux, as previously mentioned.

The data also shows about 5 percent performance advantage of IBM compilers over GNU compilers on SUSE Linux.

FTDock

Fourier Transform Dock (FTDOCK) is a Fourier transform-based application to perform rigid-body docking on two biomolecules for predicting their correct binding geometry. With the number of individual protein structures growing

rapidly and the relatively small number solved complexes, predictive docking becomes an important theoretical method. It discretizes the two biomolecules on orthogonal grids and performs a global scan of translational and rotational space. FTDock outputs multiple predictions that can be screened using bio-chemical information [15]. FTDock requires a public domain FFT package, FFTW. FFTW V2.1.13 is used in the current study.

The benchmark in the current study used two protein molecules, 5pti and 2pka. Two different grid sizes, 64^3 and 128^3 , were used for performance benchmarking. For each grid size, 600 different orientations of the global scan were performed.

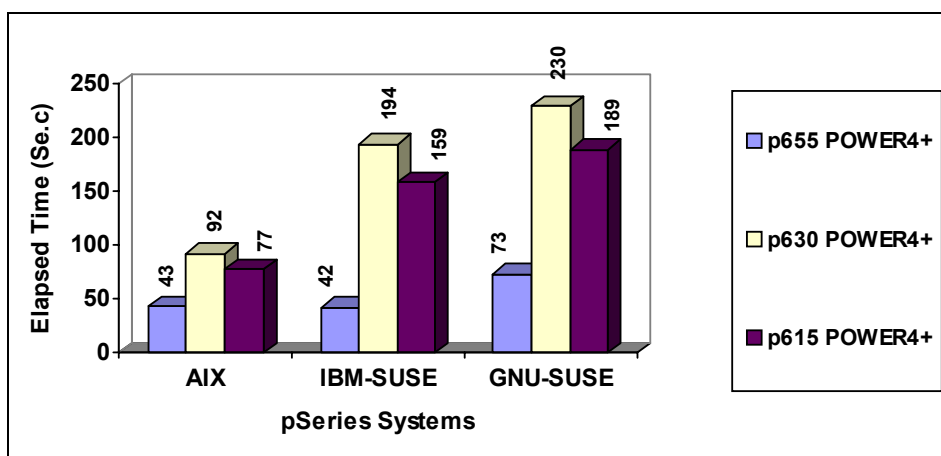


Figure 8 FTDock 5pti benchmark with a grid size of 64^3 (see footnote 1)

Figure 8 illustrates the performance of the 5pti benchmark. The trend observed here is similar to the AMBER case, that is, binaries built on AIX show a performance advantage, except on the p655 POWER4+ where the Linux on POWER version is faster by 2 percent. On the other hand, the p655 POWER4+ shows an unusually large speedup when compared to the p630 POWER4+ and p615 POWER4+. An additional set of similar benchmarks will be carried out to identify the origin of this discrepancy.

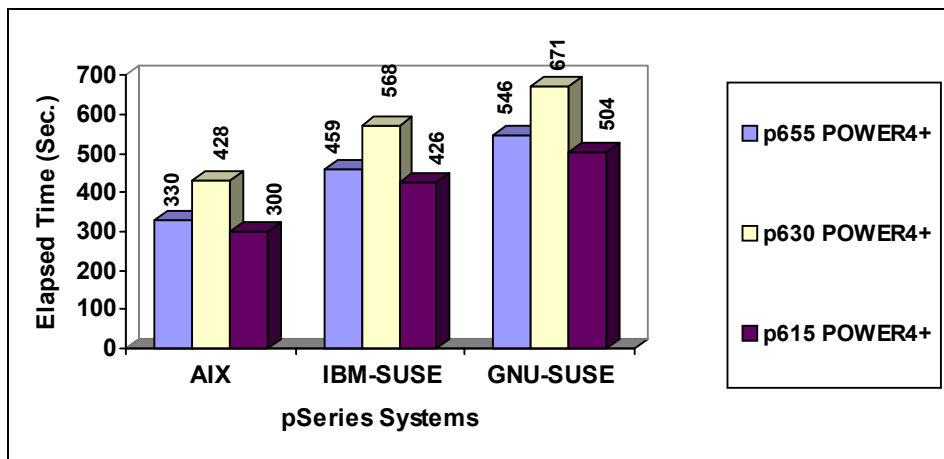


Figure 9 FTDock 2pka benchmark with grid sizes of 128^3 (see footnote 1)

Figure 9 summarizes a similar comparison as above, but in this case for the 2pka benchmark. Although the magnitudes are different, the qualitative trends in performance are similar.

AUTODOCK

AutoDock is a docking tool [16]. It is used to predict how small molecules can interact with receptors. Small molecules are normally considered to be substrates or drug candidates. In other words, a ligand outside a protein probes translations, orientations, and conformations until an optimal fit is located [16]. This package consists of three separate programs: AutoTors selects the bonds that will be part of the degrees of freedom; AutoGrid replaces the protein substrate with a precalculated grids; and AutoDock is used for docking a ligand to the target, where the target protein is described by a set of grids calculated by AutoGrid.

AutoDock can be used in areas such as X-ray crystallography, lead optimization, virtual screening, combinatorial library design, protein-protein docking, and chemical mechanism studies [16].

In this study, we ran a benchmark for HIV-1 protease using the Metropolis method, Genetic algorithm, and Modified Genetic algorithm. This particular benchmark has been identified by the AutoDock developers as having a high degree of difficulty due to many torsional angles and structural complexity [16]. All the benchmarks were performed using approximately the same number of energy evaluations. This approach is similar to the one used by the developers.

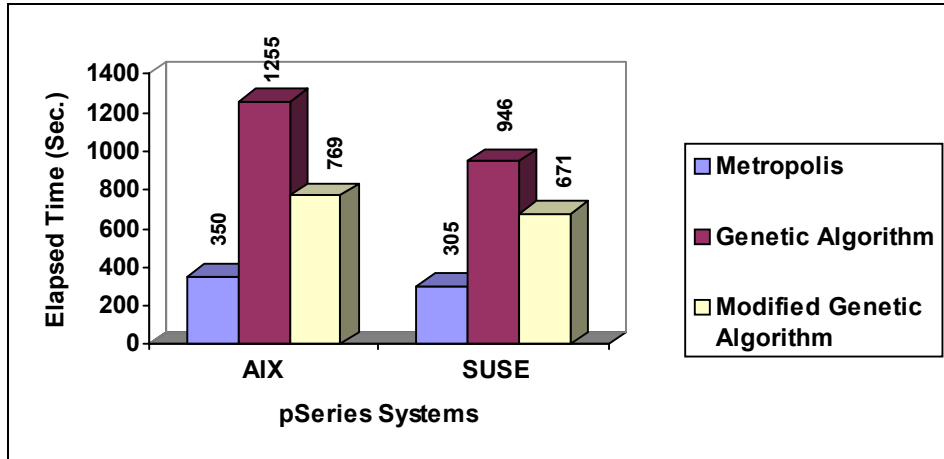


Figure 10 HIV-1 protease timings running on different pSeries systems under AIX and SUSE; only run 4 is displayed in this plot

Figure 10 shows that basically in all the cases SUSE on POWER4 tends to be faster than running on AIX. The difference ranges from about 7 to 25 percent. This set of results shows a different trend from AMBER or FTDOCK. This difference is difficult to quantify due to the nature of the algorithms, that is, random walks are performed on a particular configuration. Thus, it becomes indeterministic.

BLAST

Basic Local Alignment Search Tool (BLAST) is a set of similarity search programs designed to explore all of the available sequence databases regardless of whether the query is protein or nucleic acid [17]. The BLAST programs have been designed for speed, with a minimal sacrifice of sensitivity to distant sequence relationships. The scores assigned in a BLAST search have a well-defined statistical interpretation, making real matches easier to distinguish from random background hits. BLAST uses a heuristic algorithm that seeks local as opposed to global alignments and is therefore able to detect relationships among sequences that share only isolated regions of similarity [18].

In this study we have considered three types of benchmarks with the blastn program:

- ▶ Small single query
- ▶ Medium single query
- ▶ Large single query

The type of query is classified by the total number of characters involved in the query. A small query corresponds to about 2000 characters, medium is about 50000 characters, and large is about 200000 characters. The database used is the human genome DNA sequence [19]. This approach is consistent with a previous study [20].

The performance of BLAST benchmarks is measured on single processor. For small query, the SUSE Linux performance is almost identical to the AIX version.

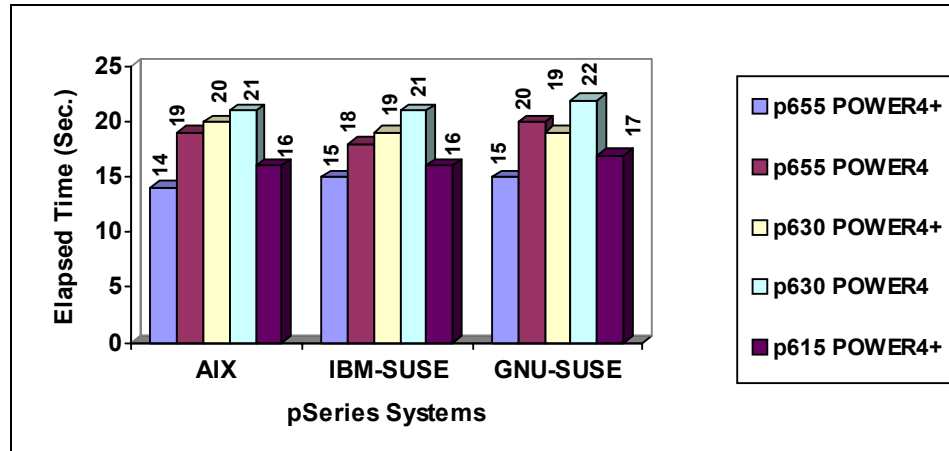


Figure 11 *blastn* small benchmark¹ on page 7

Figure 11 illustrates the performance of *blastn* using a small query and the human genome database. The only case where benchmarks run faster on AIX is on the p655 POWER4+. Although small, the executables built with the IBM compilers tend to be between 0 to 6 percent faster.

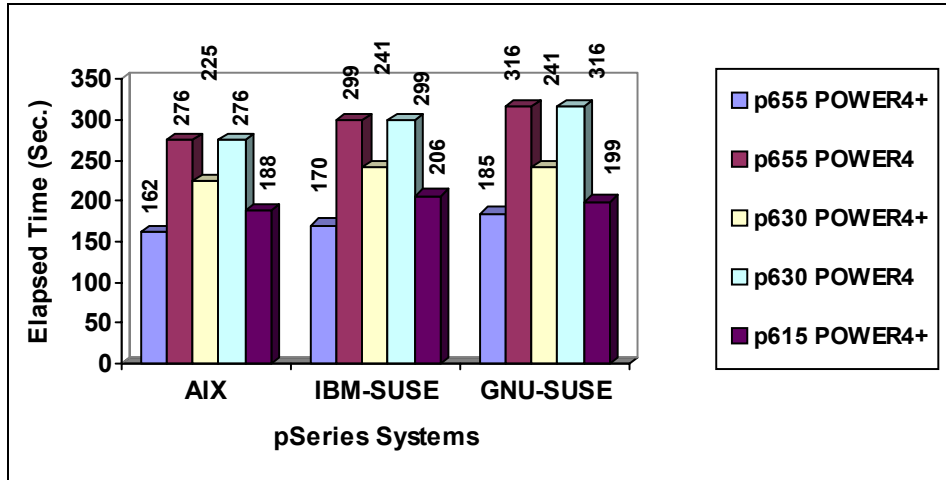


Figure 12 blastn medium query benchmark¹ on page 7

As the query gets larger we see that the AIX version of blastn is between 5 to 10 percent faster. Again, binaries built with the IBM compiler are faster than the ones built with the GNU compiler. On average they are 4 percent faster. All the results for the medium size query are summarized in Figure 12.

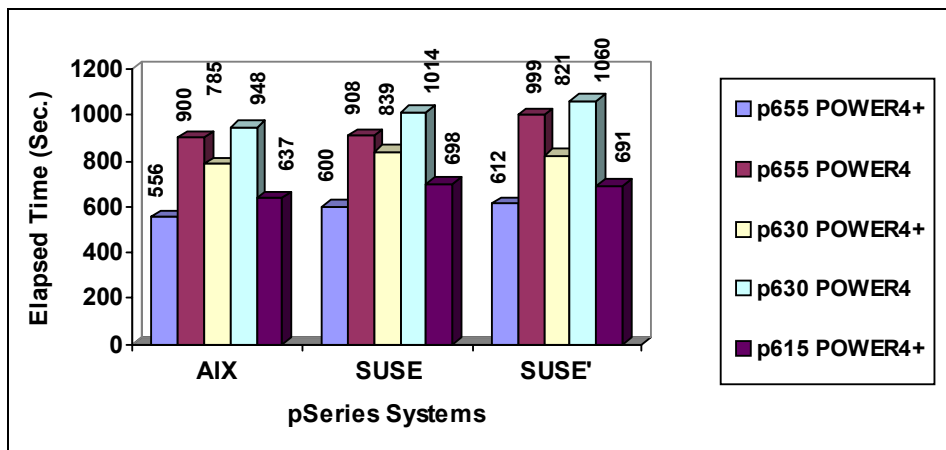


Figure 13 blastn large query benchmark¹ on page 7

Figure 13 shows performance using the large query. The results are very close to the previous case. The main difference is when comparing the performance of the IBM compilers versus the GNU compilers. In this case the average difference in performance is 1 percent.

FASTA

FASTA is a collection of sequence comparison programs for biological sequence databases. These programs are widely used in bioinformatics and can be used to search sequence databases, evaluate similarity scores, and identify periodic structures based on local sequence similarity. FASTA itself is one of the programs of the FASTA suite to compare a protein sequence to another protein sequence or to a protein database, or a DNA sequence to another DNA sequence or a DNA library [21].

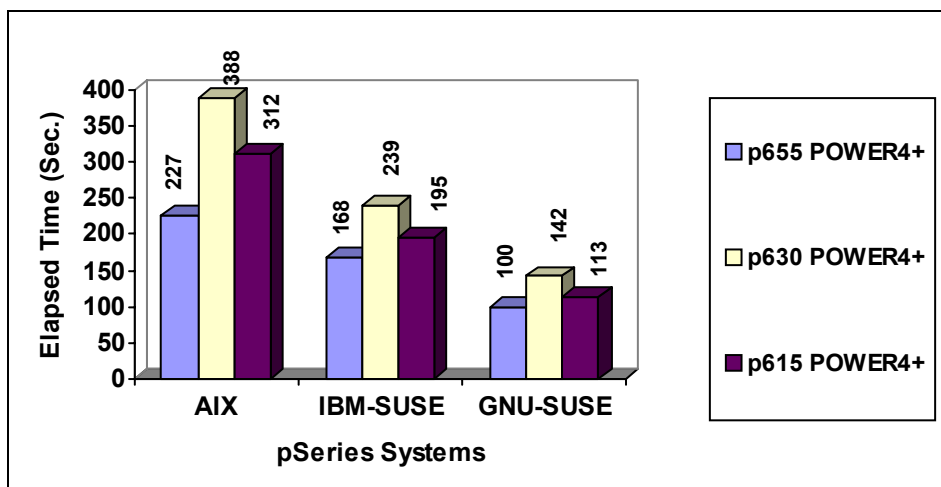


Figure 14 FASTA chr22 benchmark¹ on page 7

FASTA is an integer intensive C application. In this study we have used as a query ch22queries.seq searching the chr22.tfa library.

The trends observed for this benchmark on pSeries POWER4+ are consistent with our previous report [1]. The improved performance observed with GNU compilers is further being studied by IBM compiler developers.

HMMER

HMMER is a suite of applications for making and using Hidden Markov Models (HMM) of biological sequences. Algorithms in the HMMER package use profile hidden Markov models that can be used to do sensitive database searching using statistical description of a sequence family's consensus [21, 22].

We have used the hmmsearch module in this benchmark that employs a profile-based approach to match queries against the database. The Database SWISS-PROT Release 25 [24] is used in this study.

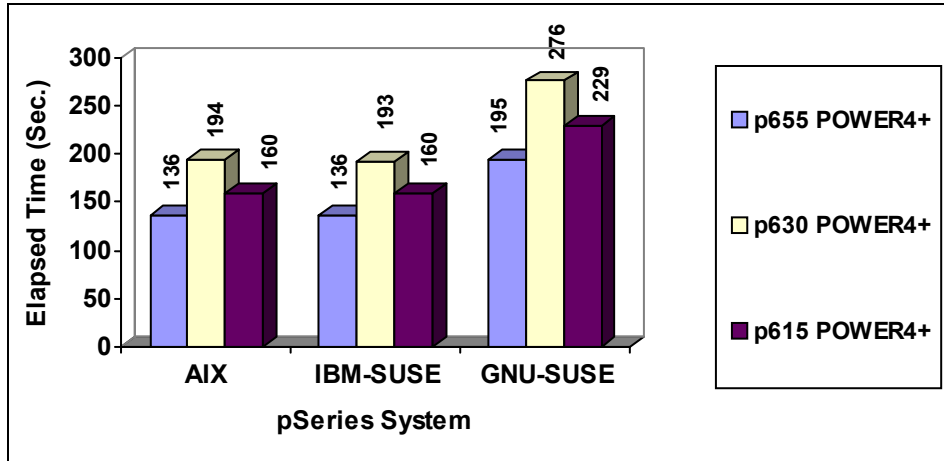


Figure 15 HMMER worm benchmark¹ on page 7

Figure 15 summarizes the HMMER benchmark performance for three different cases: AIX with IBM C/C++ compiler, SUSE Linux with IBM C/C++ compilers, and SUSE Linux with GNU gcc compilers on three different POWER4+ systems. Figure 15 illustrates that the performance is similar between AIX and Linux with IBM compilers. This is different from what we previously reported [1], that is, a large difference between AIX and SUSE. This may be attributed to more robust IBM compilers. Performance using IBM compilers for Linux on POWER is better for all the systems tested for HMMER by approximately 43 percent.

CLUSTAL W

CLUSTAL W is a widely used general purpose multiple sequence alignment program for DNA or proteins. It produces biologically meaningful multiple sequence alignments of divergent sequences, calculates the best match for the selected sequences, and lines them up so that the identities, similarities and differences can be seen [24].

CLUSTAL W is an integer intensive C application utilizing the progressive alignment, making it computationally intensive. The benchmark uses 17 HIV virus sequences for multiple sequence alignments.

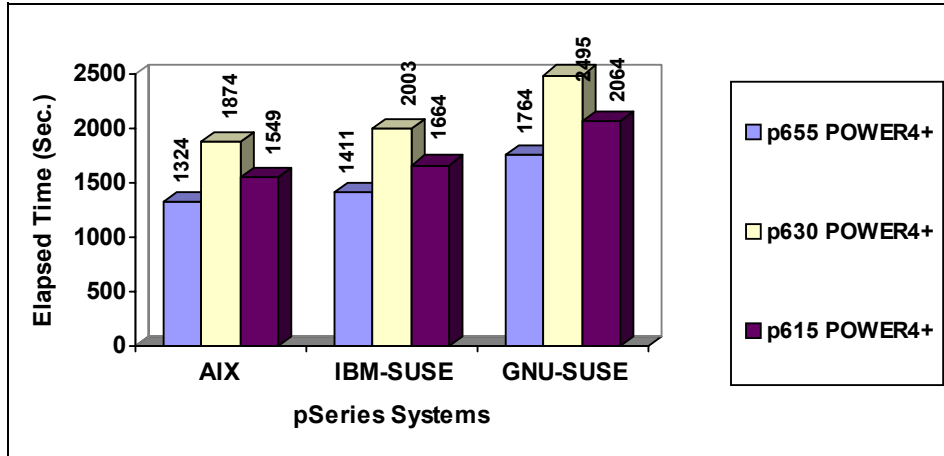


Figure 16 Clustal W 17 HIV virus sequence benchmark¹ on page 7

The performance of this benchmark on three pSeries POWER4+ systems is illustrated in Figure 16. In all the cases, a lower number indicates better performance.

It is observed that AIX is about 7 percent better over SUSE Linux with IBM compilers on the pSeries systems that were tested here.

In the case of SUSE Linux with IBM and GNU compilers, the performance is consistently better by about 25 percent with IBM.

SimWalk2

SimWalk2 is a statistical genetics computer application for haplotype, parametric linkage, non-parametric linkage (NPL), identity by descent (IBD) and mistyping analyses on any size of pedigree. SimWalk2 uses the Markov chain Monte Carlo (MCMC) method and simulated annealing algorithms to perform these multipoint analyses.

SimWalk2 requires four input files: The map, locus, and pedigree data files. The penetrance data file is necessary only for parametric linkage analysis, while the BATCH2.DAT control file contains the user-specified instruction parameters. To run this benchmark, the BATCH-33 workload for the non-parametric linkage analysis [25] was used.

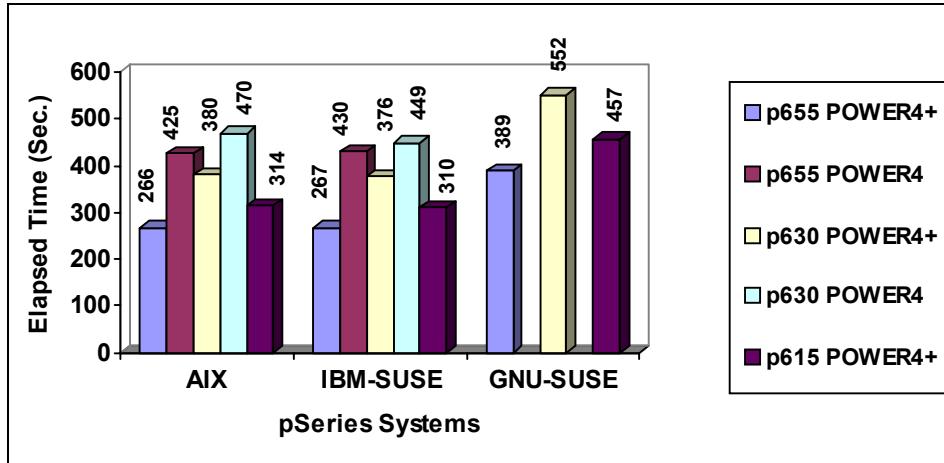


Figure 17 SimWalk2 BATCH-33 benchmark¹ on page 7

The BATCH-33 benchmark performance comparison is made in Figure 17 between AIX version and SUSE Linux version with two different pSeries POWER4 systems. The IBM XL Fortran compiler is used on both AIX and Linux. The p630s SUSE Linux version is within 5 percent faster than the AIX version, and the timings for the p655s between the two versions are almost identical.

DMO

Dip Moveout (DMO) is a seismic processing algorithm that operates on pre-stacked seismic data volumes and is basically used as a method of removing the ill effects of dipping reflectors often found beneath the earth's surface. Three Dimensional DMO (3D DMO) applies a process using three-dimensional seismic datasets. Kirchhoff Summation is one of the most commonly used methods of applying 3D DMO. Although many oil and service companies in the petroleum industry have incorporated the DMO process within prestack migration algorithms, 3D DMO is still a valuable processing tool in the seismic processing industry [27].

DMO is computationally floating point intensive Fortran code. We have used 185,000 traces for our benchmark study.

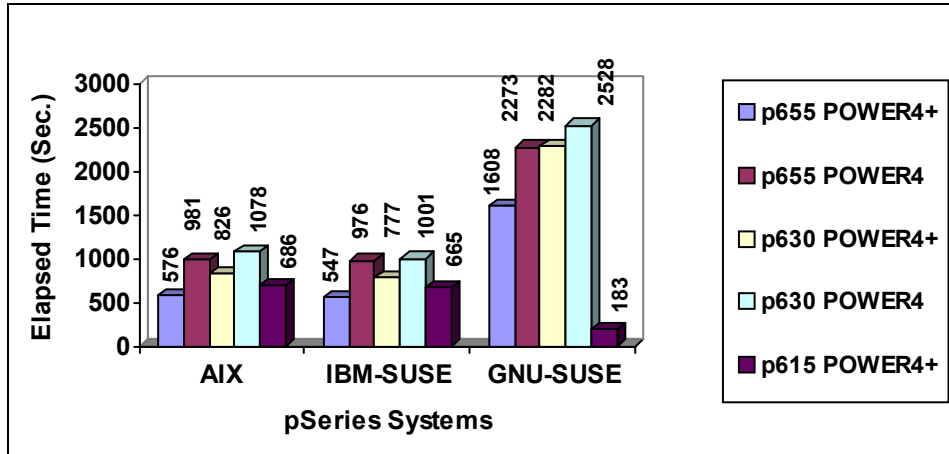


Figure 18 DMO 185000 traces benchmark¹ on page 7

The DMO benchmark performance for three different cases —AIX with IBM XL Fortran compiler, SUSE Linux with IBM XL Fortran compiler, and SUSE Linux with GNU g77 compilers on five different POWER systems—is summarized in Figure 18. It is apparent from the figure that SUSE Linux is roughly 5 percent faster than AIX on the systems tested here with IBM compilers, whereas the timings are almost identical on p655.

Also, it is observed that IBM compilers significantly outperformed GNU compilers for DMO benchmark by a factor larger than 2x, except in the case of the p615 POWER4+, where GNU tends to be faster.

MILC

The MILC code is a set of codes written in C developed by the MIMD Lattice Computation (MILC) collaboration for doing simulation of four dimensional SU(3) lattice gauge theory on MIMD parallel machines. The MILC Code is publicly available for research purposes [28].

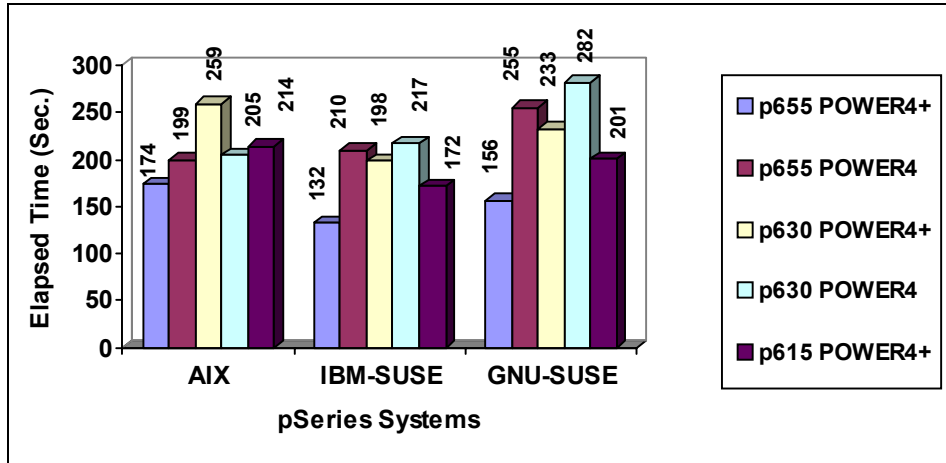


Figure 19 MISC benchmark¹ on page 7

Figure 19 depicts the performance of the MISC benchmark on two different POWER4 systems. In all the cases, smaller elapsed times indicate better performance.

Observations from this benchmark show that SUSE Linux version is within 5–6 percent of the AIX version with IBM Compilers installed on both p630 and p655 systems.

In the case of SUSE Linux with IBM and GNU compilers, IBM compilers are about 30 percent and 21 percent better than GNU compilers on p630 and p655, respectively.

Summary

In the current study, an effort is made to provide a performance comparison on IBM pSeries systems for several scientific applications running AIX 5.1 and SUSE Linux V8 utilizing IBM POWER4 systems. All the benchmarks were carried out on a single processor. Parallel performance will be published in a separate manuscript.

The overall performance (see Table 3 on page 25) between these applications running AIX and Linux on POWER is fairly similar. However, there are some pathological cases such as AMBER, which relies on the MASS libraries; currently they are not available yet on Linux on POWER. The other case is FASTA, which is currently being further analyzed.

Table 4 on page 27 shows that binaries compiled with the IBM compilers tend to outperform binaries compiled with the GNU compiler, at least for the version that we tested. As IBM compiler technology matures on Linux on POWER, we begin to see that with the exception of the pathological cases, single processor performance is becoming comparable on both systems.

Acknowledgements

Balaji Atyam and Carlos Sosa would like to thank Bruce Hurley and Jeri Hilsabeck for their encouragement and making this type of study possible. We would like to extend our gratitude to everybody that is contributing to the Linux on Power project. We also would like to thank N. Yevik for doing the initial port of AutoDock, and B. Arenburg for all his technical help and providing us with a copy of AutoDock. Thanks are also due to Billy Robinson for providing DMO source code.

References

1. B. V. Atyam, R. Panda, C. P. Sosa, Performance Comparison of Scientific Applications on AIX and Linux for PowerPC, IBM Corporation, International Technical Support Organization, RedPaper REDP-3692-00, Austin, Texas, May 2003.
2. D. Quintero, P. Attaluri, T. Baublys, X. He, C. Y. Lee, and F. Thomas, Developing Linux on IBM eServer pSeries Clusters, IBM Corporation, International Technical Support Organization, RedBook SG24-7014-00, Austin, Texas, December 2003.
3. AIX 5L™ for POWER Version 5.1 Release Notes, Release Notes Version: 1.3, <http://www-1.ibm.com/servers/aix/library/>; Additional information about SUSE may be found by visiting: http://www.SUSE.com/index_us.html.
4. D. Daines, W. Seiwald, and D. Williams *pSeries 630 Models 6C4 and 6E4 Technical Overview and Introduction*, IBM Corporation, International Technical Support Organization, June 2002.
5. H. M. Matis, J. D. McCalpin, M-C, Chiang, F. P. O'Connell, and P. Buckland *IBM pSeries 690 Configuring for Performance*, IBM Corporation, Austin, TX, 2001.
6. H. M. Mathis, J. D. McCalpin, J. Thomas, *IBM eServer pSeries 655 Basic Building Block for 21st Century High Performance Clustering*, IBM Corporation, Marketing Communication server group, November 2002, Somers, NY.

7. A. Chavan, D. Muhamedagic, J. A. Krainer, J. Co, and K. Jeong *AIX and Linux Interoperability*, IBM Corporation, International Technical Support Organization, December 2002.
8. M. W. Schmidt, K. K. Baldrige, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, and J. A. Montgomery, *J. Comput. Chem.*, 14, 1347 (1993).
9. For further information see:
<http://www.gromacs.org>
10. M. Koutek, J. van Hess, F. H. Post, A. F. Bakker, Eighth Eurographics Workshop on Virtual Environments, S Muller and W. sturg, editors, 2002.
11. For more information on the GNU compilers visit:
<http://gcc.gnu.org/>
12. David A. Case, David A. Pearlman, James W. Caldwell, Thomas E. Cheatham III, Junmei Wang, Wilson S. Ross, Carlos Simmerling, Tom Darden, Kenneth M. Merz, Robert V. Stanton, Ailan Cheng, James J. Vincent, Mike Crowley, Vickie Tsui, Holger Gohlke, Randall Radmer, Yong Duan, Jed Pitera, Irina Massova, Peter A. Kollman, AMBER7 User's Manual, University of California San Francisco, CA.
13. For more information on AMBER on IBM systems visit:
<http://www.msi.umn.edu/~cpsosa/ChemApps/MolMech/amber/amber.html>
14. Mathematical Acceleration Subsystem (MASS), MASS Version 3.0,
<http://www-106.ibm.com/developerworks/toolbox/>
15. For further information visit:
<http://www.bmm.icnet.uk/docking/>
16. G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, *J. Comp. Chem.* 19, 1639(1998).
17. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, *J. Mol. Biol.* 215, 403 (1990).
18. For further information see:
<http://www.ncbi.nlm.nih.gov/>
19. Sanger Institute Human Genome Server:
http://www.ensembl.org/Homo_Sapiens/
20. C. P. Sosa, Z. J. Tu, and P. Fast Some Practical Suggestions for Performing NCBI BLAST Benchmarks on a pSeries 690 System, IBM Corporation, February 2002
(<http://www.redbooks.ibm.com/redpapers/pdfs/redp0437.pdf>).

21. For further information see:

<http://fasta.bioch.virginia.edu>

22. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.

23. For further information see:

<http://hmmer.wustl.edu/>

24. For further information on SWISSPROT see:

<http://us.expasy.org/sprot/>

25. J. D. Thompson, D. G. Higgins, and T. J. Gibson, Nucleic Acids Res., 22, 4673 (1994).

26. For further information see:

<http://watson.hgen.pitt.edu/docs/simwalk2.html>

27. B. Robinson, private communication.

28. For further information see:

<http://www.physics.utah.edu/~detar/milc/>

Appendix

Table 2 Results for multiple AutoDock benchmark runs

Run	AIX (elapsed time in seconds)	SUSE (elapsed time in seconds)
Metropolis Method		
1	405	366
2	300	322
3	366	308
4	350	305
Genetic Algorithm		
1	1346	955
2	1257	940
3	1259	948
4	1255	946

Run	AIX (elapsed time in seconds)	SUSE (elapsed time in seconds)
Modified Genetic Algorithm		
1	780	670
2	768	665
3	772	662
4	769	671

Table 3 Summary of benchmarks (AIX versus Linux)^a on page 26

Application	Area	Language	Benchmark	AIX advantage over Linux (%) ^a
GAMESS	Life Sciences	Fortran and C	Adamantane	p655 - -8 p615 - -9 p630 - -9
			pentenal	p655 - -4 p615 - 0 p630 - -5
GROMACS	Life Sciences	Fortran and C	dppc	p655 - +10 p615 - +11 p630 - +10
			villin	p655 - +2 p615 - +1 p630 - +0
			poly-ch2	p655 - +37 p615 - +36 p630 - +35
AMBER	Life Sciences	Fortran	jac	p655 - +6 p615 - +5 p630 - +4
			gb_mb	p655 - +114 p615 - +106 p630 - +111
FTDOCK	Life Sciences	C	64 ³ size	p655 - -2 p615 - -4 p630 - -2

Application	Area	Language	Benchmark	AIX advantage over Linux (%) ^a
			128 ³ size	p655 - -9 p615 - -7 p630 - -8
BLAST	Life Sciences	C	Small query	p655 - 0 p615 - 0 p630 - 0
			Medium query	p655 - +5 p615 - +10 p630 - +7
			Large query	p655 - +8 p615 - +10 p630 - +7
FASTA	Life Sciences	C	chr22	p655 - -26 p615 - -38 p630 - -38
HMMER	Life Sciences	C	worm	p655 - 0 p615 - 0 p630 - -1
CLUSTAL W	Life Sciences	C	HIV Virus	p655 - +7 p615 - +7 p630 - +7
SimWalk2	Life Sciences	Fortran	BATCH-33	p655 - 0 p615 - -1 p630 - -1
DMO	Oil and Gas	Fortran	185,000 Traces	p655 - -5 p615 - -3 p630 - -6
MILC	Particle Physics	Fortran		p655 - -24 p615 - -20 p630 - -24
AutoDock	Life Sciences	Fortran and C		p630 - -19
Overall				p655 - +2 p615 - +1 p630 - 1

a. AIX performance advantage over Linux (%) = $(T_L - T_A) / T_A \times 100$, where T_L = Total Time on Linux system and T_A = Total Time on AIX system.

Table 4 Summary of benchmarks (IBM versus GNU compilers comparison on Linux)

Application	Area	Language	Benchmark	IBM advantage over GNU compilers on Linux (%) ^a
GROMACS	Life Sciences	Fortran and C	dppc	p655 - +33 p615 - +29 p630 - +31
			villin	p655 - +56 p615 - +57 p630 - +57
			poly-ch2	p655 - +31 p615 - +32 p630 - +32
AMBER	Life Sciences	Fortran	jac	p655 - +110 p615 - +116 p630 - +108
			gb_mb	p655 - +16 p615 - +19 p630 - +19
BLAST	Life Sciences	C	Small query	p655 - 0 p615 - 0 p630 - 0
			Medium query	p655 - +9 p615 - -3 p630 - 0
			Large query	p655 - +2 p615 - -1 p630 - -2
FASTA	Life Sciences	C	chr22	p655 - -40 p615 - -42 p630 - -41
HMMER	Life Sciences	C	worm	p655 - +43 p615 - +43 p630 - +43
CLUSTAL W	Life Sciences	C	HIV Virus	p655 - +25 p615 - +24 p630 - +25

Application	Area	Language	Benchmark	IBM advantage over GNU compilers on Linux (%) ^a
DO	Oil and Gas	Fortran	185000 Traces	p655 - +194 p615 - +183 p630 - +194
MILC	QCD	Fortran		p655 - +18 p615 - +17 p630 - +18
FTDOCK	Life Sciences	C	128 ³ size	p655 - +43 p615 - +33 p630 - +33
SimWalk2	Life Sciences	Fortran	BATCH-33	p655 - +46 p615 - +47 p630 - +47
Overall				p655 - +33 p615 - +32 p630 - +32

a. IBM advantage over GNU compilers on Linux system (%) = $(T_G - T_I) / T_I \times 100$, where T_G = Total Time with GNU compilers and T_I = Total Time with IBM compilers.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 email to:
redbook@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 905, Internal Mail Drop 9053D005, 11501 Burnet Road
Austin, Texas 78758-3493 U.S.A.

Trademarks

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

AIX 5L™
AIX®
@server®
@server®
IBM®

ibm.com®
PowerPC®
POWER™
POWER4™
pSeries®

Redbooks™
Redbooks (logo) ™
VisualAge®

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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