



Amanda Peters
Marcus E. Lundberg
P. Therese Lang
Carlos P. Sosa

High Throughput Computing Validation for Drug Discovery Using the DOCK Program on a Massively Parallel System

This IBM® Redpaper publication presents a virtual screening study of the DOCK Version 6.0 molecular docking software package on a massively parallel system, the IBM System Blue Gene® supercomputer, Blue Gene/L.¹ Virtual screening of very large libraries of small ligands requires not only efficient algorithms but an efficient implementation for docking thousands, if not millions, of compounds simultaneously in a reasonable amount of time.

This paper presents a series of receptor-ligand docking benchmarks using DOCK Version 6.0 to show performance improvements with better load balancing and file I/O improvements. In addition, a version was implemented to take advantage of the High Throughput Computing (HTC) feature on the Blue Gene supercomputer.

Introduction

Parallel processing long has been recognized as a powerful tool in computational chemistry and biology.^{2, 3, 4} Software developers in the life sciences have realized that parallel computing benefits come from both hardware that can scale to a large number of processors and software that can map to these types of architectures.

The advantages of massively parallel systems for certain scientific and engineering applications include access to a large amount of aggregated real memory and a large collection of processors. Computer simulations play an important role in scientific investigations, in particular, by providing insight into molecular interactions that cannot be obtained experimentally. The ability to inform new experiments is certainly the case in virtual screening, or *in silico* screening. This approach, in which computational algorithms are used to cull large libraries of potential drugs into more practical numbers for experiments, has attracted the attention of researchers working in pharmaceutical companies as an important application in the arsenal of tools used in the drug discovery process.⁵

In silico screening using molecular docking has been recognized as an approach that benefits from high-performance computing to identify novel small molecules that can then be used for drug design.⁶ This process consists of the identification or selection of compounds that show activity against a biomolecule that is of interest as a drug target.⁷ Docking programs place molecules into the active site of the receptor (or target biomolecule) in a non-covalent fashion and then rank them by the ability of the small molecules to interact with the receptor.⁸ There is an extensive family of molecular docking software packages.^{9–14} However, in this work, we carry out our studies with the DOCK (Version 6.0) package.

In this study, we discuss our efforts in optimizing and validating DOCK on a massively parallel system, the IBM Blue Gene/L solution. In the article “Development and Validation of a Modular, Extensible Docking Program: DOCK5” of the *Journal of Computational Aided Molecular Design*, the authors point out that their primary criterion for optimization of the DOCK package in their work was success in identifying the correct geometry and not looking at improving processor performance. In this study, we complement their work. Our approach relies on optimizing single node performance after carrying out a performance profile to identify bottlenecks in different sections of the code. We also implement better load balancing, I/O optimization, and a high throughput computing scheme to carry out *in silico* screening with thousands of processors.

Software and hardware overview

We describe the IBM Blue Gene/L platform in detail in *Unfolding the IBM eServer Blue Gene Solution*, SG24-6686. Here, we summarize the key IBM Blue Gene/L™ architectural features that are relevant for this study. The smallest component is the chip (node). The Blue Gene/L basic block is a PowerPC® 440 dual-core processor. Each processor core (processor) runs at a frequency of 700 MHz and can perform four floating-point operations per cycle, giving a theoretical peak performance of 5.6 GFlops per chip.¹⁵ A rack holds 1,024 compute nodes. In this work, we used eight racks for a total of 16,384 processors.

DOCK is an open-source molecular docking software package that is frequently used in structure-based drug design.¹⁶ The computational aspects of this program can be divided into two parts. The first part consists of the ligand atoms that are located inside the cavity or binding pocket of a receptor, which is a large biomolecule. This step is carried out by a search

algorithm.¹⁷ The second part corresponds to the scoring or identifying interactions. This is normally done by means of a scoring function.¹⁸

DOCK Version 6.0, which was used in this study, is written in C++ to exploit code modularity and has been parallelized using the message passing interface (MPI) paradigm.^{19, 20} DOCK Version 6.0 is parallelized using a master-worker scheme.²¹ The master handles I/O and tasks management while each worker is given an individual molecule to carry out simultaneous independent docking.²²

Massively parallel version

The implementation of the massively parallel version of the code can be divided into two separate efforts. The first part corresponds to single node optimization. The second part involves code parallelization, in this case making use of the fact that DOCK carries out each calculation per molecule semi-independently, as implemented by the developers via MPI.

Single processor optimization

Our initial profile analysis showed that, aside from the master node, little time is spent doing communication. Since our profile did not show large amounts of I/O when docking a single ligand, this indicated that the code is CPU bound. Most of the CPU time is spent in the scoring function.

The level of optimization performed on a single processor involved compiler flags optimization and the use of highly optimized libraries. The compiler options are described in *Unfolding the IBM eServer Blue Gene Solution*, SG24-6686, and relate to performance optimization of applications for Blue Gene/L platform.

We found that the combination of `-qgnprag=omp`, `-qarch=440d`, `-qtune=440`, `-O3`, and `-qhot` produced the best overall performance for the DOCK code. Algorithm enhancements and code changes were employed to ensure the utilization of IBM Mathematical Acceleration Subsystem (MASS) libraries.²³ A 43% improvement in speed of calculation for a single ligand was observed for this single processor optimization.

Parallel implementation

DOCK uses a master-worker scheme to parallelize the molecular docking steps during the simulation. Currently, data is distributed to the workers randomly as molecules are read by the master. The drawback of this approach is a poor load balance as the number of processors is increased. This random distribution resulted in some workers receiving many short running tasks preceding a long running task. As a result, workers with short running tasks were idle while other workers completed the longer, time consuming tasks.

However, by sorting the input file to distribute tasks with ligands with the greatest number of rotatable bonds and therefore greater number of potential docking conformations first, we were able to improve scalability. We also sorted with respect to the number of atoms per ligand. In addition, we reduced the amount of file I/O, by storing temporary files in memory, which had a large impact on the scalability.

Finally, in looking to scale to large systems with over eight thousand processors, it became necessary to determine the best paradigm to increase scalability. The master-worker scheme has been shown to not scale well with a large number of nodes due to overloading of the master.²⁴ With previous applications, this problem was overcome by employing a multi-level

master-worker scheme.²⁵ However, with the DOCK code, we chose to use a new mode available on the Blue Gene/L platform for HTC that uses a dispatcher to asynchronously send work units to individual compute nodes to be executed.²⁶

The DOCK program can be run as a collection of independent tasks in which each processor conducts independent calculations on different ligands in the library. This allowed us to implement an HTC version of the DOCK program. The HTC version consists of individual instances of the non-MPI version running on subsets of the ligand library simultaneously. Both output parsing and the work dispatcher run on the Blue Gene/L front end rather than a traditional master. An additional advantage of this scheme is increased robustness if a node fails. The dispatcher simply resets the node and redistributes the work task, which preserves the work on all the other nodes. We chose this method both for the scalability it offers along with the node resiliency.

Performance results

In this section, we discuss the performance results that we observed.

Data sets

The receptor corresponds to the Human Immunodeficiency Virus-1 (HIV-1) reverse transcriptase in complex with nevirapine as used and described in the article “Development and Validation of a Modular, Extensible Docking Program: DOCK5” of the *Journal of Computational Aided Molecular Design*. The ligand library corresponds to a subset of 27,005 drug-like ligands from the ZINC database.²⁷ Also, to assess the scalability of our parallel version of the code, we constructed a set of ligands with 128,000 copies of nevirapine as recommended in the same DOCK5 article to remove dependence on the order and size of the compound. This set was used to look at the I/O performance using a Network File System (NFS) and a General Parallel File System™ (GPFS™).²⁸

Results

The evaluation of load balancing of the massively parallel version of the DOCK software package was carried out by using a data set consisting of 27,005 ligands. Figure 1 on page 5 illustrates the performance improvement in load balancing optimization for 27,005 ligands on 2,048 processors. By sorting the molecules in terms of the number of atoms or rotatable bonds prior to being dispatched to the workers, we show a performance improvement by using the MPI version.

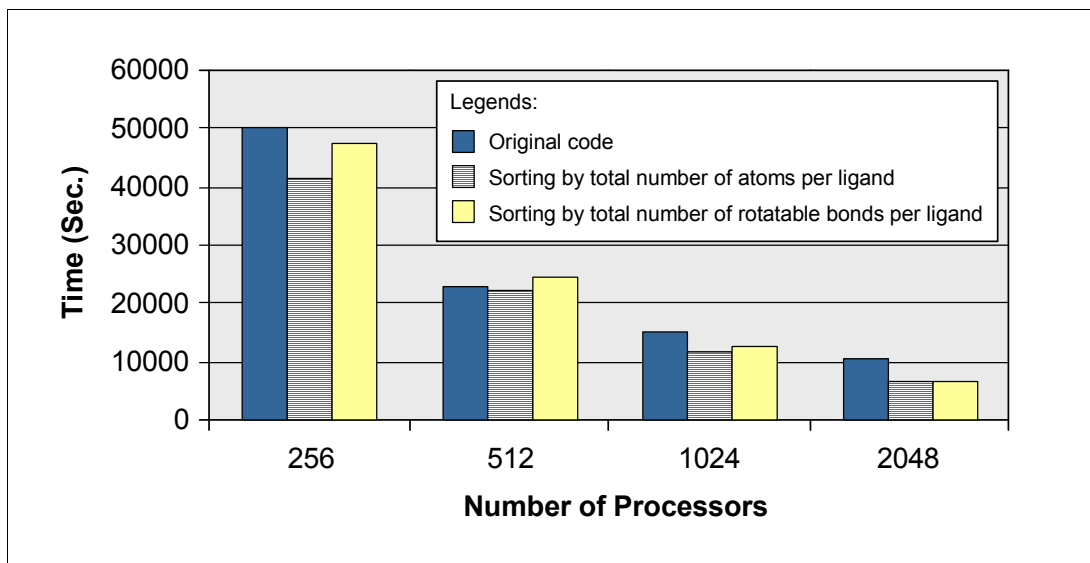


Figure 1 Effect of load balancing optimization

The optimized version as a function of rotatable bonds shows an increase in efficiency from 60% using the original code to 92% on the 2,048 processors.

We also analyzed I/O performance by testing NFS versus GPFS. Figure 2 summarizes the results when running 128,000 copies of nevaripine. As the number of processors is increased, the overall performance of the GPFS is considerably better.

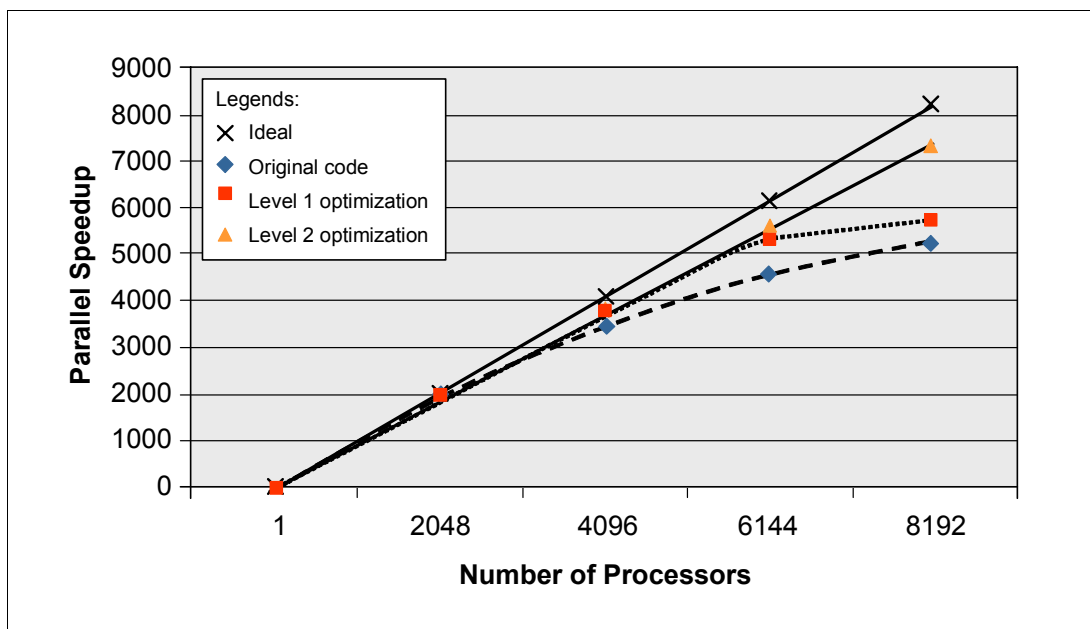


Figure 2 Performance comparison between file I/O optimizations

The original port is shown by the dotted line. The level 1 of optimization shows the improvement gained by compiler options using the NFS. The level 2 of optimization shows the improvement gained by using the GPFS.

Figure 3 illustrates parallel efficiency. It shows a comparison between the MPI version and the HTC version. In both cases, the scalability is nearly linear up to 8,192 processors. The efficiency for the MPI version and HTC version are 88% and 98%, respectively. When the number of processors was greater than 8,192, the MPI version started to suffer due to saturation of the master node. HTC runs are independent, circumventing problems associated with master-worker systems and inter-node communication.

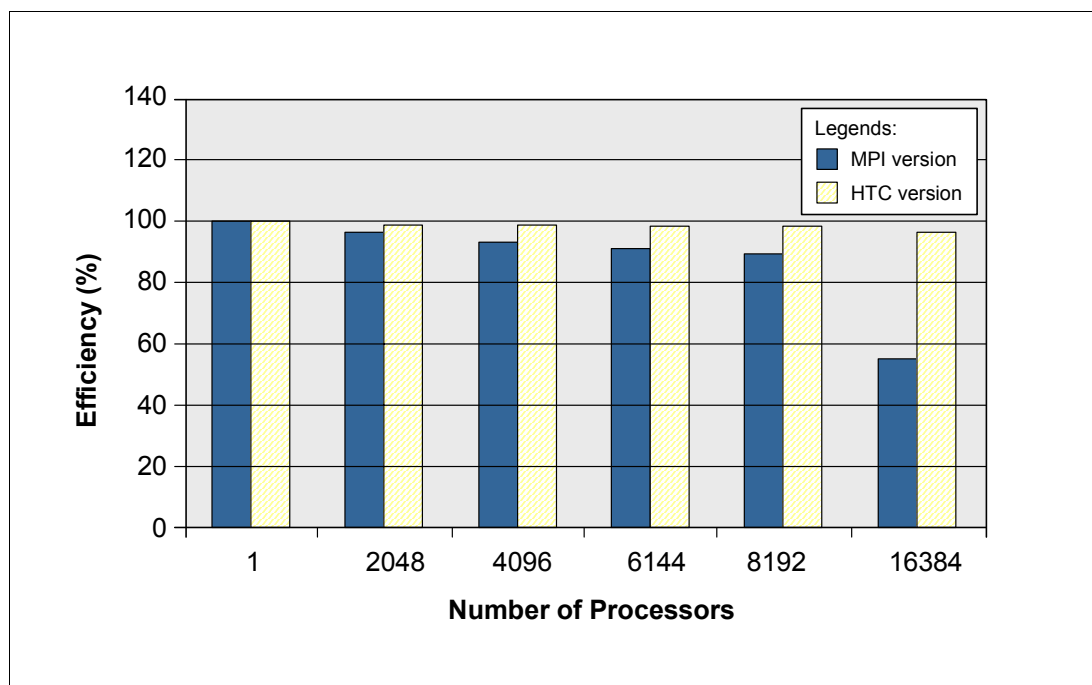


Figure 3 Parallel efficiency of the HTC and MPI versions

Discussion

The work presented in the article “Development and Validation of a Modular, Extensible Docking Program: DOCK5,” of the *Journal of Computational Aided Molecular Design*, illustrated the performance of the MPI implementation of the DOCK software package version 5 with a library of 500 and 1000 different ligands or small molecules. In addition, they used a set with 1000 copies of nevaripine to test scalability as previously described. In the study, they showed that *in silico* screening can benefit from parallel processing. They also hypothesized that the speedups for the heterogeneous library will continue to improve as a function of rotational bonds, but it would be difficult to fully remove the overhead coming from I/O and communication.

In our study, our objective was to validate the DOCK software package on a massively parallel system. Since docking programs are extensively used in drug discovery, we explored how to optimize DOCK on the Blue Gene/L platform to provide a high throughput tool for drug discovery.

Our first step involved complementing the work carried out in the article “Development and Validation of a Modular, Extensible Docking Program: DOCK5” from the *Journal of Computational Aided Molecular Design*. In Figure 1 on page 5, we showed that the load balance can be improved by sorting molecules before they are distributed to all the workers. This showed an improvement in performance from 60% by using the original code to 92% by using the optimized version.

To address the file I/O problem as the number of nodes increases, we eliminated the use of temporary files by storing them in memory. We also identified GPFS as an improvement over NFS. Figure 2 on page 5 illustrated that, for this particular case, when using more than four racks, GPFS becomes more efficient to handle file I/O.

Finally, we looked at the overhead mentioned in the article “Development and Validation of a Modular, Extensible Docking Program: DOCK5” of the *Journal of Computational Aided Molecular Design*. Clearly, as the number of nodes increased, both MPI communication and file I/O became a problem. After reaching 8,192 nodes or more, the performance of the MPI version decreased. To try to improve this, we implemented an HTC version. Figure 3 on page 6 showed that by using the HTC, for the case test here, most of the overhead is eliminated.

Conclusions

In this paper, we proposed a series of modifications to increase scalability of the popular DOCK (Version 6.0) software package on a massively parallel system, Blue Gene. In this work, we have shown that our new implementation shows good scalability and can be used to screen thousands of small molecules to identify lead compounds in drug discovery. We report several levels of optimization, starting with compiling with optimized compiler flags, storing files in memory, the introduction of a high throughput computing version, and better molecule distribution for load balancing. With this new implementation, we demonstrated that for the cases tested here, DOCK Version 6.0 is well suited for a system with thousands of distributed compute nodes to carry out in silico screening. The scalability is nearly linear for all the hardware configurations tested in this study.

The team that wrote this paper

This paper was produced by the following team of specialists:

P. Therese Lang is a postdoctoral scholar in the Alber Lab at the University of California, Berkeley. She recently received her Ph.D. degree from the Irwin D. Kuntz and Thomas L. James labs at the University of California, San Francisco, for assisting in developing DOCK Version 6 and exploring drug design for RNA targets. She has published several papers on the validation and application of docking programs for drug design.

Marcus E. Lundberg is currently a student in Scientific Computing at Uppsala University in Sweden. Marcus has a Bachelor of Science (BS) degree in Physics.

Amanda Peters is an IBM Software Engineer for the Blue Gene supercomputers, in Rochester, Minnesota. She received a BS degree in both computer science and physics from Duke University in 2005. She is involved with the porting, validating, and optimizing of life science applications.

Carlos P. Sosa is a Senior Technical Staff Member in the Blue Gene Development Group of IBM, where he has been the team lead of the Chemistry and Life Sciences high-performance effort since 2006. For the past 18 years, he has focused on scientific applications with emphasis in life sciences, parallel programming, benchmarking, and performance tuning. His areas of interest are future IBM POWER™ architectures, Blue Gene, Cell Broadband, and cellular molecular biology. He has authored or coauthored multiple papers and was a coauthor of two IBM Redbooks® publications. He received a Ph.D. degree in Physical Chemistry from Wayne State University and completed his post-doctoral work at the Pacific Northwest National Laboratory. He is a member of the IEEE, the IEEE Computer Society, the American Chemical Society, and the International Society for Computational Biology.

Thanks to the following people for their contributions to this project:

Dr. J. J. Irwin

Dr. A. Pugliese

Carl Obert

Members of the IBM On Demand Center in Rochester, MN

Deep Computing Institute at IBM Watson

LindaMay Patterson of the ITSO, IBM Rochester

References

1. (a) <http://dock.compbio.ucsf.edu> (b) Moustakas, D. T., et al. "Development and Validation of a Modular, Extensible Docking Program: DOCK5." *Journal of Computational Aided Molecular Design*. 20, pages 601-609 (2006).
2. C. P. Sosa, J. Ochterski, J. Carpenter, and M. J. Frisch. "Ab Initio Quantum Chemistry on the Cray T3E Massively Parallel Supercomputer: II." *Journal of Computational Chemistry*. 26, pages 1053-1063, 1998.
3. Thorsen, O., et al. "Parallel genomic sequence-search on a massively parallel system." *Conference On Computing Frontiers: Proceedings of the 4th international conference on Computing frontiers*. ACM, 2007, pages 59-68.
4. *Unfolding the IBM eServer Blue Gene Solution*, SG24-6686
5. S. Kraljevic, P. J. Stambrook, and K. Pavelic. "Accelerating Drug Discovery." *EMBO Reports*. Vol. 5, 2004, pages 837-842.
6. B. Waszkowycz, T. D. Perkins, R. A. Sykes, and J. Li. "Large-scale Virtual Screening for Discovering Leads in the Postgenomic Era." *IBM Systems Journal*, Vol. 40, 2001, pages 360-376.
7. G. L. Patrick. "An Introduction to Medicinal Chemistry, 3rd Edition." Oxford University Press, Oxford, UK, 2005.
8. M. Kontoyianni, L. M. McClellan, and G. S. Sokol. "Evaluation of Docking Performance: Comparative Data on Docking Algorithms." *Journal of Medical Chemistry*. Vol. 47, 2004 pages 558-565.
9. I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langidge, and T. E. Ferrin. "A Geometric Approach to Macromolecule-ligand Interactions." *Journal of Molecular Biology*. Vol. 161, 1982, pages 269-288.
10. G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson. "Automated Docking Using a Lamarckian Genetic Algorithm and Empirical Binding Free Energy Function." *Journal of Computational Chemistry*. Vol. 19: 1998, pages 1639-1662.
11. G. Jones, P. Willett, R. C. Glenn, A. R. Leach, and R. Taylor. "Development and Validation of a Genetic Algorithm to Flexible Docking." *Journal of Molecular Biology*. Vol. 267, 1997, pages 904-911.
12. M. Rarey, B. Kramer, T. Lengauer, and G. A. Klebe. "A Fast Flexible Docking Method Using an Incremental Construction Algorithm." *Journal of Molecular Biology*. Vol. 261, 1996, pages 470-489.
13. Schrödinger, Portland, OR 972001.
14. Y. P. Pang, E. Perola, K. Xu, F. G. Prendergast. "EUDOC: A Computer Program for Identification of Drug Interaction Sites in Macromolecules and Drug Leads from Chemical Databases." *Journal of Computational Chemistry*. Vol. 22, 2001, pages 1750-1771.
15. See note 4.
16. See note 1 and note 9.
17. See note 1.
18. Ibid.

19. P. T. Lang, D.M. Moustakas, S. B. Brozell, N. Carrascal, S. Mukherjee, S. Pegg, K. Raha, D. Shivakumar, R. Rizzo, D. Case, B. Shoichet, I. D. Kuntz, "DOCK 6 Users Manual" at the following Web address:

<http://dock.compbio.ucsf.edu>

A. P. Graves, D.M. Shivakumar, S. E. Boyce, M.P. Jacobson, D. A. Case, B. Shoichet. "Rescoring docking hit lists for model cavity sites: predictions and experimental testing." *Journal of Molecular Biology*. Vol. 337, 2008, pages 914-934.

P. T. Lang, S. R. Brozell, S. Mukherjee, E. Pettersen, E. Meng, V. Thomas, R. Rizzo, D. A. Case, T. L. James, and I. D. Kuntz. "DOCK 6: Combining Techniques to Model RNA-Small Molecule Complexes."

20. The MPI Forum. The message-passing interface (MPI) standard. May 1995

<http://www.mcs.anl.gov/mpi/standard.html>

21. See note 1 on page 9.

22. Ibid.

23. MASS Libraries for Blue Gene/L, IBM

<http://www.ibm.com/software/awdtools/mass/bg1/>

24. See note 3 on page 9.

25. Ibid.

26. *IBM System Blue Gene Solution: Application Development*, SG24-7179

27. J. J. Irwin and Shoichet, B. K. "ZINC - A Free Database of Commercially Available Compounds for Virtual Screening." *Journal of Chemical Information and Modeling*. 45, 177 (2005) pages 177-182.

28. H. Yu, R. K. Sahoo, C. Howson, G. Almasi, J. G. Castanos, M. Gupta, J. E. Moreira, J. J. Parker, T. E. Engelsiepen, R. B. Ross, R. Thakur, R. Latham, W. D. Gropp. "High Performance File I/O for the Blue Gene/L Supercomputer." *The Twelfth International Symposium on High-Performance Computer Architecture*, 2006, pages 187-196.

<http://csdl2.computer.org/persagen/DLabsToc.jsp?resourcePath=/dl/proceedings/&oc=comp/proceedings/hpca/2006/9368/00/9368toc.xml>

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 illustrate 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.

This document REDP-4410-00 was created or updated on April 16, 2008.



Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:
ibm.com/redbooks
- ▶ Send your comments in an email to:
redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD 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:

Blue Gene/L™
Blue Gene®
eServer™

General Parallel File System™
GPFS™
IBM®

PowerPC®
Redbooks (logo) ®
Redbooks®

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