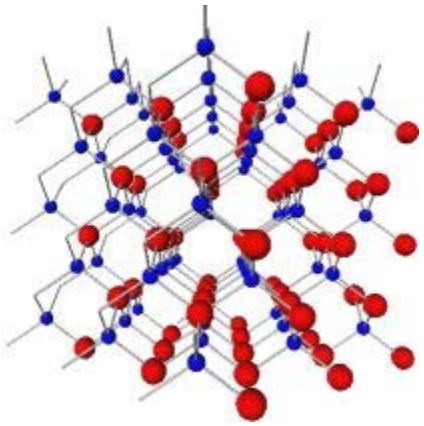
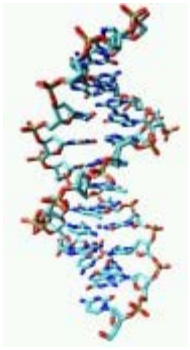
# Scalable Informatics

Computational Chemistry Performance Analysis on various current architectures.

Joseph Landman, Ph.D.
Scalable Informatics LLC
http://www.scalableinformatics.com
landman@scalableinformatics.com





## Purpose and Scope

This paper was prepared to document benchmark performance for various computational chemistry applications running on popular computing platforms. (Tests were run in December, 2004 on publicly released products at that time, with additional confirmation tests in January 2005.) The study was limited to molecular dynamics and electronic structure as computational chemistry sub-topics of interest. The applications and computing platforms are representative of a broader class of tools available to researchers within the field of computational chemistry. The report provides basic data that readers will find useful for platform evaluations. As noted, users may run variations on these tests to accommodate the unique aspects of their applications or research requirements. Results presented in this paper will help validate user specific testing and benchmarking.

It is assumed that the audience has an understanding of subtopics and applications within computational chemistry. There is also the assumption that the audience has a general knowledge of computing platform alternatives. A detailed understanding of microprocessor or system architecture is not required to use this paper.

Appendices include a price/performance analysis based on 2/28/2005 system pricing and a summary of compiler options used.

## Background

Computational chemistry is a resource intensive research endeavor, with several major foci. Examples include but are not limited to molecular mechanics and dynamics, electronic structure, biochemistry, bulk property simulation, and reaction simulation. System requirements for performing computational experiments vary depending on the program and research application of interest to the researcher. Each program uses algorithms that have particular characteristics. These characteristics combine to place a burden on the hardware platform, the software OS and middleware layers. The effect of this burden varies by application or set of applications. Hence the only performance measurements that are relevant use actual applications and input data sets.

This study was limited to molecular dynamics and electronic structure as sub-topics of interest in computational chemistry. These applications have significantly different resource requirement profiles. Molecular dynamics codes such as Amber, Charmm, NAMD, and related codes typically have "reasonable" memory requirements, and often can split work among many processors using variants on message passing interfaces. With each additional processor, the memory requirements per processor for a particular job decrease, though there are asymptotic limits to this decrease. These calculations are often bound more by the speed of processing non-bonded interactions. As the number of atoms doubles, execution time increases by a factor of 4 or more. As each data structure consumes a relatively small amount of memory, the most important set of resources for these calculations are typically the speed of the processor's floating point unit, and the memory latency, as the algorithms typically traverse neighbor lists. Neighbor list generation is computationally relatively expensive, so it is often performed every $n^{th}$ iteration, where n is typically five or larger.

Electronic structure codes are concerned with more quantum mechanical features, and often rely upon pre-computed integral tables. These tables are often quite large, and for complex analyses such as coupled cluster CCSDT, there is a requirement for multiple passes through these tables. These codes typically make extensive use of IO in terms of large block sequential reads of binary data files. Some of these codes attempt to bypass the OS memory management. These codes typically require a very high performance IO system as a result of their designs. Also, linear algebra operations, typically matrix-vector and matrix-matrix operations are used extensively. Some of these codes will use BLAS libraries and variants. These libraries are often exceptionally well tuned to the underlying platforms, such as the AMD ACML, the Intel MKL, and the Apple libraries. End users will often notice a significant difference in performance when using the vendor supplied optimized libraries.

The electronic structure codes such as GAMESS and Gaussian, all require very high performance file system and disk access for reasonable and large sized problems. They also require high memory bandwidth and high performance math processing capabilities of CPUs in order to perform well. For these codes, large memory configurations are often important to obtaining reasonable performance.

The performance studies were limited to 64-bit computing platforms based on processors from AMD, Intel, and IBM. In April 2003 AMD introduced the AMD64 architecture, effectively bringing 64-bit computing to commodity computing status. Since then, wide-scale interest in this platform has generated demand for tools and software which take advantage of its capabilities, thus providing end users with more cost effective, non-proprietary computing platforms.

**Applications Tested**

The major applications covered in this report are Charmm 30b2 as supplied in binary form by Accelrys, GAMESS (April 2004), and AMBER 8. These applications are commonly used across large numbers of computational chemistry research teams. These three codes were chosen due to level of interest from end users polled in preparation for this work, as well as accessibility of code and/or binaries. (This list is not intended to be either complete or comprehensive.) This report and information will be revisited periodically, and where sensible and practical, additional codes and tests will be added.

Moreover, it is important to note that the tests performed are not necessarily representative of all types of research, and your specific problems of interest may not have similar performance characteristics. Users are urged to test their specific codes. This report provides a testing framework that will hopefully help simplify the testing regime.

Of the codes tested, only Charmm was restricted to binary-only with access to Accelrys release 30b2. This code was specifically compiled for a RedHat 7.2 platform without optimization for modern CPUs or 64-bit architectures. This Charmm installation was a 32-bit “legacy” program. This is important from the point of view of being able to preserve existing investment in programs for users without source code to recompile.

In all the other cases, original source code was compiled. The compilers used for these tests include Portland Group's AMD64 compiler version 5.2-4, PathScale compiler version 1.4, Intel Compiler version 8.1, gcc, and IBM's xlf/xlc compiler for PPC970.

In all cases except Charmm, the programs were compiled for best optimization that would pass all of the supplied tests. If a code produced errors in testing, the results would be reviewed to determine if the event was significant. An insignificant detrimental event was defined as an error in the last digit of energy. Significant detrimental events occurred when the results were off by several digits or the test did not run to completion.

Some platforms did not pass a significant number of the tests for some codes regardless of the optimization level settings. These are noted in benchmarks results as either significant detrimental events or test failed to complete. Resolving significant detrimental events and test failures was beyond the scope of the project.

**Evaluation Platforms**

The platforms tested included:

- AMD Opteron™ 246 processor (2.0 GHz) based upon a Tyan 2880 motherboard, with 2 GB DDR333 RAM, and 146 GB SATA RAID0 drive with an XFS file system running Fedora Core 2;
- AMD Opteron 250 processor (2.4 GHz) based upon a Newisys motherboard with 4 GB DDR400 RAM, and dual 80 GB SATA drives with ext3 file system running SuSE 9.2 x86_64;
- Dell PowerEdge 1850 with Intel Xeon EM64T (3.6 GHz) processors, 4 GB DDR2 RAM with dual SCSI 73 GB drives and XFS file system running SuSE 9.2 x86_64;
- Apple G5 dual CPU system with 4 GB RAM running OSX 10.3; and
- Itanium2 (1.5 GHz 6 MB cache) system with 64 GB RAM, and a large ext3 RAID5 array running SuSE 9.

The EM64T system was run in the 64-bit AMD64 mode as were the AMD Opteron based systems. The Apple system ran OSX 10.3, and is a 32 bit system. The Itanium2 is a 64 bit platform.

As was noted earlier, some platforms did not execute all codes. This was the case with GAMESS on the Apple platform with the IBM xlf/xlc compiler, and the EM64T based system with the Intel compiler.

It should be noted that the tests performed may not necessarily be representative of a particular usage pattern and requirements. Users' specific computational problems may have different performance characteristics.

**Benchmark Test Collection**

The AMBER 8 benchmark consisted of the internal test cases distributed with AMBER 8. as well as *jac*[1] and similar tests. The Charmm benchmark is the carboxy myglobin system with 3830 waters surrounding the protein. 1000 time steps were used for the 14026 atom simulation. Charmm 30b2 from Accelrys Inc. was used. Source code was not available for this test. The GAMESS benchmarks used the T.U. Dresden benchmark tests[2] . The GAMESS version used in these tests is the April 2004 code.

Each benchmark test was wrapped in an XML experiment file for BBS[3]. The benchmarks were run on idle machines with no other users. For Amber and GAMESS, the code was compiled with available relevant compilers. For AMD Opteron and the Intel EM64T three different compilers were used. The fastest benchmark time obtained from the different compilers was presented. For the Apple, the IBM xlf/xlc compiler was used.

Options were set in accordance with compiler recommendations for each code. Recommendations were obtained from compiler vendors, or groups that had extensive experience with the codes. Significant efforts were made on all platforms for GAMESS and AMBER to get the best performance while passing all test cases, using options that should result in faster execution. Performance was compared to the recommended options in all cases, and the best performing versions which passed the test cases was used. It should be noted that the IBM xlf/xlc compiler with GAMESS on the Apple machine was not able to pass all of the tests regardless of the optimization level (including turning off the optimization). As no other compilers for this platform were available at the time of the testing, resolving this problem was beyond the project scope. Likewise, the Intel compiler had problems with some of the test cases for Amber.

The procedure for handling these issues was to reduce the optimization levels gradually until the problem disappeared. If the problem did not go away after completely turning off optimization, this was noted. Differences of one or two digits in the last place were considered normal, and not an indication of a failure. It is to be expected different CPUs, with different compilers, optimization, and code generation pathways will emit binaries that accumulate round-off error slightly differently. Even on a single machine, different compilers produced different results depending upon the optimization levels. Only cases for real failures (significantly different results, erroneous results, and crashes) were given special consideration.

---

[1] *Jac* and related AMBER results may be found at http://amber.scripps.edu/amber8.bench1.html
[2] These benchmarks may be found at http://www.msg.ku.edu/~msg/MGM/links/bench.html
[3] BBS is the bioinformatics benchmark system, which despite its name, is a powerful general purpose benchmarking measurement tool.
http://www.scalableinformatics.com/metadot/index.pl?iid=2230&isa=Category&op=show#where

For the AMD Opteron processor and EM64T platforms, the compilers were used, each with settings which should generate the best code for the target platform. The PathScale and Portland Group compilers are capable of generating 64 bit code for all of these machines, as well as 32 bit code that target each of these processors.

Whenever possible for each application, relevant libraries such as the AMD ACML, the Intel MKL, and the Apple optimized libraries were linked. These libraries are run-time links and do make a significant difference in performance in many cases. In short, significant efforts were made to get optimal performance on each individual platform, subject to the requirement that the optimized program generated correct results for the supplied testing cases.

**Benchmark Results**

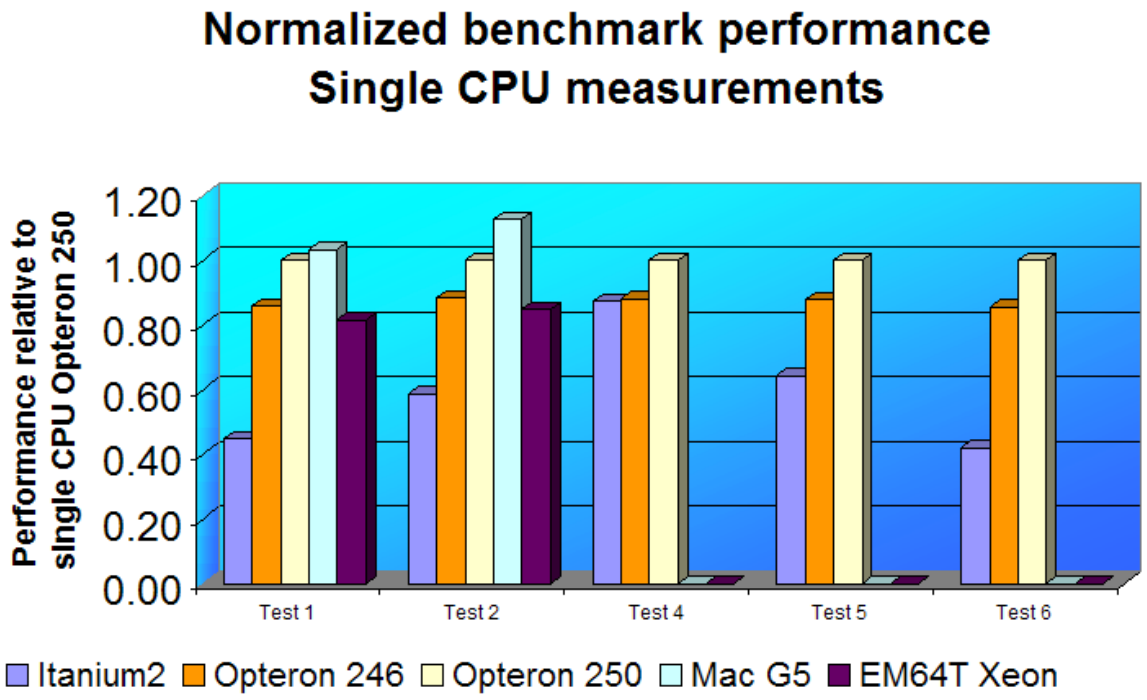
There are several ways users may chose to look at results and compare. One standard method is simply to compare run times, and normalize results to a particular "standard". Another method is to measure the cost performance, which would entail dividing the price by the execution time, to give a measure of the cost effectiveness of the platform.

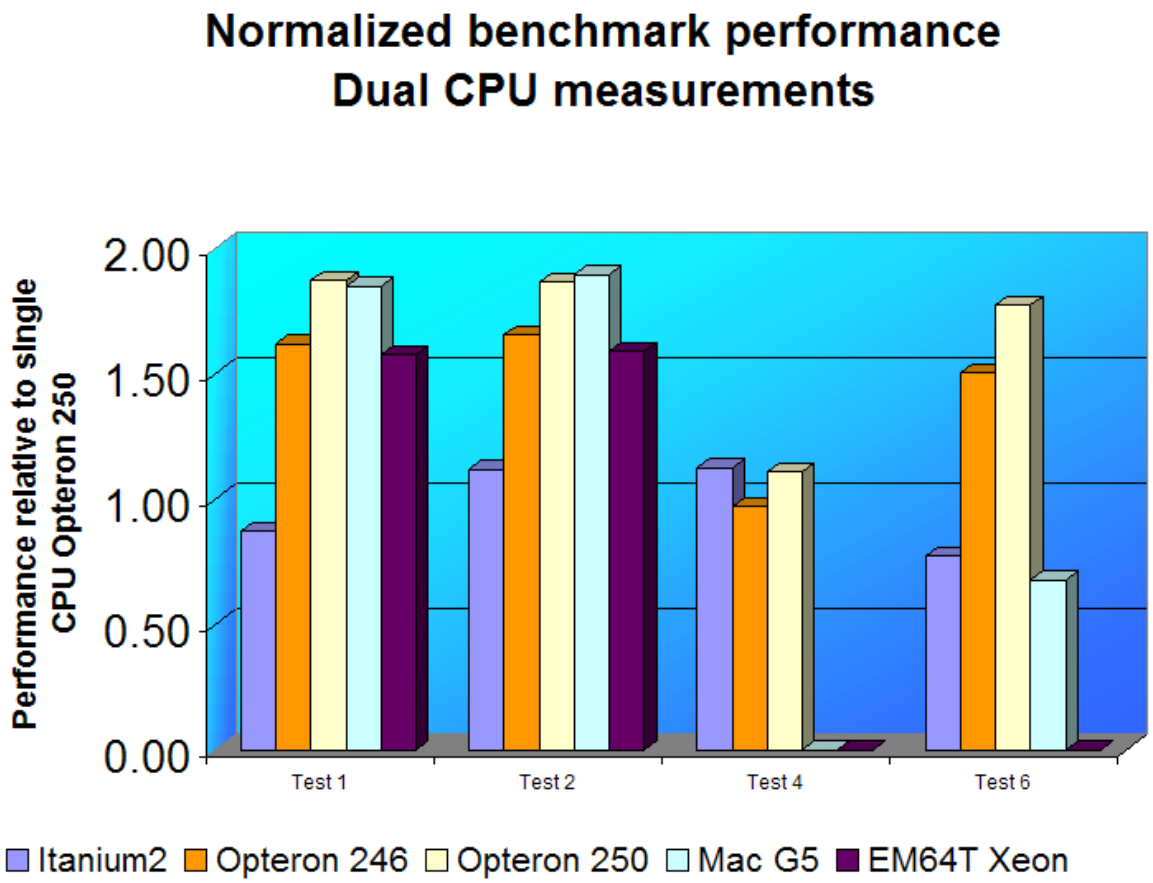
There are many other methods available, though for simplicity this report uses the metrics noted above. Further, where a machine failed a test (due to a crash), this will be reported. If all machines failed a test (this did occur with one GAMESS test), the test was eliminated.

## GAMESS results:

Data normalized to the single CPU AMD Opteron 250 time for particular test case, so that for each test case, values greater than one (1) indicate that the run completed faster than the AMD Opteron 250 processor (with the value being the ratio of the AMD Opteron 250 time on one CPU to the time for the test machine on as many CPUs as the test was run). So for Test 1, with two (2) CPUs, the AMD Opteron 250 is 1.87 times faster than the AMD Opteron 250 processor running the same test with a single CPU. Where individual tests resulted in a significant detrimental event or test failed to complete, they are so indicated. Though efforts were undertaken to attempt to understand the problems for future testing efforts, as noted earlier, resolving these problems was beyond the scope of the project.

| | | Machine Architecture | | | | |
|---|---|---|---|---|---|---|
| *Case* | *NCPU* | *Opteron 246* | *Opteron 250* | *EM64T Xeon* | *Mac G5* | *Itanium2* |
| *Test 1* | 1 | 0.86 | 1.00 | 0.81 | 1.03 | 0.45 |
| | 2 | 1.62 | 1.87 | 1.58 | 1.85 | 0.87 |
| *Test 2* | 1 | 0.88 | 1.00 | 0.85 | 1.13 | 0.59 |
| | 2 | 1.65 | 1.86 | 1.59 | 1.90 | 1.12 |
| *Test 4* | 1 | 0.88 | 1.00 | Failed | Failed | 0.87 |
| | 2 | 0.97 | 1.11 | Failed | Failed | 1.13 |
| *Test 5* | 1 | 0.88 | 1.00 | Failed | Failed | 0.64 |
| *Test 6* | 1 | 0.86 | 1.00 | Failed | Failed | 0.42 |
| | 2 | 1.50 | 1.77 | Failed | 0.68 | 0.77 |






An interesting set of trends appears to emerge. First, as clock speed is scaled by 20% on the AMD Opteron™ processor (from the AMD Opteron 246 to AMD Opteron 250[JEO1] processor), the performance increase on the test code is between 14-24%.[JEO2] This is likely due to the increase in speed of the on-chip memory controller, and the memory bandwidth bound nature of some aspect of these problems. Second, the 3.6 GHz EM64T appears on these tests to perform similarly to a 2.0 GHz AMD Opteron processor. It is important to note that this is similar to other detailed performance observations[4] on GAMESS, though using the GAMESS-UK variant.

## AMBER:

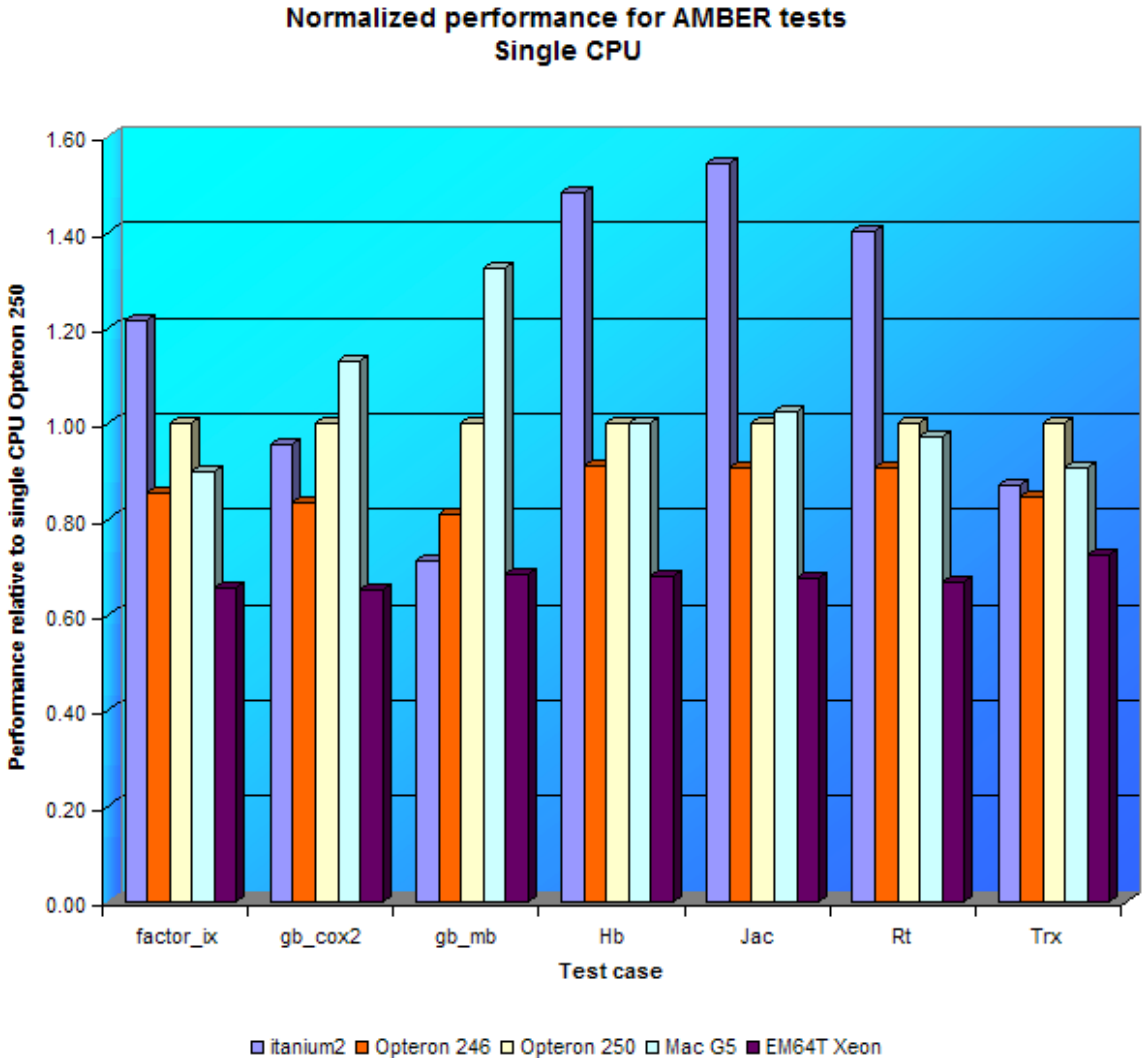
Similar to the GAMESS tests, all timing data was normalized to the AMD Opteron 250 processor time. Again, optimized binaries were generated subject to them passing all tests on the platforms.

The Itanium2 proved to be the best single CPU performer on these tests, as AMBER makes highly efficient use of its resources. As depicted in the data below, the AMD Opteron 250 processor and Mac G5 appeared to be the second best performers on these

[4] c.f. http://www.cse.clrc.ac.uk/disco/mew15-cd/Talks/Guest_CCLRC.pdf specifically pages 29, 30

tests. Note that some of the tests mapping very well into the the AMD Opteron™ architecture (eg. factor_ix) and others mapped better into the G5 architecture (eg. g_mb). Based on Jac, overall performance was comparable between the two architectures.

| | *Machine architecture* | | | | |
|---|---|---|---|---|---|
| ***Test case*** | ***Opteron 246*** | ***Opteron 250*** | ***EM64T Xeon*** | ***Mac G5*** | ***Itanium2*** |
| ***factor_ix*** | **0.85** | **1.00** | **0.65** | **0.90** | **1.21** |
| ***gb_cox2*** | **0.83** | **1.00** | **0.65** | **1.13** | **0.96** |
| ***gb_mb*** | **0.81** | **1.00** | **0.69** | **1.32** | **0.71** |
| ***Hb*** | **0.91** | **1.00** | **0.68** | **1.00** | **1.48** |
| ***Jac*** | **0.91** | **1.00** | **0.68** | **1.02** | **1.54** |
| ***Rt*** | **0.91** | **1.00** | **0.67** | **0.97** | **1.40** |
| ***Trx*** | **0.85** | **1.00** | **0.73** | **0.91** | **0.87** |



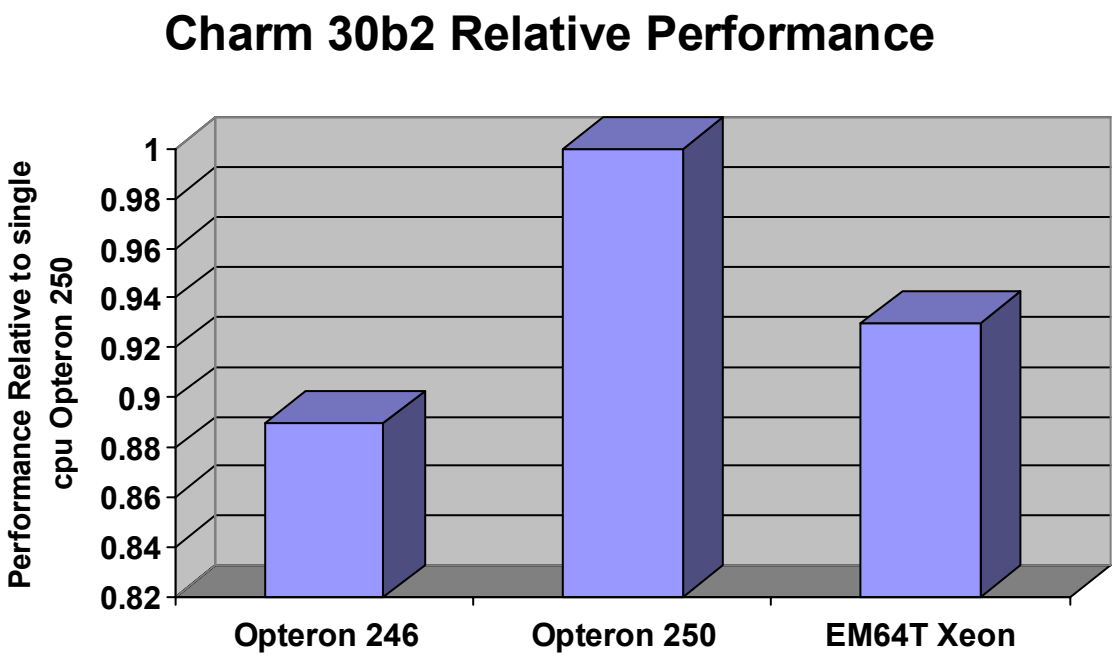
With AMBER in particular, users with a price/performance sensitivity should reference the information on price/performance presented in more detail in the appendix. While the Itanium2 was the best performer by nearly a factor of 2 in some cases, the high cost of the platform may effectively eliminate any performance advantage, as multiple non-Itanium2 platforms may be purchased at substantially the same price point. This can provide a performance advantage to these other platforms for the same price in a configuration that may be more versatile.

## Charmm:

Charmm 30b2 binaries, compiled for RedHat 7.2 as provided on the Accelrys CD, were tested on the AMD Opteron™ and EM64T platforms. No tuning or optimization is possible using pre-compiled binaries, so these tests were limited in scope. Regardless of these limitations, performance data was gathered, and from this a price/performance measurement was established.

The normalized timing information is provided in the table below. Again, timing is normalized to the AMD Opteron 250 platform.

| *Test* | *Opteron 246* | *Opteron 250* | *EM64T Xeon* |
|---|---|---|---|
| ***Mbco*** | 0.89 | 1.00 | 0.93 |



## Benchmark:

Benchmark input data and BBS input XML files will be supplied at http://www.scalableinformatics.com and http://scalability.org. Users may also log individual results, as well as suggested improvements for these tests and new tests at scalability.org. It is expected that platform, compiler, and OS enhancements will improve performance over time. Your input will help maintain current and relevant tests wherever possible.

## Conclusions:

It is worth noting that the only benchmarks that are really meaningful to most users would use their own codes and their own input data sets. This analysis makes use of some commonly used chemistry programs. However, there are many different programs, and exhaustive testing is not possible. Nevertheless, the tests were selected to support the principle that users would run similar configurations to come to conclusions like those in this report.

First, the AMD Opteron™ platform is performance competitive with the best of platforms for these classes of analyses. Given the characteristics of the codes run, it is reasonable to assume that one will obtain similar relative performance for customized computations.

Second, the AMD Opteron platform appears to have excellent price performance characteristics. In most cases, the codes demonstrated better performance. In one instance, (predominantly Amber codes) performance normalized for price continues to show an advantage to AMD Opteron processors. (See Appendix A for an exercise to measure price/performance.)

Third, based on the test criteria established for acceptable results, the compiler demonstrating the fastest results for the AMD Opteron and EM64T platforms was often the Portland Group, with the PathScale compiler occasionally generating better code. In most of the cases tested, the Intel compiler did not generate the fastest code across all three platforms, including the EM64T platform specifically. This was not explored in detail.

These tests were undertaken, in part, to understand the performance of these varied platforms. There are plans to revisit these results periodically to track progress in compilers, platforms, and related issues. Specific elements of these benchmarks will be included in the BBS baseline benchmark series.

## Appendix A
## Price/performance analysis

### Configuration pricing

The cost of each system as tested is in the table below

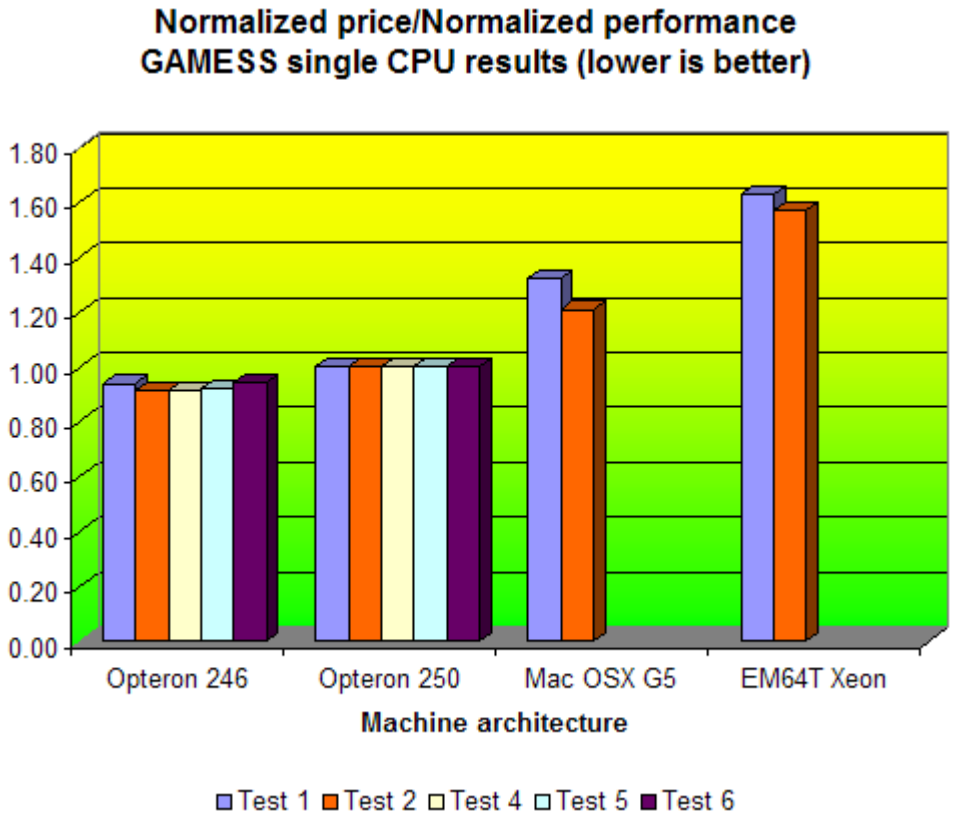
| Architecture | Configuration | | | Cost (in $US) |
|---|---|---|---|---|
| | CPU | Memory | compiler | |
| **AMD** | Dual AMD Opteron™ 250 CPU | 4 GB | PathScale 1.3, Portland Group 5.2-4, Intel 8.1 | $4250 |
| | Dual Opteron 246 CPU | 2 GB | PathScale 1.3, Portland Group 5.2-4, Intel 8.1 | $3430 |
| **Intel** | Dual EM64T 3.6 GHz CPU | 4 GB | PathScale 1.3, Portland Group 5.2-4, Intel 8.1 | $5651 |
| **Apple** | Dual G5 2.5 GHz CPU | 4 GB | IBM xlf/xlc | $5799 |
| **Intel** | SMP Itanium2 1.5 GHz 6MB cache CPU | 32 GB | Intel 8.1 | >$30000 |

Pricing data was gathered from various public sources including vendor web sites and resellers. Pricing reflects what is publicly available as of 28-February-2005. Pricing was for the systems and did not include operating system, compilers, or other tools. This information is not intended to be used as a pricing guide, only to set the numbers for price/performance analysis. Users are encouraged to substitute quoted prices to normalize the price/performance analysis.

### Price/performance analysis

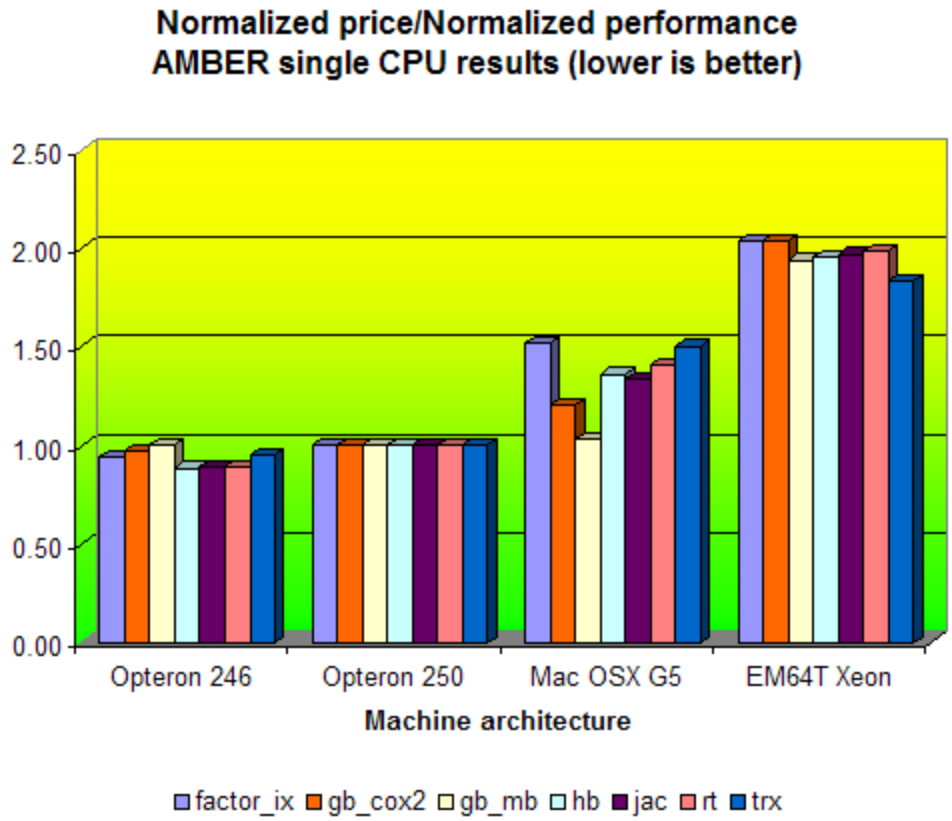
GAMESS

Working with the pricing data, and normalizing it to the AMD Opteron 250 processor-based system price allows a construction of an approximate price/performance metric. The following graph summarizes this analysis. For each test, the normalized price was divided by the normalized performance. Good price performance data would be a low value of this ratio, representing high performance for the money spent. The Itanium2 system was omitted from this graph as it had the largest values and plotting its values on the same graph would have unduly obscured the differences among the remaining systems.




From this analysis, the AMD Opteron™ 246 processor-based system appears to provide the best price performance on these tests, followed very closely by the AMD Opteron 250 processor-based platform, and further out, the Mac G5 platform, and the Intel EM64T Xeon.

AMBER




There are many factors that should be considered in comparing platforms for use. The Itanium 2 was not included in this chart because its price/performance factor was a significant outlier relative to the other systems (relatively poor in comparison). While the Itanium was the best performer by nearly a factor of 2 in some Amber cases, the high cost of the platform as previously discussed effectively eliminates any price/performance advantage. It has a different utilization/design profile. Similarly with the Mac G5, while several of the tests mapped well into its architecture, the cost effectiveness is appears to be better for the AMD Opteron™ platforms in this analysis of these test cases.

**Appendix B**
**Compiler options for various compilations**

| Program | architecture | Compiler | Compiler options |
|---|---|---|---|
| Amber8 | AMD64 | Portland Group 5.2-4 | F90: -O2 -tp k8-64 -g77libs -Msecond_underscore -fastsse<br>CC: -O2 -tp k8-64 -g77libs -Msecond_underscore |
| | AMD64 | PathScale 1.4 | F90: -Ofast -D/SETBOX/=/sztbks/<br>CC: -Ofast |
| | Itanium2 | Intel 8.1 | -w95 -ftz -vec_report3 -opt_report -opt_report_level max -opt_report_phase all -V -v –O3 -IPF_fma |
| | EM64T | Intel 8.1 | |
| | PPC970 (G5) | IBM xlf/xlc | |
| | | | |
| GAMESS | AMD64 | Portland Group 5.2-4 | -fastsse -Mipa=fast,safe -i8 |
| | AMD64 | PathScale 1.4 | -O3 -msse2 -mcpu=opteron -Ofast -mtune=opteron -LNO:OPT=0 |
| | Itanium2 | Intel 8.1 | -O3 -w95 -ftz -cm -WB -i8 (except for modules morokm,zheev,fsodci,grd2b,grd2c which needed –O0 to correctly build) |
| | EM64T | Intel 8.1 | -O3 -axNPW –quiet<br><br>-O3 -axNPW -unroll8 -pad -opt_report (for dftgrd,eigen,fmo, and int* modules) |
| | PPC970 (G5) | IBM xlf/xlc | |