Defense Technical Information Center Compilation Part Notice

ADP023780

TITLE: Sustained Systems Performance Test on HPCMP Systems

DISTRIBUTION: Approved for public release, distribution unlimited

This paper is part of the following report:

TITLE: Proceedings of the HPCMP Users Group Conference 2007. High Performance Computing Modernization Program: A Bridge to Future Defense held 18-21 June 2007 in Pittsburgh, Pennsylvania

To order the complete compilation report, use: ADA488707

The component part is provided here to allow users access to individually authored sections of proceedings, annals, symposia, etc. However, the component should be considered within the context of the overall compilation report and not as a stand-alone technical report.

The following component part numbers comprise the compilation report:
ADP023728 thru ADP023803

Sustained Systems Performance Test on HPCMP Systems

Paul M. Bennett
USACE Engineer Research and Development Center, Major Shared Resource Center (ERDC MSRC), Vicksburg, MS
Paul.M.Bennett@erdc.usace.army.mil

Abstract

This paper presents a brief description of the sustained system performance (SSP) test, which is designed to quantitatively track the performance of Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP) high performance computing (HPC) systems throughout their life cycles. The effects of incremental changes in system software, workarounds for hardware problems, and changes to scheduler operations on system performance can all be quantitatively evaluated. Data from runs of the SSP test conducted on selected HPC systems in the HPCMP will be used to demonstrate the benchmark.

1. Introduction

A standard operating practice for the HPC systems currently in use by the DoD HPCMP is to routinely upgrade system software, including the compilers and numerical libraries, perform hardware repairs or improvements, and adjust job queuing strategies. The goal is typically to eliminate as many software bugs as possible, work around hardware problems, optimize code for faster execution, or adjust job scheduling so as to improve overall job throughput or to benefit certain classes of jobs. Incremental changes of this nature tend to make the HPCMP's HPC systems more stable and efficient over time. However, greater stability and efficiency may be offset by increasing numbers of hardware failures because of aging equipment. Moreover, software improvements do not necessarily result in faster code, nor do changes to the scheduling strategies necessarily provide greater job throughput.

These considerations raise the issues of how to quantitatively monitor the HPCMP's HPC systems over their life cycles; how to determine the manner in which codes in the various computational technology areas (CTAs) respond to routine system updates; and how the system updates have influenced job execution. One approach is to execute a series of well-defined benchmarks that are carefully chosen to represent the CTAs that predominate on HPCMP systems at the shared resource centers. Variations in run times and overall job throughput can then be tracked in this production environment, thus giving a snapshot of the performance and throughput of the HPC system being tested.

Similarly to the test of the same name originally developed a the National Energy Research Scientific Computing Center (NERSC)[1], the SSP benchmark has been developed to achieve these goals on behalf of the DoD HPCMP. The benchmark consists of five benchmark codes and test cases carefully chosen from the Technology Insertion 2007 (TI-07) benchmark test suite. The test is executed periodically in order to track the performance over time of the systems under test.

2. SSP Test Considerations

The SSP test needs to be minimally visible to users of the HPCMP systems under test (SUT). This minimizes the numbers of jobs to run in each instance and the amount of time spent computing each test case. The total wall time to completion also needs to be minimized in order to require minimal utilization of the SUT.

On the other hand, the best data possible need to be collected in order to ensure the most complete and thorough evaluation possible. Test jobs executed at multiple CPU counts would allow the best scaling analysis, and the needs of the program would be best served by running jobs representative of each primary CTA.

Additionally, the CPU counts should be consistent across platforms and codes, and complications should be minimized. Therefore, codes with purely empirical evaluations of correctness should be reduced in number or eliminated entirely.

Furthermore, the test should be run as frequently as possible to regularly monitor the performance and throughput of the SUT.

0-7695-3088-5/07 $25.00 © 2007 IEEE

Unfortunately, these goals are mutually exclusive. Minimization of computation time tends to reduce the number of codes and the number of different CPU counts at which to execute them. Minimization of complication reduces the number of codes, making it more difficult to represent each CTA. On the other hand, thorough scaling analysis benefits from maximizing the number of codes and CPU counts, but at the cost of greater impact to the users.

With these competing needs in mind, the SSP has been assembled, consisting of the codes, test cases, and CPU counts listed in Table 1. The standard CPU count is the number of processing elements used to execute the standard test case jobs, and the large CPU count is the number used to execute the large test case runs.

Table 1. Codes, test cases, and CPU counts in the SSP test

Code	CTA	Standard CPU count	Large CPU count
AVUS	CFD	64	384
CTH	CSM	64	384
GAMESS	CCM	64	384
HYCOM	CWO	59	385
OOCORE	CEA	64	384

Both standard and large test cases are included, with data sets taken from the TI-07 Applications Benchmark Test Package. The CPU counts for the standard test cases are set to be 64, with the exception of HYbrid Coordinate Ocean Model (HYCOM). For that code, the closest CPU count is 59. For the large test cases, the CPU count is 384, again with the exception of HYCOM. Its CPU count must be 385. Codes with empirical accuracy tests are excluded, but every CTA occurring in the acquisition benchmark suite is represented. The sum of walltimes benchmarked in TI-07 is 2,438.6 CPU hours on Kraken, an IBM Power4+ architecture at the Naval Oceanographic Office Major Shared Resource Center (NAVO MSRC). Assuming perfect packing on 2,832 Kraken CPUs, the test is estimated to run to completion in 2 hours, minimizing the impact to users, although the test will not be conducted in this dedicated mode. Moreover, all of these codes have been demonstrated to run correctly on most HPC systems administered by the HPCMP. Scalability studies are based upon extrapolating from SSP walltimes observed at these CPU counts using data from TI-07 benchmarks for these five codes, and the SSP test is run on each SUT quarterly, preferably during the first month. Ideally, this frequency should fall between scheduled time-outs.

3. SSP Test Codes and Test Cases

The Air Vehicles Unstructured Solver (AVUS), formerly called Cobalt_{60}, is a parallel, implicit computational fluid dynamics (CFD) code that solves the compressible Euler and Navier-Stokes equations subject to the ideal gas equation of state[9]. The models can be two-dimensional (2D), three-dimensional (3D), or axisymmetric spaces, and unstructured grids with arbitrary cell types are permitted. Domain decomposition breaks the grid into subdomains called groups, blocks, or zones, permitting parallel processing in which each zone resides on a separate processor. This is accomplished using ParMETIS, the Message Passing Interface (MPI)-based, parallel grid-partitioning library that performs both static and dynamic graph partitioning and fill-reducing reordering. More information in the numeric algorithms in AVUS can be found in References 5, 10, and 13.

ParMETIS produces roughly equally sized zones, which produces good load balancing, and each zone has a minimized surface area, thus reducing communications overhead. Consequently, AVUS's excellent scalability may be attributed to two characteristics: (i) good load balancing with minimal communications overhead attributable to ParMETIS, and (ii) high computational intensity requiring little communication. More detailed information on ParMETIS may be found in Reference 10.

Both input test cases are 3D and model turbulent viscous flow over the geometries. The run times of the cases are set by advancing the solutions forward in time-varying numbers of steps. The standard test case ("wingflap_big") is a wind tunnel model of a wing with a flap and endplates, and it has 7,287,723 cells. The simulation runs for 200 time-steps. The large test case ("waverider") is a generic configuration for a supersonic/hypersonic vehicle that "rides" the shock wave that forms below the vehicle at such speeds, i.e., the attached shock generates lift for the vehicle. This model has 31,080,000 cells, and the simulation runs for 200 time-steps.

CTH is a computational structural mechanics code used to study the effects of strong shock waves on a variety of materials using many different models. It can solve problems on a static or adaptive mesh on a physical domain. Parallelism is implemented using MPI. CTH is a 3D version of an earlier code called "CHARTD," which stands for "Computational Hydrodynamics and Radiative Thermal Diffusion." CTH was developed by Sandia National Laboratories for modeling complex multidimensional and multimaterial simulations involving large deformations and strong shock physics. Hence, problems including penetration and perforation, compression, and detonations can be explored with the CTH software package.

The CTH code is highly portable and can be built to run on a single processor or with multiple processors utilizing MPI message passing. CTH has been designed to run on most platforms, including work stations, massively parallel machines, etc. CTH has been ported to and tested on numerous platforms under a variety of operating systems and programmatic interfaces, including the Cray XT3, the IBM Power4+, the SGI Origin and Altix, the SUN SPARC, and the Linux clusters. Only the MPI version of CTH is used in the SSP benchmark tests. The version is 7.1. More detailed information on CTH can be found in Reference 14.

The standard case is an Adaptive Mesh, Long Rod Penetrator Oblique Impact. This test case requires 840 MBytes of memory at all MPI processor counts. The model describes a 7.67 cm-long, 0.767 cm-diameter rod made of ten materials impacting a 0.64 cm-thick plate made of eight materials at an angle of 73.5 degrees. The initial velocity of the rod is 1,210 m/s. Each MPI process is allowed a maximum of 520 blocks, where each block contains $8 \times 8 \times 8$ cells. A maximum of five refinement levels can be used. The results are validated by looking at the depth of penetration of the long rod at the final time-step.

The large case is a Fixed Grid, Long Rod Penetrator Oblique Impact. This test case requires 204 GBytes of memory divided by the number of MPI processes. The model describes a 7.67 cm-long, 0.767 cm-diameter rod made of ten materials impacting a 0.64 cm-thick plate made of eight materials at an angle of 73.5 degrees. The initial velocity of the rod is 1,210 m/s. The fixed grid has $1840 \times 230 \times 460$ cells. The results are validated by looking at the depth of penetration of the long rod at the final time-step.

GAMESS stands for the General Atomic and Molecular Electronic Structure System. It is a quantum chemistry code currently maintained by Ames Laboratory and Iowa State University, available online at http://www.msg.ameslab.gov/GAMESS/GAMESS.html. GAMESS originated primarily from an early version of HONDO, which was funded by the National Science Foundation, the Department of Energy (DOE), and IBM. The development of GAMESS is ongoing at Iowa State University with sponsorship from the Air Force Office of Scientific Research and, more recently, the DOE. Many individuals and several research organizations have contributed to the development of GAMESS, of which a complete listing can be found in the user's guide. The guide is included in the source code distribution, and it contains information on the extensive set of capabilities of GAMESS, how to use them, and instructions on how to install GAMESS on many HPC platforms. More information on GAMESS may be found in References 1-3. Additional information on algorithms employed in GAMESS may be found in Reference 5. The version of GAMESS used in TI-07 was 27 JUN 2005 (R5).

The standard test case is a grid-based density functional theory computation of a polyhedral oligomeric silsesquioxane molecule. The molecular energy and gradient are computed using restricted Hartree-Fock calculation over a split-valence basis set of energy functionals, namely 6-311G**[6]. This basis approximates core electron shells using six Gaussian-type orbital functions, with two additional Slater-type orbitals to model the valance shells holding the molecule together. On hydrogen atoms, the additional orbital is modeled by a p type polarization function; but on all heavier atoms, it is modeled by a d type polarization function[7,8]. The standard test case is intended to require about 1–2 hours on 64 CPUs on the IBM Power4+ system "Kraken" at the NAVO MSRC. More information about this test case may be found in Reference 4.

The large test case is a second-order Moller-Plesset computation, based on an RHF computation, but with perturbations in the form of excitations included to account for electron correlation[9]. The molecule has the formula $BC_4N_{12}O_{12}$, and it has an extra electron making it a radical. The basis set for the electron shells is the same as in the standard test case. However, in addition to extra Slater-type valance orbitals in the standard test case, there are also an extra diffuse s shell on hydrogen atoms and an extra diffuse sp shell on atoms in the second and third rows of the periodic table, i.e., lithium, beryllium, and heavier nonmetals, halogens, metalloids, and post-transition metals in those tow rows[7]. The large test case is intended to require about 1 to 2 hours on 384 CPUs and between 0.5 and 1.0 GByte of memory on each of 256 CPUs of Kraken. More information about this test case may also be found in Reference 4.

HYCOM, or the Hybrid Coordinate Ocean Model, was developed by the HYCOM Consortium, which is part of the US Global Ocean Data Assimilation Experiment funded by the National Ocean Partnership Program. It is an open-source code, available at http://www.hycom.rsmas.miami.edu. HYCOM uses the traditional isopycnic coordinates in the open stratified ocean, like its predecessors NLOM, the Naval Layered Ocean Model, and MICOM, the Miami Isopycnic-Coordinate Ocean Model, both developed by the Naval Research Laboratory. However, HYCOM can transition between different types of vertical coordinates in shallow or mixed-layer waters. It uses a conventional second-order finite difference hydrostatic primitive model horizontally and an arbitrary Lagrange Eulerian coordinates system vertically. While communication can be performed using MPI or Cray's SHMEM library, MPI was used for the TI-07 benchmarks. Domain decomposition is used to distribute the problem in tiles of equal area. More detailed information may be found in References 11 and 12.

The TI-07 standard test case is a fully global ocean model with ¼-degree resolution at the equator. It simulates one model day, and it includes a representative amount of file input/output (I/O), computation, and communication. The initial state is generated from representative ocean profiles.

The large test case is also a fully global ocean model, but with 1/12-degree equatorial resolution. It also uses a representative amount of file I/O, computation, and communication, and it is also initialized using representative ocean profiles.

OOCORE is a code developed by Oak Ridge National Laboratory, the University of Tennessee at Knoxville, and several other universities. OOCORE was selected to represent the core execution of SWITCH, an electromagnetics code developed by Northrop Grumman. SWITCH itself cannot be used as an application benchmark because of its sensitivity. OOCORE solves systems of linear equations that may be too large for the main memory of a typical set of CPUs to contain. In lieu of main memory, OOCORE stores the coefficient matrix in temporary files on the system's disk, so it generates a large amount of disk I/O. A single OOCORE input parameter allows the user to artificially restrict the amount of main memory available for storing the matrix. Thus, large disk I/O can be ensured when needed for testing purposes such as benchmarking. Since OOCORE can be configured to perform by far the largest amount of disk I/O of any application code in the suite, it is the best available test in the suite of a platform's disk I/O capabilities. Additionally, OOCORE serves as the kernel of a code used by DoD HPC researchers investigating electromagnetic signatures. Including that code in the TI benchmarks was desirable but impossible because access to it is restricted. However, most of the cycles taken up in a typical run are spent in the kernel, so OOCORE acts as a useful benchmarking surrogate. More information on OOCORE can be found in Reference 4.

The OOCORE test cases consist of the standard test case, which solves a linear system with 53,000 double complex unknowns, and the large test case, which solves a linear system with 78,000 double complex unknowns. The factorization method in both cases is the LU decomposition. On each system individually, the memory usage in the standard test case is set to be 80 percent of the memory per CPU, so that the matrix will fit in memory on 64 CPUs and then be in-core. However, in the large test case, OOCORE is restricted to store a relatively small maximum of 1.8×10^6 matrix elements in the memory of each processor, so that the calculation is always out-of-core.

4. Results

Metrics such as application code compilation time, queue wait time, total SSP test throughput time, and application code execution time, for example, can be gathered to monitor a system's performance. A thorough discussion of all of these metrics is impossible in the space allotted here, so only the application code execution times will be discussed here.

The SSP test has been implemented on Sapphire, the Cray XT3 at the Engineer Research and Development Major Shared Resource Center (ERDC MSRC); Kraken, an IBM Power4+ at the Naval Oceanographic Office (NAVO) MSRC; Falcon, the HP XC at the Aeronautical Systems Center (ASC) MSRC; and Eagle, the SGI Altix at the ASC MSRC. The SSP test benchmark times are shown in the following tables, presented by system and test case size. The first column presents the SSP benchmark codes, and the second column presents the benchmark time observed in TI-07, in seconds. In a few instances, the TI-07 benchmark time could not be generated. Additional columns are labeled by the date on which the SSP test was initiated on the given system.

Table 2 presents the SSP benchmark times for the standard test cases executed on Sapphire. Sapphire is a Cray XT3 that currently features dual-core AMD 64-bit Opteron chips operating at 4.52 GFLOPS per core, 1 chip per node, connected by Cray's proprietary 3D torus. The system is a distributed-memory architecture with 2 GB memory per core, and Catamount as the operating system on the compute nodes. Sapphire was upgraded from a single-core AMD 64-bit Opteron system, operating at 4.52 GFLOPS with 1 chip per node, 2 GB memory per chip, in March 2007. The job scheduler was also changed from a Load Sharing Facility to a Portable Batch System, at that time.

In TI-07, Catamount was version 1.2.23 or earlier. For the first SSP test, Catamount's version was 1.3.38, and Catamount was at version 1.4.43 for the December 2006 SSP test. The current version of Catamount is 1.5.39.

The FORTRAN and C compilers are The Portland Group (PGI)'s, starting at version 6.1.1 in TI-07. On June 1, 2006, versions 6.1.6 of the compilers were installed. The current versions are 6.2.5, installed on November 2, 2006.

The AVUS times in September 2006, and again in December 2006, increased about 5 percent relative to the TI-07 benchmark time, because of changes in the operating system. The time increased about 15 percent relative to the December 2006 time after the upgrade to dual-core Opterons. The CTH times were quite close for the first two SSP benchmarks, but two additional compiler flags were used to compile CTH in the December 2006 SSP test. The first was a precompiler flag, –DMPI_CHECKSUM, which adds a great deal of overhead to MPI calls, and the second was the compiler flag –Kieee, which turns off certain optimizations that may produce non-IEEE compliant operations. These two compiler options produced significantly slower code. The options were not used in the April 2007 benchmark. GAMESS did not run to completion within a 4 hour maximal time limit for the September 2006 benchmark, because of revisions in the operating system, but for the benchmark in December 2006, the ACML library was upgraded from 2.7 to 3.0, with significant improvements in routines used in GAMESS. There were also improvements to the MPI routines. After the upgrade to dual-core Opterons, the time for GAMESS was 3 percent slower relative to the December time. The first two HYCOM benchmark times were within 2 percent of the TI-07 time, so the jump of 15 percent to 1,874 in April 2007 is significant. The OOCORE benchmark times were larger by 4 and 7 percent, respectively, relative to the TI-07 time, in the September and December tests. However, the time observed in the April test was only 5 percent longer than the TI-07 time. The OOCORE standard test case benchmarks were all performed in-core, so the entire matrix fit within main memory, showing that access times to main memory are the same for Sapphire after the upgrade as before.

Table 2. SSP test on Sapphire - standard test case

Code	TI-07	9/15/06	12/14/06	04/01/07
AVUS	1,791	1,884	1,860	2,151
CTH	1,420	1,463	2,021	1,999
GAMESS	12,711	14,456+	7,217	7,425
HYCOM	1,636	1,601	1,646	1,874
OOCORE	2,262	2,357	2,421	2,382

The large test case benchmark times on Sapphire are presented in Table 3. For AVUS, a variation of around 2 percent is seen, relative to the TI-07 time in September and then again in December 2006, but an increase of 15 percent in April 2007. The CTH large test case could not be completed in September 2006 because of a bug in MPICH which was reported by Tom Oppe and fixed in time for the December 2006 benchmarks. The CTH large test case times suffered from the same problem as the CTH standard test case times in December, and the April CTH benchmark time is about 34 percent larger than the TI-07 benchmark time. There were SSP test development issues that prevented the HYCOM large test case from executing in September 2006. However, the issues were resolved in time for the December and April benchmarks. The times were 7 and 6 percent larger, respectively, than the TI-07 HYCOM benchmark. The OOCORE large test case also suffered SSP test development issues that prevented completion of the benchmark runs in September 2006, but they were also resolved in time for the December 2006 and April 2007 benchmarks. In December, OOCORE's execution time was 18 percent longer than the TI-07 time, but in April 2007, the benchmark time was 19 percent shorter. This difference in time may be attributed to improvements in Sapphire's I/O hardware when the work directory was moved into a new partition as part of the upgrade to dual-core. The matrix must be written out to disk in the large test case, so improvements to the hardware resulted in a significant speedup to the benchmark time.

Table 3. SSP test on Sapphire - large test case

Code	TI-07	9/15/06	12/14/06	04/01/07
AVUS	2,527	2,584	2,572	2,896
CTH	2,087	—	2,846	2,793
GAMESS	8,883	9,269	6,230	6,261
HYCOM	1,892	—	2,037	2,003
OOCORE	2,023	—	2,379	1,638

In general, the performance of the SSP codes suffered when Sapphire was changed to a dual-core system. For CTH, which is more memory intensive than the other codes, the degradation was more severe, but the degradation was more than offset for OOCORE, which is I/O intensive, by hardware upgrades, and for GAMESS by improvements in messaging software before December 2006.

The SSP standard test case benchmark times observed on Kraken are presented in Table 4. Kraken is an IBM Power4+ system featuring p655 chips running AIX 5.2, which is a 64-bit operating system. The chips are single-core operating at 7.1 GFLOPS. There are eight chips per logical node, the nodes being connected by a high-speed network. The nodes used for the SSP benchmarks have 16 GB shared memory apiece, the nodes arranged in distributed-memory architecture.

AIX was upgraded from 5.2.0.75 to 5.2.0.86 in June 2006 and then to 5.2.0.97 in March 2007.

The FORTRAN and C compilers are IBM's, starting from versions 7.0.0.1 and 9.1.0.4, respectively, in TI-07, then being upgraded to 8.0.0.4 and 9.1.0.6 in August 2006. Upgrades to 8.0.0.7 and 9.1.0.8, respectively, were made on March 6, 2007. Additionally, LSF replaced LoadLeveler as the job scheduler after the TI-07 benchmark times were obtained.

The SSP test executed in May 2007 implemented IPM for the first time.

The AVUS benchmark time in December 2006 was longer than TI-07's by 3 percent, which may be due to the change in scheduler. The times in March and May were longer than TI-07's by about 6 percent. CTH's time in December was longer than in TI-07 by 6 percent, again possibly because of the change in batch scheduler. In the March and May SSP tests, CTH's times were longer by about 7.5 percent. In December, GAMESS' benchmark time was 8 percent longer than in TI-07, because of the environment variable MP_STDINMODE being set to the nondefault value of "0" for the TI-07 run but left at its default value of "all" in the December run. In the March SSP test and again in the May SSP test, executed with Internetwork Performance Monitor (IPM) enabled, both benchmark times were within 1.5 percent of the TI-07 benchmark. MP_STDINMODE was set to "0" for both runs. The HYCOM SSP test runs are within 5 percent of each other, but they are all significantly less than the TI-07 benchmark value. The reason is most likely that a particular environment variable that was set in the run scripts under LoadLeveler in TI-07 was not set in the run scripts under LSF for the later SSP test runs. OOCORE's performance has been very consistent throughout the TI-07 and SSP benchmark runs, varying by at most 1.1 percent, recorded in the December SSP benchmark run.

Table 4. SSP test on Kraken - standard test case

Code	TI-07	12/01/06	3/30/07	05/07/07
AVUS	2,793	2,887	2,951	2,955
CTH	2,432	2,571	2,615	2,636
GAMESS	4,859	5,268	4,936	4,888
HYCOM	2,502	2,069	2,108	2,087
OOCORE	7,181	7,100	7,105	7,142

As mentioned for the standard test case times, the SSP test executed in May 2007 was conducted with IPM enabled. A consequence of enabling IPM is that additional time is required to write and postprocess the data files containing the IPM data. These actions are automated, but the time required for 384 CPUs' worth of data is significant and is included in the SSP benchmark time. IPM, however, also records the execution time, without the IPM overhead. Table 5 records the benchmark times reported in the SSP test with IPM enabled compared with IPM's record of the benchmark time. The SSP benchmark times for the standard test case are listed under the column titled "SSP – std." The corresponding IPM times are listed in the next column under the heading "IPM – std." The column with heading "SSP – lrg" lists the SSP large test case times, and the corresponding IPM times are under the column with heading "IPM – lrg." For standard test case jobs, the overhead incurred by enabling IPM was around 1.5 percent. However, for the large test case jobs, the overhead varied from 3 percent for GAMESS and OOCORE to 22 percent for CTH.

Table 5. Comparison of benchmark times for the SSP test with benchmark times reported by IPM

Code	SSP - std	IPM - std	SSP - lrg	IPM - lrg
AVUS	2,955	2,910	4,649	4,158
CTH	2,636	2,537	4,330	3,545
GAMESS	4,888	4,836	7,163	6,983
HYCOM	2,087	—	2,502	—
OOCORE	7,142	7,052	4,688	4,560

Table 6 lists the times observed for the SSP benchmarks on Kraken. The May benchmark times are those reported by IPM. For AVUS, the SSP benchmark times in December and March were 10.6 percent and 10.1 percent longer than in TI-07. The IPM benchmark time was only 8.6 percent longer, being 59 seconds less than the March time. CTH's performance, on the other hand, was consistent, varying by at most 0.5 percent for the non-IPM runs from the TI-07 run. The IPM time was 111 seconds less than the March time. GAMESS' performance was similarly consistent, with the exception of the December time because leaving the value of the environment variable MP_STDINMODE at its default value and not set to "0" as with all other SSP benchmark times. The HYCOM times for the two non-IPM SSP benchmark tests differ by 6 seconds, and the IPM benchmark time is 85 seconds less than time recorded in March. However, all three times are roughly 80 percent of the TI-07 benchmark time. OOCORE has differences similar to HYCOM's except that the difference between the December and March benchmarks is 1 second and all three SSP benchmark times are roughly 90 percent of the TI-07 time. The IPM time for the OOCORE large test case was 70 seconds less than the March SSP time.

Table 6. SSP test on Kraken - large test case

Code	TI-07	12/01/06	03/30/07	05/07/07
AVUS	3,831	4,238	4,218	4,158
CTH	3,691	3,706	3,656	3,545
GAMESS	6,933	8,830	7,059	6,983
HYCOM	3,110	2,581	2,587	2,502
OOCORE	5,139	4,629	4,630	4,560

Falcon is an HP XC system at ASC MSRC featuring AMD 64-bit single-core Opteron chips operating at 5.6 GFLOPS per CPU. There are two CPUs per node, and the nodes are networked together by Infiniband. The system is a distributed-memory architecture with 2.25 GB memory per node. Jobs are scheduled by LSF. The operating system is a version of Red Hat Linux Release 4. The FORTRAN and C compilers are PGI's, version 6.1-2 in TI-07 and 7.0-4 for the May SSP test. Regrettably, the compiler version was not recorded for the January SSP test, but it was most likely 6.1.

The SSP standard test case times are recorded in Table 7. The AVUS benchmarks are all within 2 percent of the TI-07 times, and the CTH times are all within 1 percent of the TI-07 time. The GAMESS SSP times display much wider variation. This is possibly due to the specific node footprint upon which GAMESS executes at run time. A wider footprint means that the Falcon nodes are more widely separated, corresponding to a longer benchmark time by requiring more hops between switches in the interconnect. A narrower footprint yields a smaller benchmark time. The variation in GAMESS times may also be due to differences in the specific pathways between nodes that Infiniband computes in setup immediately prior to parallel execution. The other SSP application test codes do not exhibit such variations in the run times. The HYCOM SSP test times are within 5 percent of the TI-07 time, 5 percent being 58 seconds. Similarly, the OOCORE SSP benchmark times are within 2 percent of the TI-07 time, or 40 seconds difference.

Table 7. SSP test on Falcon - standard test case

Code	TI-07	01/09/07	05/16/07
AVUS	1,729	1,760	1,744
CTH	1,820	1,831	1,838
GAMESS	7,229	7,064	8,134
HYCOM	1,154	1,119	1,105
OOCORE	1,966	1,976	1,926

The SSP large test case benchmark times are presented in Table 8. AVUS displays small variation, similarly to the SSP standard test case times, all times falling within 30 seconds of the TI-07 benchmark time. The CTH large test case was 2 percent longer in the January SSP test than in TI-07, and 18.4 percent longer in the May SSP test. As with the standard test case times, the GAMESS large test case times vary by 35 percent less and 2.5 percent, respectively, for the January and May SSP tests. The HYCOM time in January was 2.5 percent less than the TI-07 time, but the time in May was nearly 27 percent longer than the TI-07 time. OOCORE also has a larger variation than for the standard test case times, being 5.5 and 7.1% percent shorter in January and May, respectively. As with GAMESS, this may be a consequence of node footprints of narrower width in the later runs, or it may be due to differences in the specific pathways that Infiniband sets for codes in setup immediately prior to parallel execution.

Table 8. SSP test on Falcon - large test case

Code	TI-07	01/09/07	05/16/07
AVUS	2,669	2,641	2,699
CTH	2,576	2,630	3,052
GAMESS	9,776	6,301	9,545
HYCOM	1,653	1,608	2,093
OOCORE	1,477	1,396	1,372

Eagle is an SGI Altix system featuring Intel Itanium two chips operating at 6.4 GFLOPS per CPU. Eagle has four shared-memory nodes, each consisting of 512 processors apiece, of which the user may access up to 500 at a time. Parallel processing across nodes is made possible by using Pluggable Authentication Modules (PAM). Each compute processor has 875 MB of user accessible memory, and LSF is the job scheduler. The compilers are currently Intel's FORTRAN and C compilers, versions 9.1.040 and 9.1.042, respectively. In TI-07, the versions were 8.1.020 and 8.1.023, respectively. The January SSP test codes were compiled using the current versions. The operating system is a version of Linux known as SLES9, kernel 2.4 at the time of the TI-07 benchmark test, but 2.6 at the time of the January SSP test. The kernel is currently 2.6.5-7.282-sn2.

The SSP standard test case benchmark times are presented in Table 9. The AVUS SSP time was 10 seconds shorter than the TI-07 time, but the CTH SSP benchmark time was 320 seconds longer than the TI-07 time. This is most likely due to setting ptiles in the TI-07 CTH benchmark run but not in the SSP CTH benchmark run. The effect of setting ptiles in LSF is to select CPUs that are next to each other, and therefore require much shorter latency and much less contention than CPUs that are more widely spaced. The effect is shorter job execution times when ptiles are specified. The GAMESS SSP benchmark time, on the other hand, is 768 seconds shorter, or 21.3 percent less, than the TI-07 benchmark time. Ptiles were not specified for either benchmark run, so the TI-07 test was most likely conducted on CPUs more widely spread apart than the SSP benchmark run. The HYCOM SSP time is 73 seconds, or 3.5 percent, shorter than the TI-07 benchmark time. However, the OOCORE SSP benchmark time is a whopping 87 percent less than the TI-07 benchmark time. The reason is that unoptimized portable basic linear algebra subroutines (BLAS) from SGI's ProPack 3 were used in the TI-07 benchmark test, but optimized BLAS from ProPack 4 were used in the January SSP benchmark test. Additionally, the newer operating system kernel had a positive effect in the SSP benchmark test.

Table 9. SSP test on Eagle - standard test case

Code	TI-07	01/09/07
AVUS	2,403	2,393
CTH	2,328	2,648
GAMESS	3,589	2,821
HYCOM	2,105	2,032
OOCORE	2,257	2,843

Table 10 lists the SSP large test case benchmark times observed on Eagle. Only CTH and GAMESS have benchmark times in both columns. The AVUS SSP benchmark failed because of a software issue concerning an obsolescent routine, drand48. A workaround to compile that particular routine at optimization level –O1 has been suggested but not yet implemented in the SSP tests on Eagle. The CTH large test case benchmark required 282 more seconds to complete in the January SSP test than the TI-07 benchmark, again because of ptiles not being specified in the SSP benchmark run. The GAMESS SSP benchmark was 194 seconds shorter in the SSP test than in the TI-07 benchmark run. Ptiles was not specified in either case. The HYCOM large test case could not be completed in TI-07. The reason is not clear. However, then benchmark was completed in the SSP test. The OOCORE large test case benchmark run could also not be completed, but this is due to the issue of nonoptimized portable BLAS in ProPack 3 being used in TI-07 as opposed to highly tuned BLAS in ProPack 4 being used in the SSP benchmark run.

Table 10. SSP test on Eagle - large test case

Code	TI-07	01/09/07
AVUS	4,648	—
CTH	3,918	4,200
GAMESS	3,869	3,675
HYCOM	—	2,902
OOCORE	—	7,610

Systems Used

The Cray XT3, Sapphire, at the ERDC MSRC; the IBM Power4+, Kraken, at the NAVO MSRC; and the SGI Altix, Eagle, and the HP XC, Falcon, both at ASC MSRC, are currently tested quarterly.

Computational Technology Areas

This work spans the computational biology, chemistry, and materials science (CCM), computational electromagnetics and acoustics (CEA), (CFD), computational structural mechanics (CSM), and climate/weather/ocean modeling and simulation (CWO) computational technology areas.

Acknowledgments

The SSP test data presented here were obtained by Dr. Paul M. Bennett on behalf of the High Performance Computing Modernization Program. Dr. Sam B. Cable and Dr. William A. Ward, Jr., assisted with initial planning. Dr. Thomas Oppe from the ERDC MSRC has assisted the bug diagnosis and testing effort on Eagle. This work was supported in part by a grant of computer time from the DoD HPCMP at the ASC, ERDC, and NAVO MSRCs.

References

1. Kramer, W., J. Shalf, and E. Strohmaier, "The NERSC Sustained System Performance (SSP) Metric." *Paper LBNL-58868*, Lawrence Berkeley National Laboratory, 2005.

1. Schmidt, M.W., K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S.J. Su, T.L. Windus, M. Dupuis, and J.A. Montgomery, "General Atomic and Molecular Electronic Structure System." *J. Comput. Chem.*, 14, pp. 1347–1363, 1993.

2. Gordon, M.S. and M.W. Schmidt, "Advances in electronic structure theory: GAMESS a decade later." In *Theory and Applications of Computational Chemistry, the first forty years*, eds. C.E. Dykstra, G. Frenking, K.S. Kim, and G.E. Scuseria, Elsevier, Amsterdam, 2005.

3. Bennett, P., S. Cable, R. Alter, M. Mahmoodi, and T. Oppe, "Targeting CCM-, CEA-, and CSM-Based Computing to Specific Architectures Based upon HPCMP Systems Assessment." *IEEE Proceedings of the DoD HPCMP Users Group Conference 2006*, Denver, CO, 26–29 June 2006, pp. 360–366.

4. Tracy, F.T., T.C. Oppe, W.A. Ward, Jr., and R.E. Peterkin, Jr., "A survey of the algorithms in the TI-03 application benchmarking suite with emphasis on linear system solvers." *IEEE Proceedings of the DoD HPCMP Users Group Conference 2003*, Bellevue, WA, 9–13 June 2003, pp. 332–336.

5. INPUT.DOC accompanying the GAMESS source code distribution.

6. REFS.DOC accompanying the GAMESS source code distribution.

7. http://en.wikipedia.org, entry on Computational Chemistry.

8. Tamaro, R.F., W.Z. Stang, and L.N. Sankar, "An Implicit Algorithm for Solving Time-Dependent Flows on Unstructered Grids." *AIAA 97-0333*, 35th Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan. 1997.

9. Cable, S.B., T.C. Oppe, W.A. Ward, Jr., R.L. Campbell, Jr., R.E. Gordnier, V.S. Burnley, M.J. Grismer, and P. G. Burning, "CFD-Based HPCMP Systems Assessment Using AERO, AVUS, and OVERFLOW-2." *IEEE Proceedings of the DoD*

HPCMP Users Group Conference 2005, Nashville, TN, 27–30 June 2005, pp. 349–355.

10. Leach, C.L., T.C. Oppe, W.A. Ward, Jr., and R.L. Campbell, Jr., "CWO-Based HPCMP Systems Assessment Using HYCOM and WRF." *IEEE Proceedings of the DoD HPCMP Users Group Conference 2005*, Nashville, TN, 27–30 June 2005, pp. 356–359.

11. Bleck, R., "An Oceanic General Circulation Model Framed in Hybrid Isopycnic-Cartesian Coordinates." *Ocean Modeling*, 4, pp. 55–88, 2002.

12. Tracy, F.T., "Role of Algorithms in Understanding Performance of the TI-05 Benchmark Suite." *IEEE Proceedings of the DoD HPCMP Users Group Conference 2005*, Nashville, TN, 27–30 June 2005, pp. 420–426.

13. Bennett, P., S. Cable, R. Alter, M. Mahmoodi, and T. Oppe, "CCM, CEA, and CSM-Based HPCMP Systems Assessment Using GAMESS, OOCORE, and RF-CTH." *IEEE Proceedings of the DoD HPCMP Users Group Conference 2005*, Nashville, TN, 27–30 June 2005, pp. 344–348.