CINT2.0 and CFP2.0 Benchmark Descriptions

By Kaivalya Dixit, Sun Microsystems, Inc.

CPU INTENSIVE INTEGER SUITE (CINT2.0)

The new component level CPU intensive integer benchmark suite contains six benchmarks. The benchmarks represent the following application areas: circuit theory, LISP interpreter, logic design, text compression, spreadsheet and software development (GNU compiler).

008.espresso

008.espresso is a tool for the generation and optimization of Programmable Logic Arrays (PLAs). It characterizes work done in the EDA market and logic-simulation and routing-algorithm areas. The elapsed time to run a set of four different input models characterizes espresso performance. The EECS department of the University of California at Berkeley distributes espresso through the Industrial Liaison Program. espresso is an integer benchmark written in C. It is a relatively small program that manipulates arrays in loops. This program exercises storage allocation in computing the PLAs and is sensitive to cache size.

022.li

022.li is a LISP interpreter written in C. It is a CPU intensive integer benchmark. The performance is measured by the time li takes to solve the famous nine queens problem. This benchmark was originally developed by David Michael Betz and is based on XLISP 1.6. XLISP is a small implementation of LISP with object-oriented programming. The backtracking algorithm is recursive and poses a challenge for register window architectures.

023.eqntott

023.eqntott is a CPU intensive integer benchmark written in C. The benchmark translates a logical representation of a Boolean equation to a truth table. The original source is from the Industrial Liaison Program of the University of California at Berkeley. The primary computation performed is a sort operation. The program fits comfortably in the instruction cache of most machines but the large data size may cause data cache misses in some machines.

026.compress

026.compress is an application that reduces the size of files using adaptive Lempel-Ziv coding. The SPEC version compresses and uncompresses a 1 MB file 20 times. It is a CPU intensive C benchmark that performs some I/O.

The amount of compression obtained depends on the size of the input, the number of bits per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50-60%. Compression is generally much better than that achieved by Huffman coding.

072.sc

The benchmark was originally developed by James Gosling. 072.sc is written in the C language and mainly performs integer computations.

072.sc is a spreadsheet benchmark. 072.sc performs standard operations expected of a spreadsheet: cursor movements, data entry, data movements, file handling, row and column operations, operations on ranges, and numeric expressions and evaluations. There are three input models that calculate in automatic mode:

- Budgets,
- SPECmarks,
- Fifteen-year amortization schedules.

All output is directed to a file to alleviate problems with outputting to different size windows. An efficient "curses" package should improve the performance on this benchmark.

085.gcc

085.gcc is GNU C compiler version 1.35. The original source came from Richard Stallman of Free Software Foundation. It is a CPU intensive C benchmark that spends about 10% of its time in I/O operations and thus is also an excellent system level benchmark. This benchmark characterizes work done in a workstation-based software engineering environment. The performance is sensitive to cache size and speed of the I/O device. 085.gcc shows high cache misses. The benchmark measures the time it takes for the GNU C compiler to convert a number of its preprocessed source files into optimized Sun-3 assembly language (.s files) output. The 085.gcc benchmark compiles 76 input files.

CPU INTENSIVE FLOATING POINT SUITE (CFP2.0)

The new component level CPU intensive floating point benchmark suite contains 14 benchmarks. The benchmarks represent the following application areas: circuit design, Monte Carlo simulation, quantum chemistry, optics, robotics, quantum physics, astrophysics, and other scientific and enginering problems. The suite contains two kernel benchmarks.

013.spice2g6

013.spice2g6 is an analog circuit simulation tool. It was developed by the IC group of the Electronics Research Laboratory and the EECS department of the University of California at Berkeley. It is written mostly in FORTRAN. The UNIX interface of the program is written in C. spice2g6 is a CPU intensive floating point (double precision) application. It is a real application heavily used in the EDA markets. It is a large program that runs five copies of the grey code circuit. The data accesses cause high cache misses. More than 80% of assignments are memory to memory transfer.

015.doduc

015.doduc is a Monte Carlo simulation of the time evolution of a thermo-hydraulical model ("hydrocode") for a nuclear reactor's component. It uses floating point numbers with 64-bit precision. The benchmark was developed by Nhuan Doduc. This is a synthetic benchmark that represents ECAD and high-energy physics applications. doduc is a non-vectorizable FORTRAN benchmark. It is a large kernel extracted from the original program and is popular in the scientific community. It has little I/O, abundance of short branches, loops, and executes code spread over many subroutines.

034.mdljdp2

034.mdljdp2 was developed by Steve Thompson at Cornell University. It is a double precision FORTRAN benchmark that represents quantum chemistry applications.

This program solves the equations of motion for a model of 500 atoms interacting through the idealized Lennard-Jones potential. This would be a typical system used to model the structure of (say) liquid argon. At each time step in the calculation, the position and velocity of each particle in the model system are used to calculate the configurational energy and pressure through the equations of statistical mechanics. These properties can be directly compared with experimental measurements, which gives some idea of the quality of the interatomic potential used (Lennard-Jones in this case).

The density and temperature for the model is supplied through an input file. One difficulty of comparing theoretical calculations with experimental measurements is the uncertainty in the interatomic potentials, which in general are not known for real systems. However, molecular dynamics calculations provide essentially exact results for a given model interatomic potential, and so provide a source of experimental data with which theory can be compared free of uncertainties in the potential.

039.wave5

The original source for this benchmark came from Los Alamos National Labs.

039.wave5 is a large FORTRAN scientific benchmark with single precision floating point arithmetic. A two-dimensional, relativistic, electromagnetic particle-in-cell simulation code used to study various plasma phenomena. WAVE5 solves Maxwell's equations and particle equations of motion on a cartesian mesh with a variety of field and particle boundary conditions. The benchmark problem involves 500,000 particles on 50,000 grid points for 5 time steps.

Changes for SPEC include:

- Scaled down from 20 to 5 time steps.
- Initialized some uninitialized (assumed to be zero) values.
- Disabled some code paths not used in the benchmark.
- Fixed some inconsistent common blocks.
- Normalized the code to prevent floating point exceptions in routine VSLV1P.
- Eliminated internal calculation of elapsed time (to permit SPEC's mechanical verification).

047.tomcatv

047.tomcatv is a highly (90 - 98 %) vectorizable program for the generation of two-dimensional boundary-fitted coordinate systems around general geometric domains such as airfoils, cars, etc. It is based on the method introduced by Thompson in 1974 which uses two Poisson equations to produce a mesh which adapts to the physical region of interest. The transformed non-linear equations are replaced by a finite difference approximation, and the resulting system is solved using successive line overrelaxation. The program is floating point intensive. The original FORTRAN source is from Dr. Wolfgang Gentzsch. This benchmark favors super-scalar and vector processors. It causes high data cache misses on several platforms.

048.ora

048.ora is a CPU intensive double precision floating point scientific FORTRAN program.

048.ora traces rays through an optical system composed of spherical and plane surfaces. Double precision is necessary on computers with 32 bit word length. Single precision is adequate on computers with 48 bit or greater word length. The SPEC benchmark will execute in double precision mode.

Optical Research Associates (ORA) sanitized a version of their proprietary geometric ray tracing benchmark.

The following modifications were made by SPEC:

- Internal timing commented out.
- Iteration count increased.
- Checksum modified to account for higher iteration count.
- Checksum validation test added.
- "Double Precision" changed to REAL*8 to avoid

problems on 64 bit machines.
- All constants were labeled COMMON in separate files.

052.alvinn

The original source came from D. A. Pomerleau. It is a CPU intensive single precision robotic application C program.

This program trains a neural network called ALVINN (Autonomous Land Vehicle In a Neural Network) using back propagation. ALVINN is designed to take as input sensory data from a video camera and a laser range finder and give as output the direction for a vehicle to travel in order to stay on the road.

The 1220 input units comprise the two input retinas, one from a video camera and one from a laser range finder. The 35 output units are a linear representation of the direction in which the network thinks the vehicle should travel. The network is fully connected and has 30 hidden units.

The file in_pats.txt contains 30 road scenes as input to the network and the file out_pats.txt contains the correct travel direction vectors for each of the scenes.

056.ear

The original source came from Apple. This is a single precision floating point intensive C benchmark. It makes extensive use of complex FFTs and other library functions. The program creates a 1 MB output data file.

This program simulates the human ear. The program takes as input a sound file (in a number of different file formats) and produces either a cochleagram or a correlagram. A cochleagram is a representation that roughly corresponds to spectral energy as a function of time. A correlagram is our implementation of Licklider's duplex model of pitch perception. A correlagram shows the short time autocorrelation for each time slice of each cochlear channel. The result is a two dimensional movie.

SPEC has modified the program to include validation of results and "filters" suggested by Malcolm Slaney. All machine-specific references (e.g. Cray) were removed.

077.mdljsp2

This benchmark was developed by Steve Thompson at Cornell University. It is a single precision FORTRAN benchmark that represents Quantum Chemistry applications.

This program solves the equations of motion for a model of 500 atoms interacting through the idealized Lennard-Jones potential. This would be a typical system used to model the structure of, say, liquid argon. At each time step in the calculation, the position and velocity of each particle in the model system are used to calculate the configurational energy and pressure through the equations of statistical mechanics. These properties can be directly compared with experimental measurements, which gives some idea of the quality of the interatomic potential used (Lennard-Jones in this case).

The density and temperature for the model is supplied through an input file. One difficulty of comparing theoretical calculations with experimental measurements is the uncertainty in the interatomic potentials, which in general are not known for real systems. However, molecular dynamics calculations provide essentially exact results for a given model interatomic potential, and so provide a source of experimental data with which theory can be compared free of uncertainties in the potential.

078.swm256

078.swm256 is a FORTRAN scientific benchmark with single precision floating point arithmetic. The original source came from Paul Swarztrauber of the National Center For Atmospheric Research (NCAR).

The program solves the system of shallow water equations using finite difference approximations on a 256x256 grid.

swm256 was modified for SPEC as follows:

- Internal calculation of MFLOPs eliminated (to permit SPEC mechanical verification of output).
- Primary benchmark metric is no longer MFLOPs but rather elapsed time for the program to complete.
- Removed multitasking directives (considered a performance enhancement/optimization by SPEC).
- Removed the writing out of initial matrix values and replaced the writing out of the final values with a checksum test. This was done to facilitate mechanical verification of results and to ensure that all computed results were subsequently "used."
- Changed the number of iterations and output printing policy to enlarge run time and reduce interim results printout.

The benchmark spends about 48% of its execution in the subroutine calc2.

089.su2cor

The original source came from B. Bunk, University of Wuppertal (Germany). It is a vectorizable FORTRAN program with double precision computation in quantum physics.

In this application program from the area of quantum physics, masses of elementary particles are computed in the

framework of the Quark-Gluon theory. The data are computed with a Monte Carlo method taken over from statistical mechanics. The configuration is generated by the warm bath method.

Most of the benchmark's code is highly vectorizable. The following values refer to a run with 75 configurations generated.

- Vectorization effect (factor) 15.0
- Vectorization degree 98.5 %

The benchmark was modified for SPEC as follows:

- Internal timing removed.
- Detailed output suppressed.
- Validation added.

090.hydro2d

The original source came from Dr. Koessl, Max-Planck-Insitut fur Astrophysik, Garching (Germany). It is a vectorizable FORTRAN program with double precision floating point computations. In this application program from the area of astrophysics, hydrodynamical Navier Stokes equations are solved to compute galactical jets.

Almost all of the benchmark's code is vectorizable. The following values refer to a run with 200 time steps.

- Vectorization effect (factor) 8.8
- Vectorization degree 99.5 %

093.nasa7

The original source came from David H. Bailey and John T. Barton, NASA, AMES. A complete description of nasa7 can be found in NASA Technical Memorandum 86711, "The NAS Kernel Benchmark Program" by David H. Bailey and John T. Barton. It is a FORTRAN program with double precision computation in applications frequently used by NASA.

The new benchmark reads the parameters from an input file to eliminate the constant propagation problem. Additionally, the original source file was split (with fsplit) into separate subroutines and cputime.c was added. The new output file contains the performance data for seven kernels which can be used to understand and improve the performance of individual kernels.

The seven kernels are:

- MATMUL - Matrix multiply operation.
- CFFT2d - Complex radix 2 FFT on 2D array.
- CHOLSKY - Cholesky decomposition in parallel on a set of input matrices.
- BTRIX - Block tridiagonal matrix solution along one dimension of a four dimensional array.
- GMTRY - Sets up arrays for a vortex method solution and performs Gaussian elimination on the resulting arrays.
- EMIT - Creates new vortices according to certain boundary conditions.
- VPENTA - Inverts 3 matrix pentadiagonals in a highly parallel fashion.

094.fpppp

094.fpppp is a double precision floating point FORTRAN benchmark. It is a quantum chemistry benchmark which measures performance on one style of computation (two electron integral derivative) which occurs in the Gaussian series of programs. It is difficult to vectorize because it contains very large basic blocks. The input file to 042.fpppp was modified and the source code remains unchanged. 094.fpppp solves the16 electron problem. The amount of computation is proportional to the fourth power of the number of atoms.

CREDITS AND CAVEATS

Most of the information is plagiarized from the benchmark descriptions provided by the SPEC project leaders. I want to thank Jeff Reilly (Intel) and Walter Bays (Sun Microsystems, Inc.) for editing.

Kaivalya Dixit is the SPEC president and an engineering program manager for Sun Microsystems, Inc., of Mountain View, California.