HPC Challenge Benchmark Suite

Jack Dongarra (UTK/ORNL)
Piotr Luszczek (ICL/UTK)

Accidental Benchmarker

- First benchmark report from 1977;
  - Cray 1 to DEC PDP-10
- Appendix B of the Linpack Users' Guide
  - Designed to help users extrapolate execution time for Linpack software package

---

My kids
Playstation 2 unoptimized.
My PDA running the benchmark in Java.
Linpack Benchmark Over Time

- **In the beginning there was the Linpack 100 Benchmark (1977)**
  - $n=100$ (80KB); size that would fit in all the machines
  - Fortran; 64 bit floating point arithmetic
  - No hand optimization (only compiler options)
- **Linpack 1000 (1986)**
  - $n=1000$ (8MB); wanted to see higher performance levels
  - Any language; 64 bit floating point arithmetic
  - Hand optimization OK
- **Linpack TPP (1991) (Top500; 1993)**
  - Any size ($n$ as large as you can; $n=1.2M$; 11.5TB; ~3 hours);
  - Any language; 64 bit floating point arithmetic
  - Hand optimization OK
    - Strassen's method not allowed (confuses the op count)
    - Reference implementation available
  - In all cases results are verified by looking at:
    - $\|Ax-b\| = O(1)$
    - Operations count for factorization $2/3n^3 - 1/2n^2$; solve $2n^2$

Motivation for Additional Benchmarks

- **Linpack Benchmark**
  - **Good**
    - One number
    - Simple to define & easy to rank
    - Allows problem size to change with machine and over time
    - Stresses the system with a long running job
  - **Bad**
    - Emphasizes only "peak" CPU speed and number of CPUs
    - Does not stress local bandwidth
    - Does not stress the network
    - Does not test gather/scatter
    - Ignores Amdahl's Law (Only does weak scaling)
  - **Ugly**
    - Benchmarketeering hype
  - Perhaps there was a time when this was adequate.
  - From Linpack Benchmark and Top500: "no single number can reflect overall performance"
  - Clearly need something more than Linpack
- **HPC Challenge Benchmark**
  - Test suite stresses not only the processors, but the memory system and the interconnect.
  - The real utility of the HPCC benchmarks are that architectures can be described with a wider range of metrics than just Flop/s from Linpack.
DARPA’s High Productivity Computing Systems

**Goals HPC Challenge Benchmark**

- Stress CPU, memory system, interconnect
- To **complement** the Top500 list
- To provide benchmarks that **bound** the performance of many real applications as a function of memory access characteristics
  - e.g., spatial and temporal locality
- Allow for optimizations
  - Record effort needed for tuning
  - Base run requires MPI and BLAS
- Provide verification of results
- Archive results
Tests on Single Processor and System

- **Local** - only a single processor is performing computations.
- **Embarrassingly Parallel** - each processor in the entire system is performing computations but they do no communicate with each other explicitly.
- **Global** - all processors in the system are performing computations and they explicitly communicate with each other.

HPC Challenge Benchmark

Consists of basically 7 benchmarks:
- Think of it as a framework or harness for adding benchmarks of interest.

1. **HPL** (LINPACK) — MPI Global (Ax = b)
2. **STREAM** — Local; single CPU
   - STREAM — Embarrassingly parallel
3. **PTRANS** (A ↔ A + BT) — MPI Global
4. **RandomAccess** — Local; single CPU
   - RandomAccess — Embarrassingly parallel
5. **BW and Latency** — MPI
6. **FFT** — Global, single CPU, and EP
7. **Matrix Multiply** — single CPU and EP
Computational Resources and HPC Challenge Benchmarks

Computational Resources

- CPU computational speed
- Memory bandwidth
- Node Interconnect bandwidth

HPL Matrix Multiply

STREAM

Random & Natural Ring Bandwidth & Latency
Memory Access Patterns

- **Spatial Locality**
  - **Computational Fluid Dynamics**
  - **Radar Cross Section**
  - **Traveling Sales Person**

- **Temporal Locality**
  - Low
  - High

Applications

**STREAM (EP & SP)**
- **PTRANS (G)**

**HPL Linpack (G)**
- **Matrix Mult (EP & SP)**

**Radial Access (G, EP, & SP)**
- **FFT (G, EP, & SP)**

The Zoom-FFT Algorithm
How Does The Benchmarking Work?

- Single program to download and run
  - Simple input file similar to HPL input
- Base Run and Optimization Run
  - Base run must be made
  - User supplies MPI and the BLAS
  - Optimized run allowed to replace certain routines
    - User specifies what was done
- Results upload via website
- HTML table and Excel spreadsheet generated with performance results
  - Intentionally we are not providing a single figure of merit (no overall ranking)
- Goal: no more than 2 X the time to execute HPL.

HPC Challenge Languages

<table>
<thead>
<tr>
<th>Language</th>
<th>HPL</th>
<th>Random Access</th>
<th>STREAM</th>
<th>FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>UTK</td>
<td>UTK</td>
<td>UTK</td>
<td>UTK</td>
</tr>
<tr>
<td>C</td>
<td>UTK</td>
<td>UTK</td>
<td>UTK</td>
<td>UTK</td>
</tr>
<tr>
<td>C &amp; MPI</td>
<td>UTK</td>
<td>UTK</td>
<td>UTK</td>
<td>UTK</td>
</tr>
<tr>
<td>C &amp; OpenMP</td>
<td>UTK</td>
<td>UTK</td>
<td>UTK</td>
<td>UTK</td>
</tr>
<tr>
<td>UPC</td>
<td></td>
<td></td>
<td></td>
<td>ISI</td>
</tr>
<tr>
<td>C &amp; pthreads</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fortran</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matlab</td>
<td>MIT-LL</td>
<td>MIT-LL</td>
<td>MIT-LL</td>
<td>MIT-LL</td>
</tr>
<tr>
<td>Matlab &amp; MPI</td>
<td>UTK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StarP</td>
<td>UCSB</td>
<td>UCSB</td>
<td>UCSB</td>
<td>UCSB</td>
</tr>
<tr>
<td>pMatlab</td>
<td>MIT-LL</td>
<td>MIT-LL</td>
<td>MIT-LL</td>
<td>MIT-LL</td>
</tr>
<tr>
<td>Octave</td>
<td>OSC</td>
<td>OSC</td>
<td>OSC</td>
<td>OSC</td>
</tr>
<tr>
<td>Python</td>
<td>UTK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python &amp; MPI</td>
<td>UTK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapel</td>
<td>Cray</td>
<td>Cray</td>
<td>Cray</td>
<td>Cray</td>
</tr>
<tr>
<td>X10</td>
<td>IBM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fortress</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HPC Challenge Benchmark

The HPC Challenge benchmark consists of basically 7 benchmarks:

1. **HPL** - the Linpack TFP benchmark which measures the floating point rate of execution for solving a linear system of equations.

2. **DGEMM** - measures the floating point rate of execution of double precision real matrix-matrix multiplication.

3. **STREAM** - a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.

4. **STPRAM** (parallel matrix transpose) - exercises the communication between processes to measure the effectiveness of the network.

5. **SendReceive** - measures the rate of integer random updates of memory (ULPS).

6. **SfP** - measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

7. Communication bandwidth and latency - a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns based on eeff (effective bandwidth benchmark).
HPCC Kiviat Chart

Three systems using the same processor and number of processors.
- AMD Opteron 64 processors 2.2 GHz
  - Cray XD1
  - Dalco Linux Cluster Quadrics Interconnect
  - Sun Fire Cluster Gigabit ethernet Interconnect

HPCC: A Comparison of 3 Systems

- AMD Opteron 64 processors 2.2 GHz
  - Cray XD1
  - Dalco Linux Cluster Quadrics Interconnect
  - Sun Fire Cluster Gigabit ethernet Interconnect
Cray XD1, Sun cluster, SGI Altix shared memory machine

- **SGI Altix**  
  Itanium 1.6 GHz

- **Cray XD1**  
  AMD 2.2 GHz

- **Sun GigE**  
  AMD 2.2 GHz

**HPCC Awards 2005 Info and Rules**

**Class 1 (Objective)**
- **Performance**
  1. G-HPL $500
  2. G-RandomAccess $500
  3. EP-STREAM system $500
  4. G-FFT $500
- **Only full submissions from HPCC database**

**Class 2 (Subjective)**
- **Productivity (Elegant Implementation)**
  - Implement at least two tests from Class 1
  - $1500 (may be split)
- **Deadline:**
  - October 15, 2005
  - Select 3 as finalists
- **Submissions format flexible**
  - Finalists will present at SC|05 in the HPCC BOF

**Winners (in both classes) will be announced at SC|05 HPCC BOF on Tuesday November 15th at noon.**
Class 1:
If Awards Given Today, the Winners …

- **Global HPL**
  - **Cray XT3 AMD Opteron ORNL**
  - 5200 proc; 2.4 GHz Opteron
  - 20.5 Tflop/s

- **Global Random Access**
  - **Cray X1E ORNL**
  - 248 proc; Cray X1E 1.13 GHz
  - Optimized run using UPC.
  - 1.855 Gup/s

- **EP-Stream (triad) for the System**
  - **Cray XT3 AMD Opteron ORNL**
  - 5200 proc; 2.4 GHz Opteron
  - 26020 GB/s

- **Global FFT**
  - **Cray XT3 AMD Opteron DOD-ERDC**
  - 4096 proc; 2.6 GHz Opteron
  - 906 Gflop/s

Class 2: Implementation Languages (Subjective)

- **English (Paper and pencil)**
- **C/C++**
  - MPI-1, MPI-2, OpenMP, pthreads
- **Fortran 90/95/03**
- **Java**
- **Matlab**
  - MatlabMPI, StarP, pMatlab
- **Python**
  - MPI
- **UPC, CAF**
- **Chapel, X10, Fortress**

- **HPCC tests**
  - FFT
  - HPL
  - RandomAccess
  - STREAM

- **Good if 2 of the 4 tests actually run**
Committee Members

- David Bailey
  - LBNL NERSC
- Jack Dongarra (Co-Chair)
  - University of Tenn/ORNL
- Jeremy Kepner (Co-Chair)
  - MIT Lincoln Lab
- David Koester
  - MITRE
- Bob Lucas
  - ISI
- Rusty Lusk
  - Argonne National Lab
- Piotr Luszczek
  - University of Tennessee
- John McCalpin
  - IBM Austin
- Rolf Rabenseifner
  - HLRS, Stuttgart
- Daisuke Takahashi
  - University of Tsukuba

Future Directions

- Looking at reducing execution time
- Constructing a framework for benchmarks
- Developing machine signatures
- Plans are to expand the benchmark collection
  - Sparse matrix operations
  - I/O
  - Smith-Waterman (sequence alignment)
- Port to new systems
- Provide more implementations
  - Languages (Fortran, UPC, Co-Array)
  - Environments
  - Paradigms
Collaborators

♦ HPC Challenge
  ▶ Piotr Łuszczek, U of Tennessee
  ▶ David Bailey, NERSC/LBL
  ▶ Jeremy Kepner, MIT Lincoln Lab
  ▶ David Koester, MITRE
  ▶ Bob Lucas, ISI/USC
  ▶ Rusty Lusk, ANL
  ▶ John McCalpin, IBM, Austin
  ▶ Rolf Rabenseifner, HLRS Stuttgart
  ▶ Daisuke Takahashi, Tsukuba, Japan

http://icl.cs.utk.edu/hpcc/