64-Bit Floating-Point Accelerators for HPC Applications

John L. Gustafson, Ph.D.
Chief Technology Officer, HPC
ClearSpeed Technology, Inc.
Outline

• Acceleration issues in general: host-accelerator bandwidth/latency
• Accelerator performance analysis examples
• Measured and expected application acceleration:
  – Molecular Dynamics: AMBER and NAB
  – Quantum Chemistry: GAUSSIAN, Qbox, PARATEC
  – Monte Carlo models for PDEs
  – LS-DYNA and ANSYS
  – PAM-CRASH
  – MATLAB applications
• Summary
Thesis

- Performance analysis for accelerator cards is like analysis for message-passing parallelism, but with more levels of memory and communication.
- Application porting success depends heavily on attention to memory bandwidths, but (surprisingly) not so much the host-accelerator bandwidth.
The accelerator idea is as old as supercomputing itself

Even in 1977, HPC users faced issues of when it makes sense to use floating-point-intensive vector hardware.

“History doesn’t repeat itself, but it does rhyme.”
—Mark Twain
### Study Results: Accelerators

#### Use of Applications Accelerators

Q: Do you have any plans to use applications accelerators?
   Multiple responses allowed.

<table>
<thead>
<tr>
<th>Accelerator</th>
<th>Number of Mentions</th>
<th>Percentage of Responding Sites Mentioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGAs</td>
<td>28</td>
<td>90.3%</td>
</tr>
<tr>
<td>Vector coprocessors</td>
<td>13</td>
<td>41.9%</td>
</tr>
<tr>
<td>GPUs</td>
<td>10</td>
<td>32.3%</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td>3.2%</td>
</tr>
</tbody>
</table>

n = 31

Source: IDC, 2006

©2007 IDC
Is this trip necessary? Bandwidth issues

- Acceleration software tests candidates for work on the board. If too small, it leaves them on the host.
- Performance claims *must* assume host-resident data. Beware of benchmarks that leave out the time to move the data to accelerator memory.
- Remember Bailey’s 12 Ways…
Simple offload model is out of date

- Accelerator must be quite fast for this approach to have benefit
- This “mental picture” may stem from early days of Intel 80x87, Motorola 6888x math coprocessors
Acceleration model: Host continues working

- Accelerator needs only be fast enough to make up for time lost to bandwidth + latency
- Easiest use model: host and accelerator share the same task, load balanced to complete at same time
- More flexible: Host, accelerator *specialize* what they do
Host can work while data is moved

- PCI transfers might burden a *single x86 core* by 60%
- Other cores on host continue productive work at full speed
Card need not wait for all data before starting

- In practice, latency is *microseconds*; the accelerator task takes *seconds*. Latency gaps above would be microscopic if drawn to scale.
- The accelerator can be *slower* than the host, and still add performance!
Square DGEMM speeds as of December 2006

Doubling host-to-card bandwidth has minor effect because of I/O overlap and large task grain size. A zero-latency connection would not visibly affect either curve.

Note: curve only samples integer multiples of vector size
Accelerator memory hierarchy

Tier 3
- Host DRAM: 1–32 GBytes typical

Tier 2
- Bank 1
- Bank 0
- CSX DRAM: 0.5 GBytes

Tier 1
- PE95
- PE0
- Poly memory: 6 KBytes
- Per PE
- Register memory: 128 Bytes
- Swazzle path

Tier 0
- Per PE
- Arithmetic: 0.5 GFLOPS

Total:
- 1.0 GB
- 6.4 GB/s
- 1.1 MB
- 192 GB/s
- 24 KB
- 2 TB/s
- 96 GFLOPS (but only 25 watts)
Matrix multiply (DGEMM) is a perfect analog to a folded box.

Volume is the number of multiply-adds.

Surface “padding” shows overheads.
Build up the bricks through the hierarchy

Tier 1

Tier 2

Tier 1 matmul

Tier 2 matmul

$Tier\ 2\ \alpha\ \text{overhead} = 2$ → $n_2 = 1,$

$n_1 n_2 = 8$

$k_2 = 2$

$k_1 k_2 = 192$

$m_2 = 3,$

$m_1 m_2 = 288$

$\beta_1 = 4.5$

$Tier\ 2\ \beta\ \text{overhead} = 3$
Tier 3 overhead was originally harder to overlap

Simple but decent accelerator performance model:

\[
\frac{90.2}{1 + \left( \frac{400}{M} + \frac{200}{K} + \frac{200}{N} \right)} \text{ GFLOPS.}
\]
Unoptimized Tier 3 timing

\[ K=960, \ M=1920, \ N=1920 \]
Optimized Tier 3 timing

Can now accelerate matrices as small as N=576
Almost doubled sustained speed

Revised DGEMM asymptotic to ~90 GFLOPS

Previous DGEMM asymptotic to ~56 GFLOPS

Does not include host contribution, which adds 5 to 60 GFLOPS depending on host

Note: curve only samples integer multiples of vector size
Detailed profiling is essential for accelerator tuning

**HOST CODE PROFILING**
- Visually inspect multiple host threads.
- Time specific code sections.
- Check overlap of host threads

**HOST/BOARD INTERACTION**
- View host/board interactions
- Measure transfer bandwidth.
- Check overlap of host and board compute

**ACCELERATOR PIPE**
- View instruction issue.
- Visualize overlap of executing instructions.
- Get cycle-accurate timing.
- Remove instruction-level performance bottlenecks.

**CSX600 SYSTEM**
- Trace at system level.
- Inspect overlap of compute and I/O.
- View cache utilization.
- Graph performance.
Profile the code running at the instruction level

See the pipeline performance for each instruction

Tune the instruction scheduling for the application code
System level: multiple DGEMM calls

View the DGEMM calls on host and accelerator.

Each call syncs up with the host view of the accelerator.

Much higher level of detail available from the profiler.
More geometric analogs: equation solving

Dongarra’s new LAPACK will make QR, Cholesky, LU factoring, etc. much easier to accelerate.

Volume = $\frac{1}{3} N^3$

multiply-adds

Excellent test of hardware correctness
$R_{est} = \frac{1}{PQ\gamma} + \frac{3\alpha [(N_B + 1) \lg P + P]}{2N^2N_B} + \frac{3\beta(3P + Q)}{4NPQ}$

$N_B =$ Block size, the width of the “panels” used to update the linear system with DGEMM

$N =$ Dimension of the linear system (number of equations to solve)

$P, Q =$ Dimensions of the two-dimensional mapping of computational nodes

$\alpha =$ Effective point-to-point latency of MPI broadcast, in seconds

$\beta =$ Effective point-to-point reciprocal bandwidth of message broadcast, in seconds per datum

$\gamma =$ Effective floating-point operations per second of a node independent of MPI operations
Memory bandwidth dominates performance model

- Apps that can stage into local RAM (Tier 1) can go 10x faster than current high-end Intel, AMD hosts
- Apps that must reside in DRAM (Tier 2) will actually run slower by about 3x (for fully optimized host code)
- Fast Fourier Transforms can go either way!
Math functions reside at Tier 1, hence fast

Typical speedup of ~8X over the fastest x86 processors, because math functions stay in the local memory on the card
Monte Carlo PDE methods exploit Tier 1 bandwidth

Real apps do work resembling “EP” of NAS Parallel Benchmarks. “Quants” solve PDEs this way for options pricing, Black-Scholes model (a form of the Heat Equation)

- No acceleration: 200M samples, 79 seconds
- 1 accelerator: 200M samples, 3.6 seconds
- 5 accelerators: 200M samples, 0.7 seconds
Why do EP-type Monte Carlo apps need 64-bit?

- Accuracy increases as the square root of the number of trials, so five-decimal accuracy takes 10 billion trials.
- But, when you sum many similar values, you start to scrape off all the significant digits.
- 64-bit summation needed to get a single-precision result!

Single precision:
\[ 1.0000 \times 10^8 + 1 = 1.0000 \times 10^8 \]

Double precision:
\[ 1.0000 \times 10^8 + 1 = 1.00000001 \times 10^8 \]
We may need to rethink 64-bit flops...

- Every operation has an optimum number of bits of accuracy
  - Using too few gives unacceptable errors
  - Using too many wastes memory, bandwidth, joules, dollars.
- It is unlikely that a code uses *just the right amount* of precision needed.

![Optimum precision graph](chart.png)
How do HPC programmers pick FP precision?

- Assume 64-bit is plenty, and use it everywhere.
- Use what is imposed by hardware (word size).
- Try two precisions; if answers agree, use the less precise one, otherwise use the more precise one.
- Compare computed answer for special cases where an analytic answer is known.
- Compare computed answer with physical experiment (rare).
- Perform careful analysis (very rare).
Can a tool estimate joules, W, $, min. precision?

\[ a(i,j) = a(i,k) \times a(k,j) \]

<table>
<thead>
<tr>
<th>Tier 0 read/write</th>
<th>Tier 1 read</th>
<th>Tier 2 read</th>
</tr>
</thead>
<tbody>
<tr>
<td>28 pJ, $2 \times 10^{-15}$</td>
<td>50 pJ, $8 \times 10^{-14}$</td>
<td>1900 pJ, $5 \times 10^{-13}$</td>
</tr>
</tbody>
</table>

64-bit op
42 bits needed
10 pJ
$1 \times 10^{-15}$

64-bit op
39 bits needed
12 pJ
$1 \times 10^{-15}$

Cost and electrical power and precision are almost as important as timing… why not develop analysis tools for them? You can only optimize what you can measure.
Model/measure power use, not just performance

Standard Cluster Power Consumption / Node

Average Power Consumption: 430.6 watts

Base System: HP DL380 G5, Intel Xeon 5160 x 2 @ 3GHz, 14GB
Accelerated Cluster Node

Base System: HP DL380 G5, Intel Xeon 5160 x 2 @ 3GHz, 14GB
Accelerated System: As above + 1 ClearSpeed Advance X620

Average Power Consumption: 437.5 watts

20 Minutes

watts

550
500
450
400
350
300
250
Energy Usage: Standard vs. Accelerated

Comparative Cluster Power Consumption / Node

- **Standard Cluster Node**
- **CS Accelerated Node**

- **53.6% Less Energy Used**

Base System: HP DL380 G5, Intel Xeon 5160 x 2 @ 3GHz, 14GB
Accelerated System: As above + 1 ClearSpeed Advance X620

- **42 Minutes**
- **watts**

- **250**
- **300**
- **350**
- **400**
- **450**
- **500**
- **550**
AMBER 9 acceleration

- Model: memory motion is $k_1N$, operations are $k_2N^2$. Overheads easily overcome for typical $N$.
- ~4x speedup for pharmaceutical company production runs achieved recently.
- 225 hour (Opteron) run reduced to 58 hours
- Didn’t exploit symmetry of $ij$ forces, which will give another ~1.5–1.9x speedup (31–39 hours)
- Solvation model GB1; 500000 time steps
- Correctness checked incrementally throughout conversion process.
NAB and AMBER 10 acceleration

- Newton-Raphson refinement now possible; analytically-computed second derivatives
- 2.6x speedup obtained for this operation in three hours of effort (no source code changes)
- Enables accurate computation of entropy and Gibbs free energy for first time.
- Available now in NAB (Nucleic Acid Builder) code. Slated for addition to AMBER 10.
Plug-and-play quantum chemistry acceleration?

- DGEMM content is 18% to 65% in GAUSSIAN test suite, but typical sizes only ~10 to 100.
- No changes to license or to any of the source code. Just invoke dynamic linking option in makefile.
- Sample GAUSSIAN tests to date are too small to accelerate; below $N = 576$ threshold.
  - Need larger problem sizes
  - Realistic to be that large? (Lots of occupied orbitals)
- PARATEC, Qbox much better candidates. Plane wave models are over half DGEMM, huge dimensions
The economics of CAE acceleration

- Each host costs $3,000.
- Software license costs ~$30,000 per core, which discourages use of multiple cores.
- MCAE engineer costs over $200,000/year.
- In California, anyway.
- Accelerator card would be cost-effective even with a 7% performance boost. Actual performance boost should be more like 260% for large problems.

Structural Analysis
ANSYS, LS-DYNA implicit
Accelerating ANSYS, LS-DYNA with Lucas’ solver

• Potentially pure plug-and-play
• No added license fee
• Needs ClearSpeed’s 64-bit precision and speed
• Enabled by recent DGEMM improvements; still needs symmetric $A^T A$ variant
• Could enable some CFD acceleration (for codes based on finite elements, low Reynolds numbers`)

10 million degrees of freedom (sparse)

becomes...

50,000 dense equivalent

Accelerator can solve at over 50 GFLOPS

Est. 260% net application acceleration

Non-solver
Solver setup

DGEMM on x86 host

10x

Non-solver
Solver setup

DGEMM with ClearSpeed
PAM-CRASH and MATLAB acceleration

• Work done in China shows 1.40x PAM-CRASH speedup using one ClearSpeed accelerator

• We await details; this is preliminary
  – Problem size?
  – Explicit or implicit?
  – What was done to the code?
  – Compared to what host?

• Japanese industrial researcher got 5x acceleration of MATLAB waveguide model; won’t allow publication of results (?)

• *sigh* So we continue citing TOP500 results…
Summary

• Accelerator tuning demands attention to memory bandwidth at all levels (bandwidth to host, less so)

• Now seeing value for real 64-bit applications in chemistry, electromagnetics, financial modelling, crash codes, etc.

• Fit-for-purpose analysis starts with analytical model based on memory tiers, but is verified using detailed performance tools.