ENVISION. ACCELERATE. ARRIVE.

64-Bit Floating-Point Accelerators for HPC Applications

John L. Gustafson, Ph.D.
CTO, High Performance Computing
ClearSpeed Technology, Inc.
Outline

• Acceleration issues in general: FPGAs, GPUs, host-accelerator bandwidth/latency
• Description of ClearSpeed system
• Predicting LINPACK performance
• Predicting real application performance:
  – General principles
  – LS-DYNA and ANSYS
  – AMBER and NAB
  – GAUSSIAN…
Thesis – I

• Despite impressive advances in processor technology, it remains impractical to design a general-purpose chip that is also well-designed for HPC applications.

• The use of accelerator plug-in cards lets us combine the advantages of commodity x86 processors with HPC-specific hardware. It moves the HPC tradeoff decisions from the chip designer to the user.
Thesis II

- Accelerator cards can increase performance on highly specific tasks, *without* aggravating facility limits on clusters (power, size)
- **Need to consider**
  - Specificity of application
  - Software
  - Data type and precision
  - Compatibility with host (logical and physical)
  - Memory size (local to accelerator)
  - Latency and bandwidth to host
- **Surprise: latency and bandwidth aren’t as limiting as the other things!**
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 hardware.

“History doesn’t repeat itself, but it does rhyme.”
—Mark Twain
Some questions to ask

• Is my main data type integer or floating-point, and what precision do I need?
• How much data needs to be local to the accelerated task?
• Does existing accelerator software meet my needs, or do I have to write my own?
• Am I trying to improve performance, or do I want to improve the ratio of performance to something else (price, power consumption, or footprint)?
• Will the accelerator still benefit when I take into account the time to move data to it and back?
Is this trip necessary? Bandwidth issues

- Time to move $N$ data to/from another node or an accelerator is \( \sim \text{latency} + N/B \) seconds.
- Because local memory bandwidth is usually higher than $B$, acceleration might be lost in the communication time.
- Estimate the breakeven point for the task (note: offloading is different from accelerating, where host continues working). More on this later…

![Diagram of bandwidth comparison between node and accelerator](image-url)
The broad categories of computing accelerators

- **Field Programmable Gate Arrays (FPGAs)**
- **Game/Graphics Processing Units (GPUs)**
- **HPC-specific accelerators (ClearSpeed)**
FPGAs work best for bit and integer data types

- Excellent for bit-twiddling, like cryptography
- Fast for integer manipulations, like genomic algorithms for pattern matching
- Marginal for 32-bit floating-point; have to do over 200 at once to compete with current general purpose chips
- Poor at 64-bit floating-point… don’t use them as a supplementary FLOPS unit
FPGA issues

• “Programming” is really circuit design, though tools are making this easier.
• Compare speed against meticulously coded assembler on node, not casual coding on node.
• Where FPGAs make the most sense is in creating instructions very unlike those provided by the node instruction set.
• FPGAs can cost more than an entire server!
• If socket-based, you might be better off with another host CPU instead.
• Usually use less power than a host CPU, but more than a custom VLSI chip to do same task.
Attack of the Killer PlayStations: GPUs for HPC?

The Beowulf approach has its limits…

Whatever happened to…
- Sandia’s HPC cluster of Nintendo 64s?
- Argonne’s HPC cluster of PlayStation 2s?
Where GPUs can help with HPC applications

- Single-precision calculations where answer quality is less important than raw speed
  - Seismic exploration
  - Some types of Quantum Chromodynamics
- Graphics-type calculations, obviously, for visualization and result display
GPU accelerator issues

- Only 32-bit, and non-IEEE rounding degrades accuracy cumulatively
- Can use over 200 watts, multiple slots
- Often only have a few megabytes of local store (frame buffer architecture)
- Cheap hardware but very expensive software (the kind you create yourself)
- Game processors don’t match endian-ness of node CPUs
Attack of the Killer PlayStations: GPUs for HPC?

The Beowulf approach has its limits...

Whatever happened to...
• Sandia’s HPC cluster of Nintendo 64s?
• Argonne’s HPC cluster of PlayStation 2s?
Use a second car to add 8 feet to front of car

Not shown: rolling the car over to squash the roof so it won’t look silly.
There you have it! A luxury mid-engine car at a tremendous savings in cost.
To make this baby look extra-sharp, put a pizza box over the wheel wells to hide where the tires used to be.
• The Nvidia and ATI model of use has led to *dynamically-linked libraries*.

• HPC could never have overcome that hurdle by itself with ISVs. (Widespread use of MPI helped, too.)

• Plug-and-play acceleration is now available for environments like MATLAB, Mathematica®… with more on the way.
• 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 need only be fast enough to make up for time lost to bandwidth + latency
- Easiest use model: host and accelerator share the same task, like a DGEMM
- 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
The accelerator can be slower than the host, and still add performance!

In practice, latency is microseconds; the accelerator task takes seconds. Latency gaps above would be microscopic if drawn to scale.
An HPC-specific accelerator

- HPC acceleration coprocessor, the CSX600
  - Can be installed via add-in cards, e.g. PCI-X, PCI Express, …
  - …or potentially integrated on the motherboard
  - Can also be used for embedded applications, e.g. aerospace, auto, medical, defense
- Significantly accelerates some libraries and applications
  - Target libraries: Level 3 BLAS, LAPACK, FFTs; MKL, ACML
  - Mathematical modelling tools: Mathematica, MATLAB etc.
  - In-house codes: Can use SDK to port compute-intensive kernels
- ClearSpeed’s Advance™ card is aimed at x86-based servers (running Linux or Windows)
  - Dual CSX600 coprocessors
  - Sustains up to 75 GFLOPS for 64-bit matrix multiply (DGEMM)
  - Low power; dissipates about 25 Watts
Example of accelerator chip designed specifically for HPC

- Array of 96 Processor Elements; 64-bit and 32-bit floating point
- 210-250 MHz… key to low power
- 47% logic, 53% memory
  - About 50% of the logic is FPUs
  - Hence around one quarter of the chip is floating point hardware
- About 1 TB/sec internal bandwidth
- 128 million transistors
- Approximately 10 Watts

ClearSpeed CSX600
Multi-Threaded Array Processing
- Programmed in familiar languages
- Hardware multi-threading
- Asynchronous, overlapped I/O
- Run-time extensible instruction set
- Bi-endian for compatibility

Array of 96 Processor Elements (PEs)
- Each is a Very Long Instruction Word (VLIW) core, not just an ALU
Each PE is a VLIW core:

- Multiple execution units
  - 4-stage floating point adder
  - 4-stage floating point multiplier
  - Divide/square root unit
  - Fixed-point MAC 16x16 → 32+64
  - Integer ALU with shifter
  - Load/store
- High-bandwidth, 5-port register file (3r, 2w)
- Closely coupled 6 KB SRAM for data
- High bandwidth per PE DMA (PIO)
- Per PE address generators
  - Complete pointer model, including parallel pointer chasing and vectors of addresses
Memory hierarchy

Arrow labels are in GB/sec. Multiple instances not shown, for clarity.
The *Advance™* card minimizes facilities costs

- Over 70 DGEMM GFLOPS sustained for 210 MHz version
- 8.8 ounces, 8 inches long, 25 watts for entire card (at socket)
- Single PCI slot (PCI-X or PCIe 8-lane)
- At least 1 Gbyte of local DRAM... enough to stay busy
- About 1 Gbyte/s to/from card from host
All accelerators are good... for their intended purpose

**FPGAs**
- Good for integer, bit-level ops
- Programming looks like circuit design
- Low power per chip, but 20x more power than custom VLSI
- Not for 64-bit FLOPS

**Cell and GPUs**
- Good for video gaming tasks
- 32-bit FLOPS, not IEEE
- Unconventional programming model
- Small local memory
- High power consumption (> 200 W)

**ClearSpeed**
- Good for HPC applications
- IEEE 64-bit and 32-bit FLOPS
- Custom VLSI, true coprocessor
- At least 1 GB local memory
- Very low power consumption (25 W)
- Familiar programming model
Software improvements to ClearSpeed DGEMM

New DGEMM asymptotic to 85 GFLOPS

Previous DGEMM asymptotic to 50 GFLOPS

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

Note: curve only samples integer multiples of vector size
Doubling host-to-card bandwidth has minor effect because of I/O overlap. A zero-latency connection would not visibly affect either curve!

Note: curve only samples integer multiples of vector size
ClearSpeed SDK overview

- **C^n compiler**
  - C with long-established extension for SIMD control
  - Assembler
- **Linker**
- **Simulator**
- **Debugger**
- **Libraries**
- **Documentation**
- **Training materials**
- **Posix (GNU/Linux) look and feel**
- **Available for Windows and Linux (RedHat 4 and SUSE 9)**
Gdb/ddd debugger

Port of standard gdb enables standard GUIs to work with the CSX600:

- Hardware supports single step, breakpoint etc
- gdb port is multi-everything (thread, processor and card)
- Visualize all the state in the PEs
- Hardware performance counters also exposed via gdb
• The CSX600 is 8-way threaded:
  – Typically 1 compute thread and 1 or more I/O threads

• The hardware supports tracing in real-time:
  – Thread switches
  – I/O operation start/finish

• Profiler relates run-time information back to the source code
Clusters are hitting facilities limits

• 20–30 kilowatts per cabinet, 300–1500 watts/node)
• Footprint is also a constraint; each rack consumes about 10 sq. ft.
• Typical cabinet (500–700 pounds) taxes floor loading limits
• Cost of cabinet is ~$400,000 to $1,200,000 for ~50 to 150 processor sockets + communication
Example of a possible 1U server

- Two ClearSpeed PCIe cards on risers
- 1 or 2 GB DRAM/card
- Enclosure supplies 25W, cooling per card
- Total power draw: 450 W

- 277 peak GFLOPS (64-bit)
- ~225 DGEMM GFLOPS sustained with MKL+new DGEMM
- ~170 LINPACK GFLOPS sustained
- 18 or 20 GB DRAM on host: 2 or 4 GB on CS cards, 16 GB on x86 host
Solving \( N \) equations takes order \( N^3 \) work

ClearSpeed accelerates the DGEMM kernel of equation solving that takes over 90% of the time.

Volume = \( \frac{1}{3} N^3 \) multiply-adds
An accurate LINPACK cluster acceleration model

\[ R_{est} = \frac{1}{PQ\gamma} + \frac{3\alpha[(N_B + 1)\log P + P]}{2N^2N_B} + \frac{3\beta(3P + Q)}{4NPQ} \]

\( N_B = \) 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
Hypothetical 1U-based cabinet

- 40 servers with 2.66 GHz x86 quad-core
- 0.64 TB DRAM
- 3.4 TFLOPS peak
- ~2.8 TFLOPS LINPACK (82% eff.)
- 24 kW
- 10 sq. ft.
- 800 pounds
- ~$400,000 with IB

Add 80 ClearSpeed Advance cards
- 0.80 TB DRAM
- 11 TFLOPS peak
- ~7 TFLOPS LINPACK (64% eff.)
- 26 kW
- 10 sq. ft.
- 850 pounds
- < $1,000,000

ClearSpeed increases...
- Power draw by 8%
- Floor space by 0%
- Weight by 6%
- Speed by 150%

Single-cabinet TOP500 supercomputer
• Typical HPC clusters are limited by
  – Power dissipation (heat)
  – Cost
  – Space requirements
  – Usable PCI-X or PCIe slots per node

• Consider two dimensions to the optimization:
  – $N_c$ = Number of ClearSpeed accelerator cards
  – $N_h$ = Number of host nodes (multicore)
First constraint: achieving performance

- 1 PFLOP = 1 million GFLOPS
  - Each CSX600 card contributes, say, ~54 GFLOPS
  - An x86 server contributes, say, ~35 GFLOPS
- **Constraint is** $54N_c + 35N_h \geq 1000000$
Second constraint: Power budget

- Suppose the facility supports, at most, 3 megawatts
  - Each CSX600 card contributes ~25 watts
  - Each host contributes, say, ~600 watts
- Constraint is $25 \, N_c + 600 \, N_h \leq 3000000$

$\begin{align*}
N_h \\
\downarrow
\end{align*}$

$\begin{align*}
5000 \\
\downarrow
\end{align*}$

Less than 3 megawatts

$\begin{align*}
120000 \\
\downarrow
\end{align*}$

$\begin{align*}
N_c
\end{align*}$
Third constraint: PCI slots per host

- Assume we can have either 1 or 2 CSX600 cards per host
  - 2:1 ratio intersects the feasible set
  - Cheapest case: 4615 x86 hosts, two CSX600 cards each
- Under $100M, 116 cabinets, 3 MW, 1200 square feet
Announced October 9th 2006: Tokyo Tech accelerated their Linux supercomputer, TSUBAME, from 38 TFLOPS to 47 TFLOPS with 360 ClearSpeed Advance cards

- Performance increase of 24% for just a 1% increase in power consumption, 5% increase in cost
- That was with the *old* DGEMM. Watch for new results soon.
- 10,368 AMD Opteron cores with just 360 ClearSpeed cards
- #9 in November 2006 Top500
- 1st accelerated system in the Top500

*Professor Matsuoka standing beside TSUBAME at Tokyo Tech*
LINPACK speed correlates with many real applications

- Ab initio Computational Chemistry
- Structural Analysis
- Electromagnetic Modeling
- Radar Cross-Section
- Global Illumination Graphics
Uses for ClearSpeed after the TOP500 press release...

- **Dense matrix-matrix kernels**: order $N^3$ ops on order $N^2$ data
  - Boundary element and Green’s function methods
  - Gaussian, NAB, other chemistry codes use DGEMM intensively
- **$N$-body interactions**: order $N^2$ ops on order $N$ data
  - Astrophysics, low-density CFD, molecular mechanics
  - Look to MD-GRAPE for examples
- **Some sparse matrix operations**: order $NB^2$ ops on order $NB$ data where $B$ is the average matrix band size
  - Structural analysis, implicit PDEs generally
- **Time-space marching**: order $N^4$ ops on order $N^3$ data
  - Explicit finite difference methods; data must reside on card
Memory (not PCI) bandwidth dictates performance

- Apps that can stage into local RAM can go 10x faster than current high-end Intel, AMD hosts.
- Apps that reside in DRAM will actually run slower by about 3x.
- Fast Fourier Transforms can go either way!
- Cell, GPUs, face very similar issue (but much harder to program!)
Example: math function speed comparison

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

- **Monte Carlo methods are ideal for ClearSpeed acceleration:**
  - High regularity and locality of the algorithm
  - Very high compute to I/O ratio
  - Very good scalability to high degrees of parallelism
  - Needs 64-bit

- **Excellent results for parallelization**
  - Achieving 10X performance per Advance card vs. highly optimized code on the fastest x86 CPUs available today
  - Maintains high precision required by the computations
    - True 64-bit IEEE 754 floating point throughout
  - 25 W per card typical when card is computing

- **ClearSpeed has a Monte Carlo example code, available in source form for evaluation**
Monte Carlo scale like the NAS “EP” benchmark

- No acceleration: 200M samples, 79 seconds
- 1 Advance card: 200M samples, 3.6 seconds
- 5 Advance cards: 200M samples, 0.7 seconds
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:
\[ 1.0000\times10^8 + 1 = 1.0000\times10^8 \]

Double precision:
\[ 1.0000\times10^8 + 1 = 1.000000001\times10^8 \]
GAUSSIAN acceleration

• Dr. Michael Frisch, GAUSSIAN president, estimates DGEMM content at 20% to 90%
• GAUSSIAN already supports dynamic linking, so acceleration could be automatic. No changes to license or to any source code.
• Project underway to measure DGEMM content in the 300 test cases in GAUSSIAN’s test suite
• Potential speedups range from about 10% to 300% depending on how GAUSSIAN is used
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 $100,000/year.
- ClearSpeed card would be cost-effective even with a 7% performance boost. Actual performance boost can be considerably greater than 7%!
Accelerating sparse solvers: ANSYS & LS-DYNA

- Potentially pure plug-and-play
- No added license fee
- Demands ClearSpeed’s 64-bit precision and speed
- Enabled by recent DGEMM improvements; still needs symmetric $A^T A$ variant
- Could enable Computational Fluid Dynamics acceleration (for codes based on finite elements)

10 million degrees of freedom (sparse) becomes…

50,000 dense equations

Accelerator can solve at over 50 GFLOPS
New AMBER capability

- Newton-Raphson refinement now possible; large DGEMM calls from computed second derivatives will be in AMBER 10
- 2.5x speedup obtained for this operation in three hours of programmer effort
- Enables accurate computation of entropy and Gibbs Free Energy for first time.
- Available now in NAB (Nucleic Acid Builder) code.
- AMBER itself has cases that ClearSpeed accelerates by 2.6x to 6x, with 5x to 10x possible once we exploit symmetry of atom-atom interactions
Assume a range of accelerator applicability

- While high speedup applications motivate use of cards, they then have a broad benefit for other uses with less speedup.
- In many cases, the job mix will result in a net performance/price benefit when total cost of ownership is taken into account.

**Net Benefit Factor \( \approx 2.9x \)**

**Fraction of Technical Application Workload**

- 5%: 8x
- 12%: 6.2x
- 8%: 4x
- 33%: 2.8x
- 11%: 1.1x
- 31%: 1.0x
Accelerators designed for HPC applications can improve performance as well as performance per (watt, cabinet, dollar). But applicability is very specific, so be careful.

Early success in LINPACK is now translating to value for real applications in chemistry, mechanical design, etc.

Accelerators enable larger problems, and thus extend scientific/engineering capability.

Host-card latency and bandwidth are not major barriers to successful use of properly-designed accelerators.
If you know how your workload differs from the general one, you can exploit that difference by applying the right accelerator technology.

Accelerators are making HPC fun again!

It’s a tired metaphor, but watch out for this: