ClearSpeed™

64-Bit Floating-Point Accelerators for HPC Applications

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

Outline

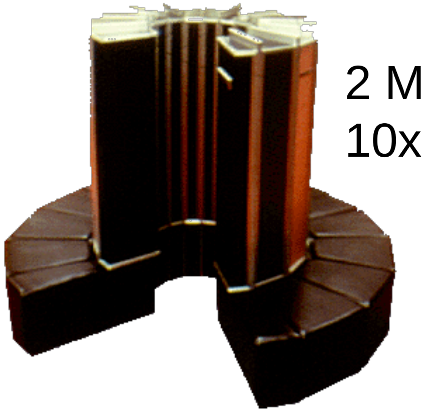
- **Acceleration issues in general: host-accelerator bandwidth/latency**
- **Accelerator performance analysis examples**
- **Measured and expected application acceleration:**
  - Molecular Dynamics: AMBER and NAB
  - Quantum Chemistry: GAUSSIAN, Qbox, PARATEC
  - Monte Carlo models for PDEs
  - LS-DYNA 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

3 MB/s



2 MB
10x speedup

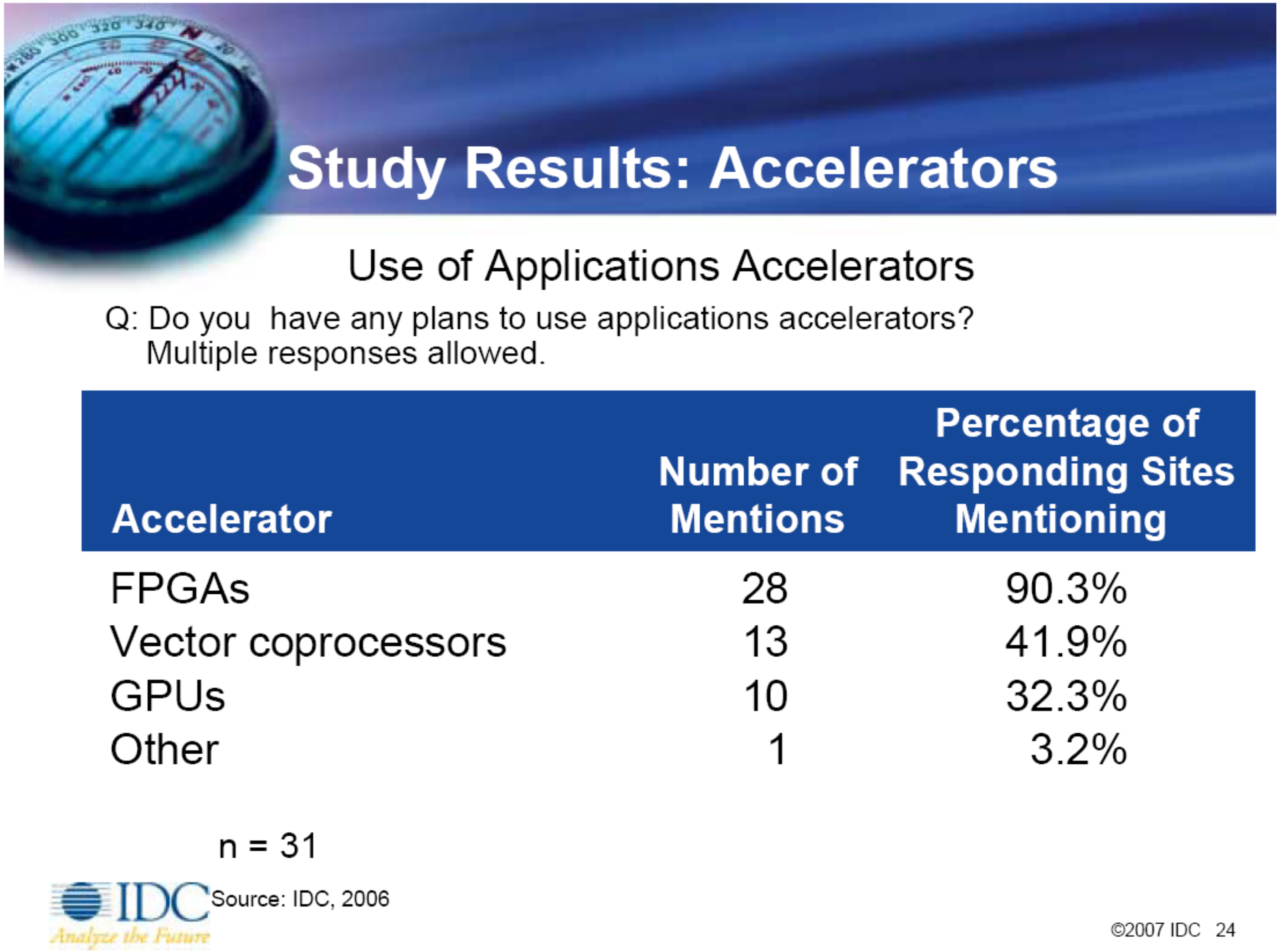
General-purpose computer
Runs OS, compilers, disk, printers, user interface

Attached vector processor accelerates certain applications, but not all

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*

IDC survey of planned HPC accelerator use



Study Results: Accelerators

Use of Applications Accelerators

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

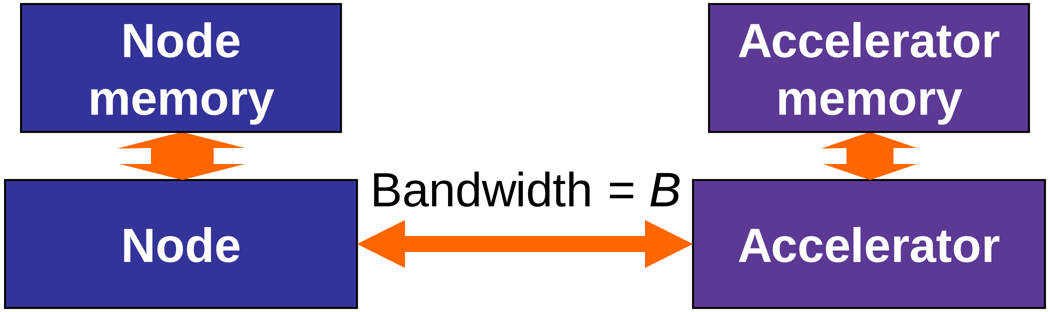
Accelerator	Number of Mentions	Percentage of Responding Sites Mentioning
FPGAs	28	90.3%
Vector coprocessors	13	41.9%
GPUs	10	32.3%
Other	1	3.2%

n = 31

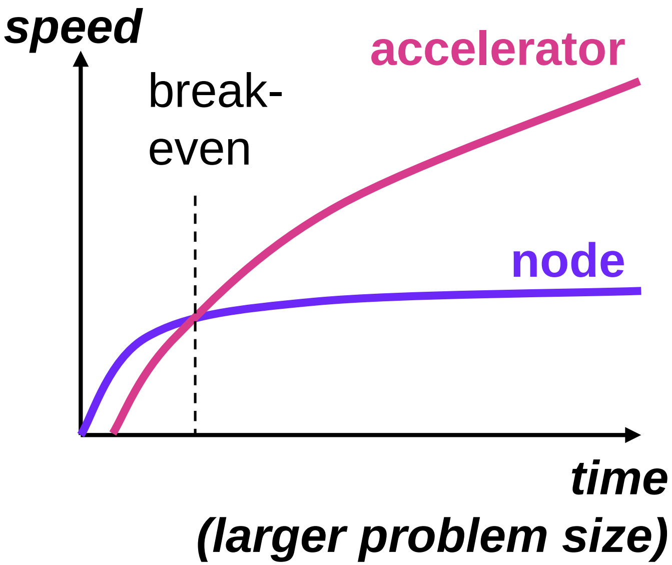
IDC Analyze the Future Source: IDC, 2006

©2007 IDC 24

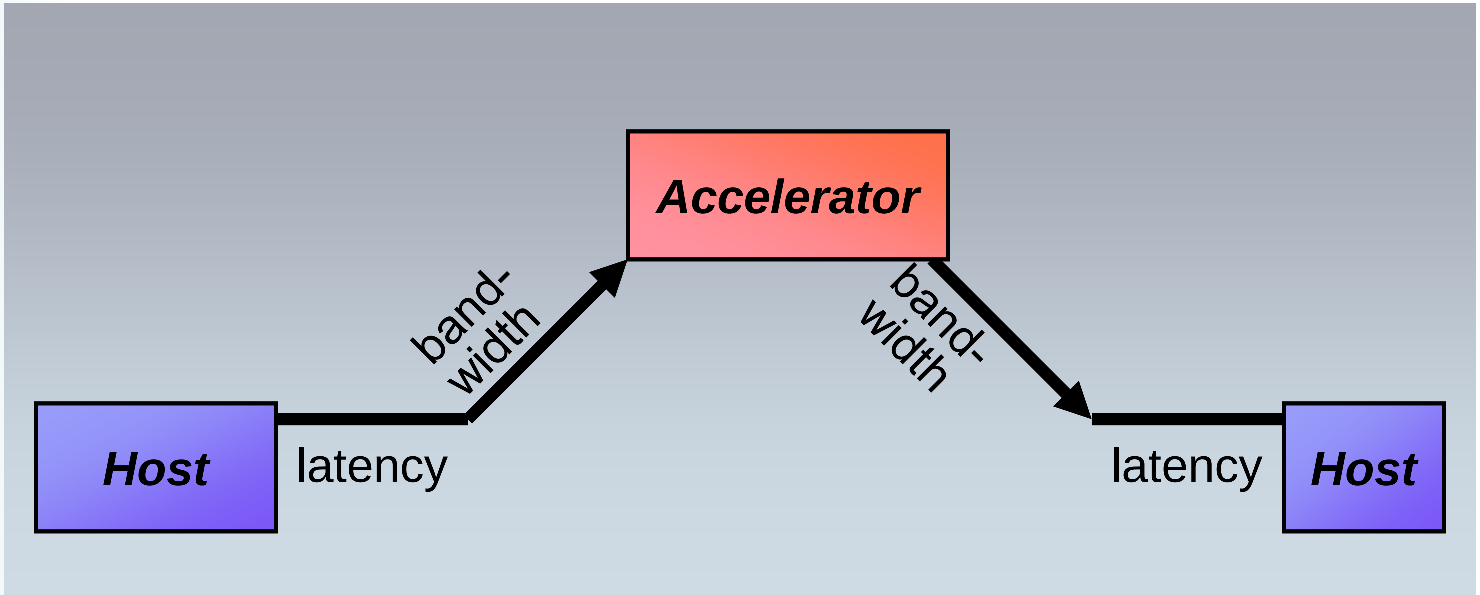
Is this trip necessary? Bandwidth issues



- 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…*

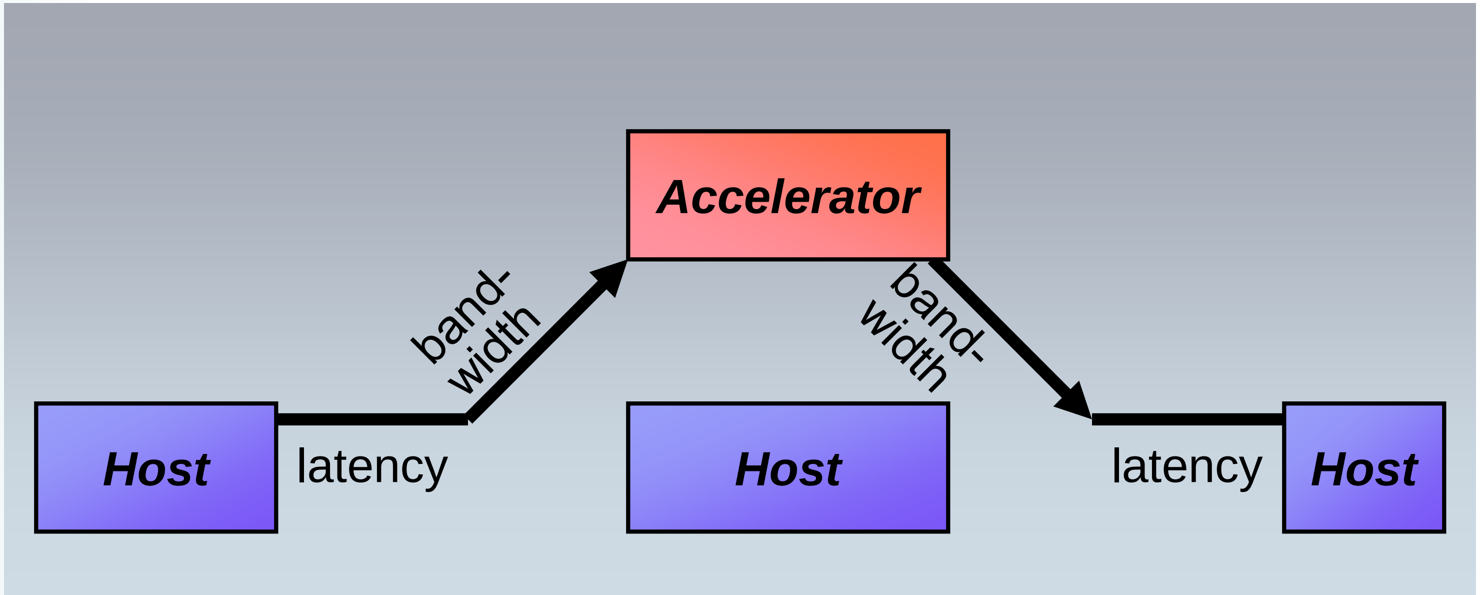


Simple offload model is out of date



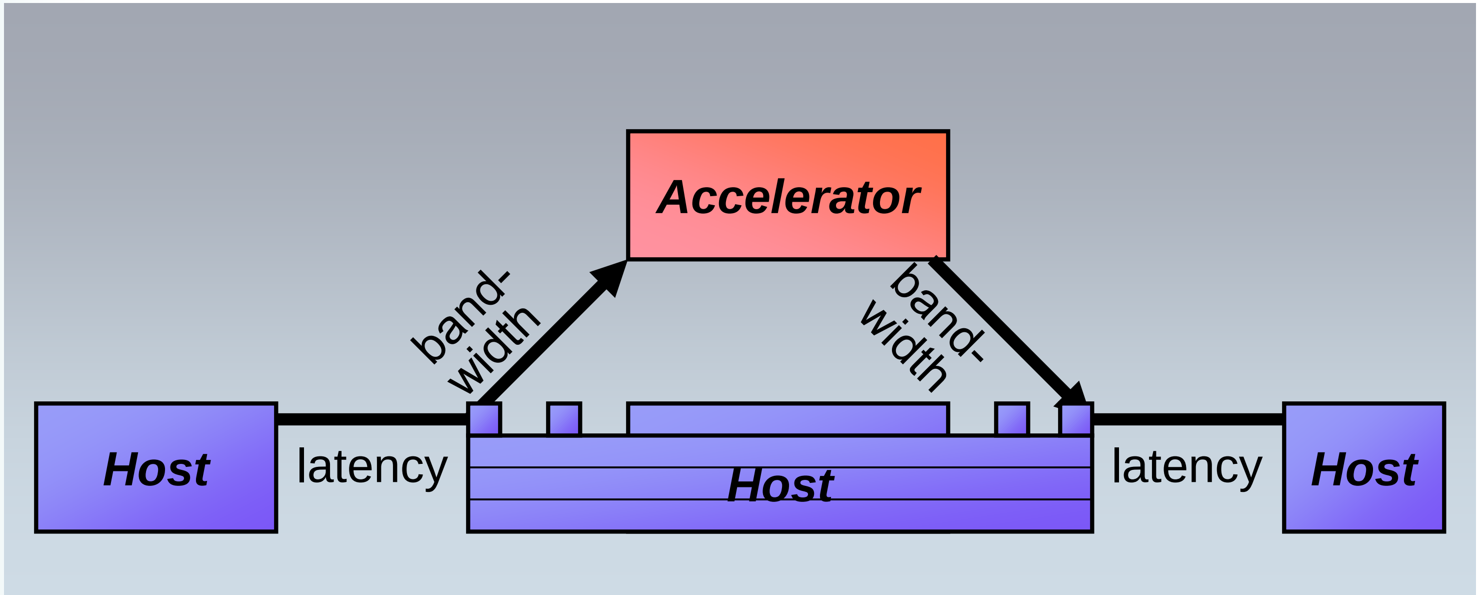
- **Accelerator must be quite fast for this approach to have benefit**
- **This “mental picture” may stem from early days of Intel 80x87, Motorola 6888x math coprocessors**

Acceleration model: Host continues working



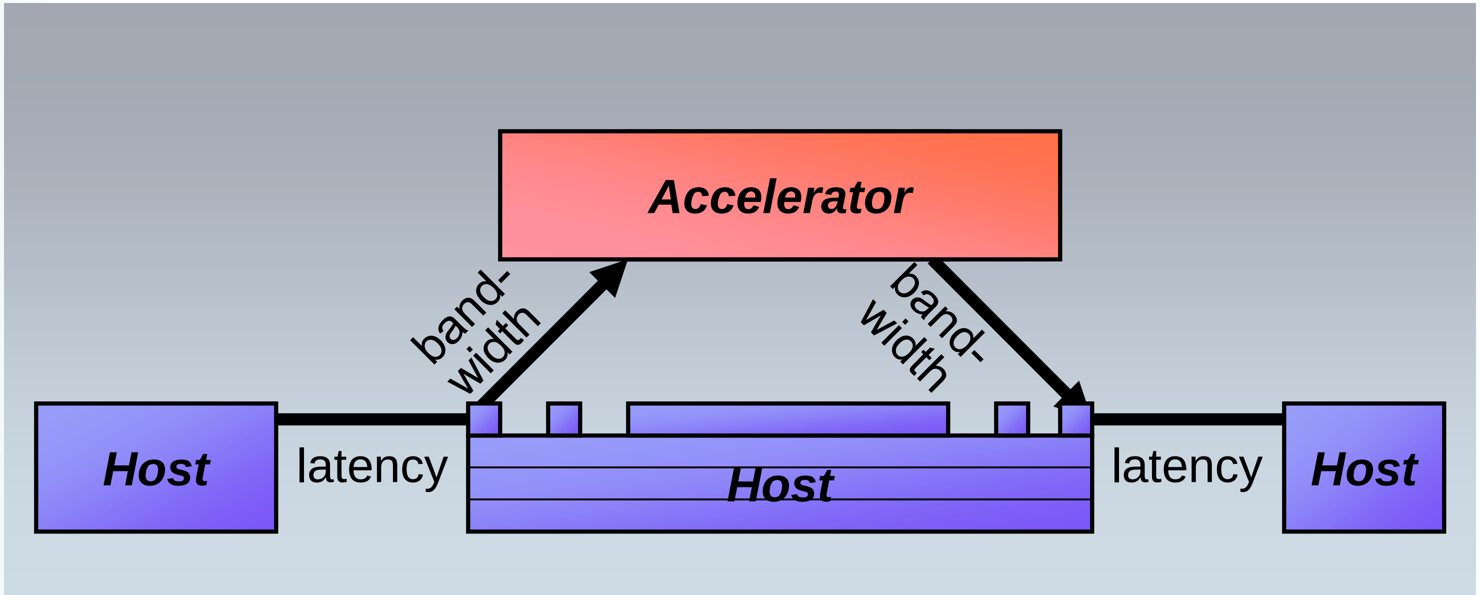
- Accelerator needs only be fast enough to make up for time lost to bandwidth + latency
- Easiest use model: host and accelerator share the same task, load balanced to complete at same time
- More flexible: Host, accelerator *specialize* what they do

Host can work while data is moved



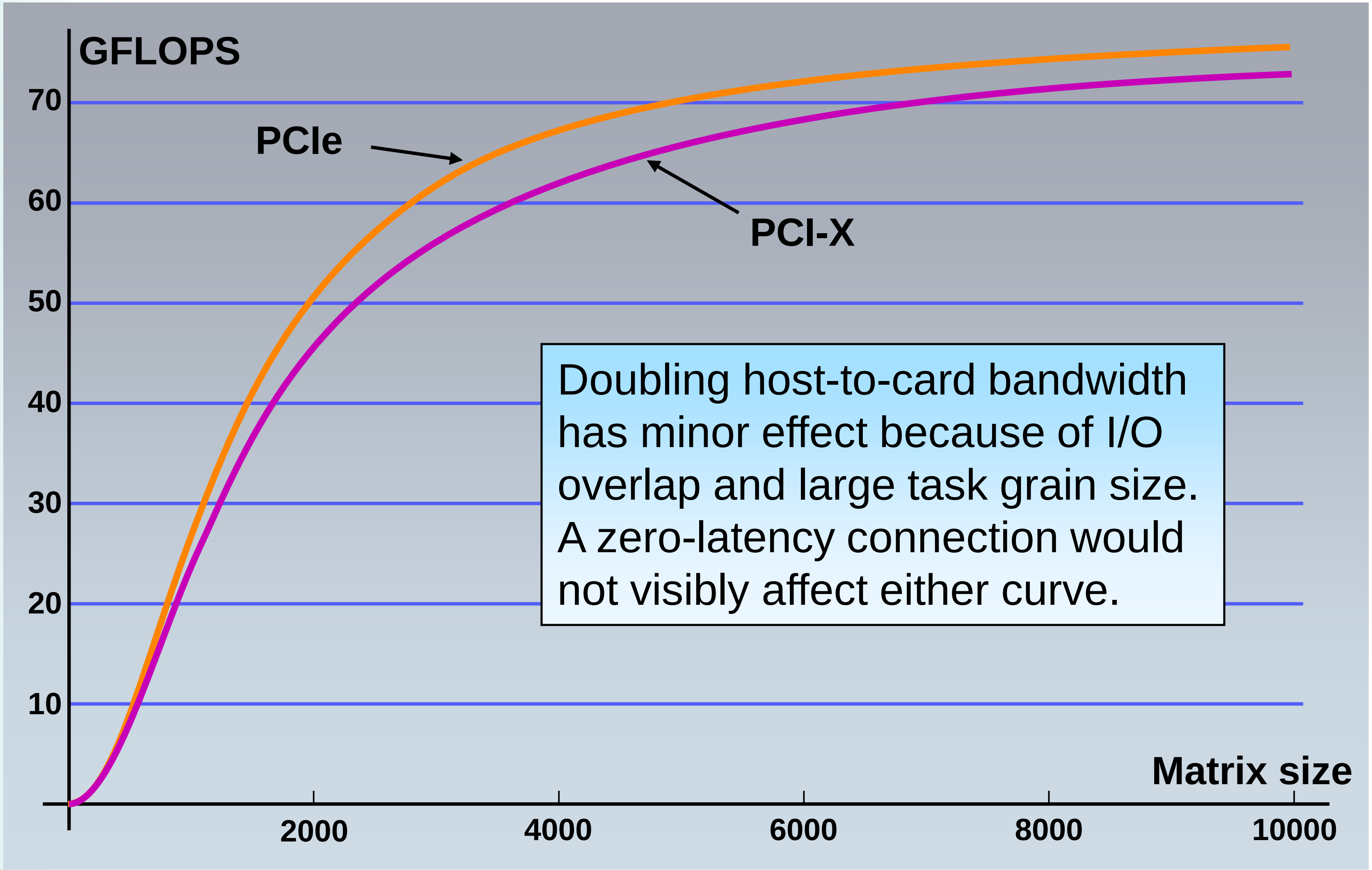
- PCI transfers might burden a *single x86 core* by 60%
- Other cores on host continue productive work at full speed

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!

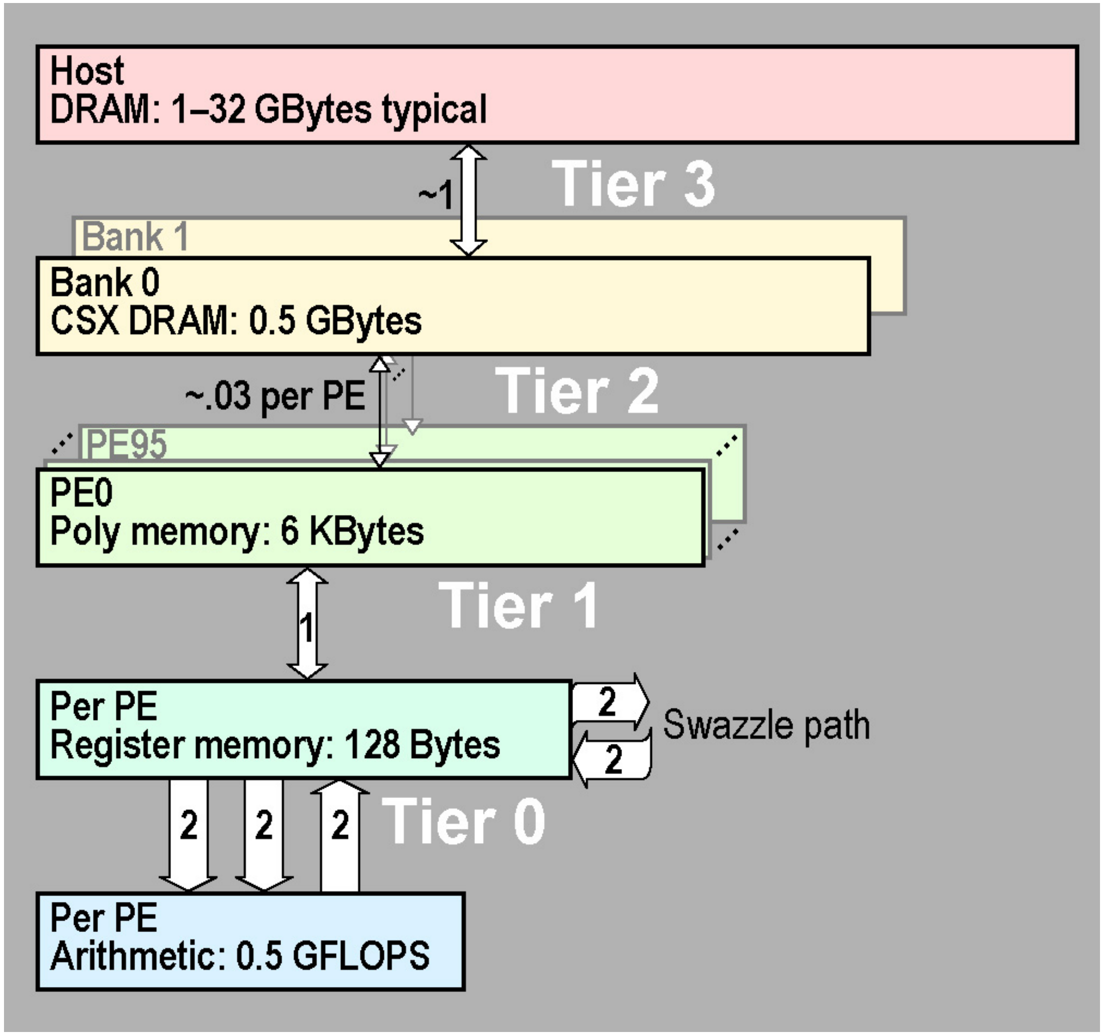
Square DGEMM speeds as of December 2006



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



Total: 1.0 GB

Total: 6.4 GB/s

Total: 1.1 MB

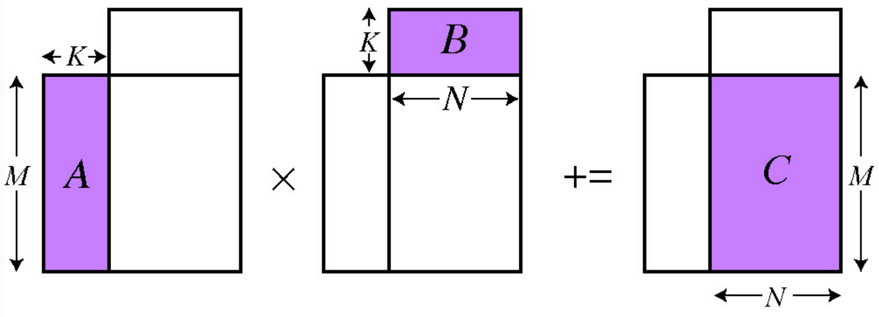
Total: 192 GB/s

Total: 24 KB

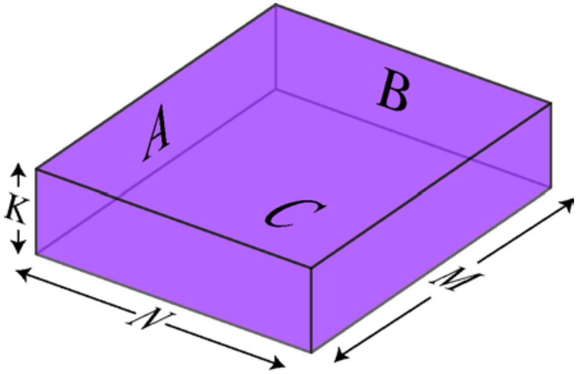
Total: 2 TB/s

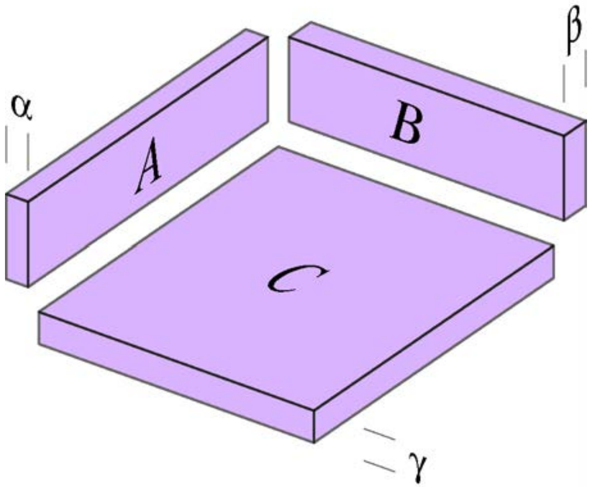
Total: 96 GFLOPS
(but only 25 watts)

Visualization of algorithm overhead: DGEMM



Matrix multiply (DGEMM) is a perfect analog to a folded box.

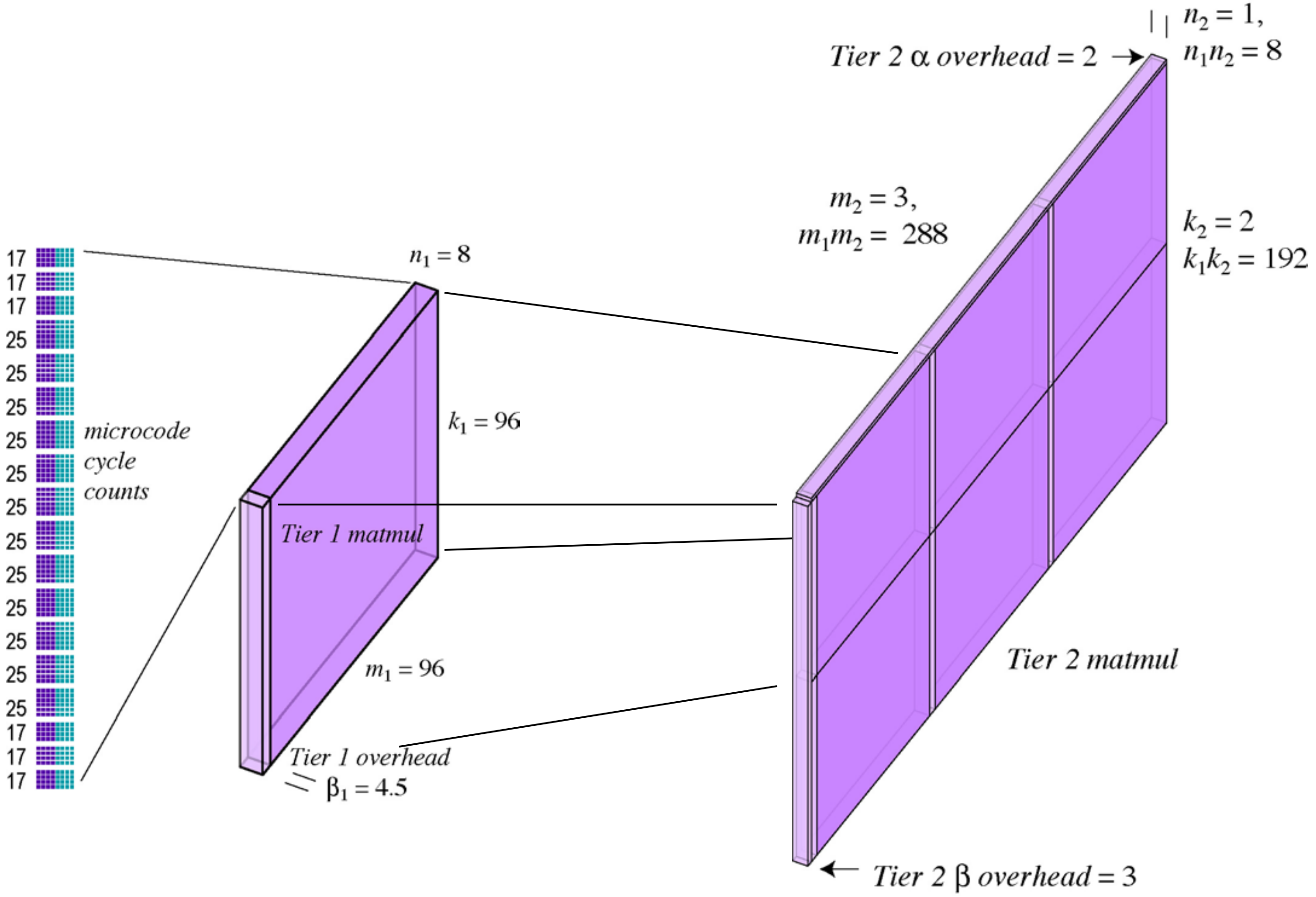




Volume is the number of multiply-adds.

Surface "padding" shows overheads

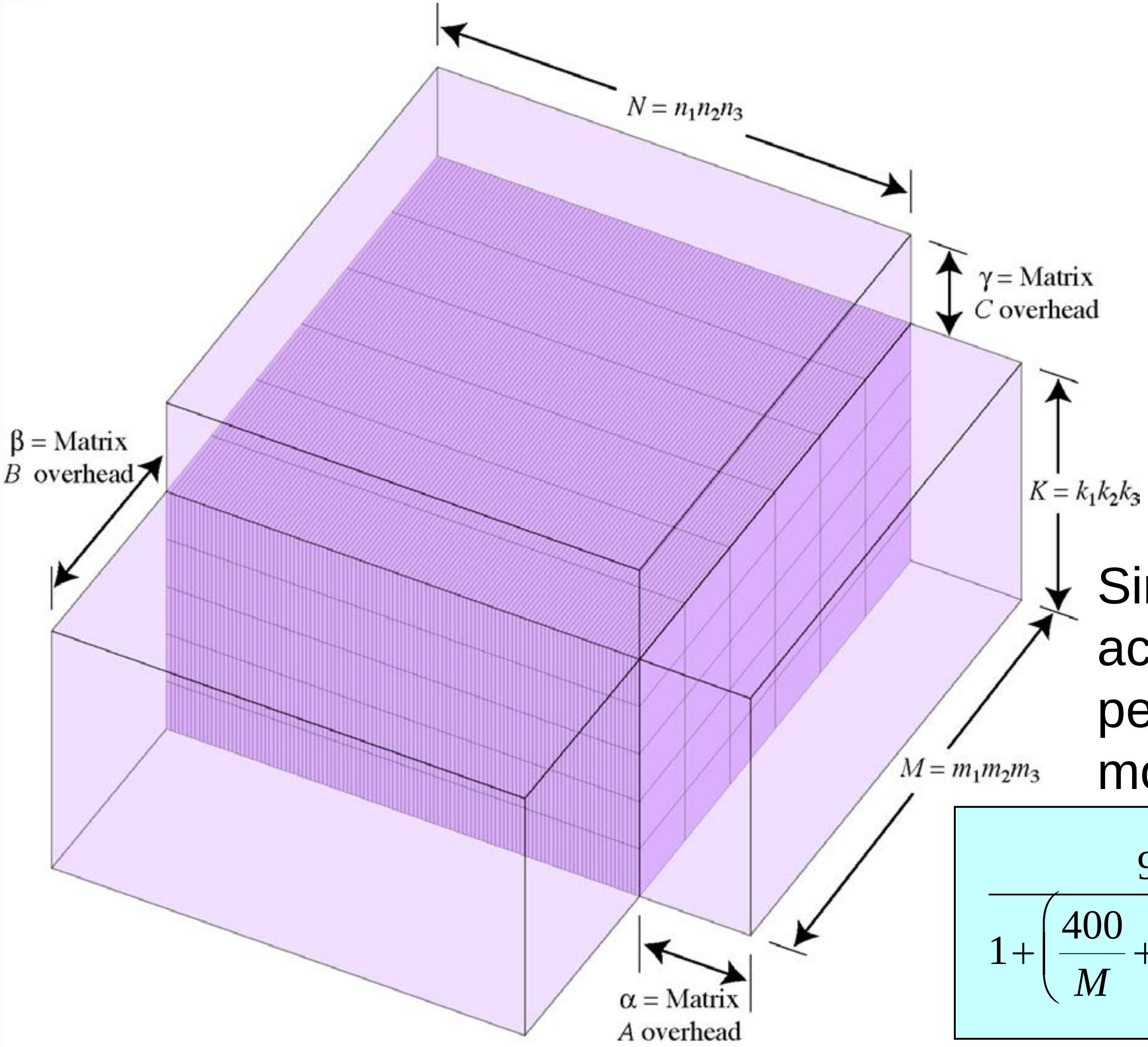
Build up the bricks through the hierarchy



Tier 1

Tier 2

Tier 3 overhead was originally harder to overlap



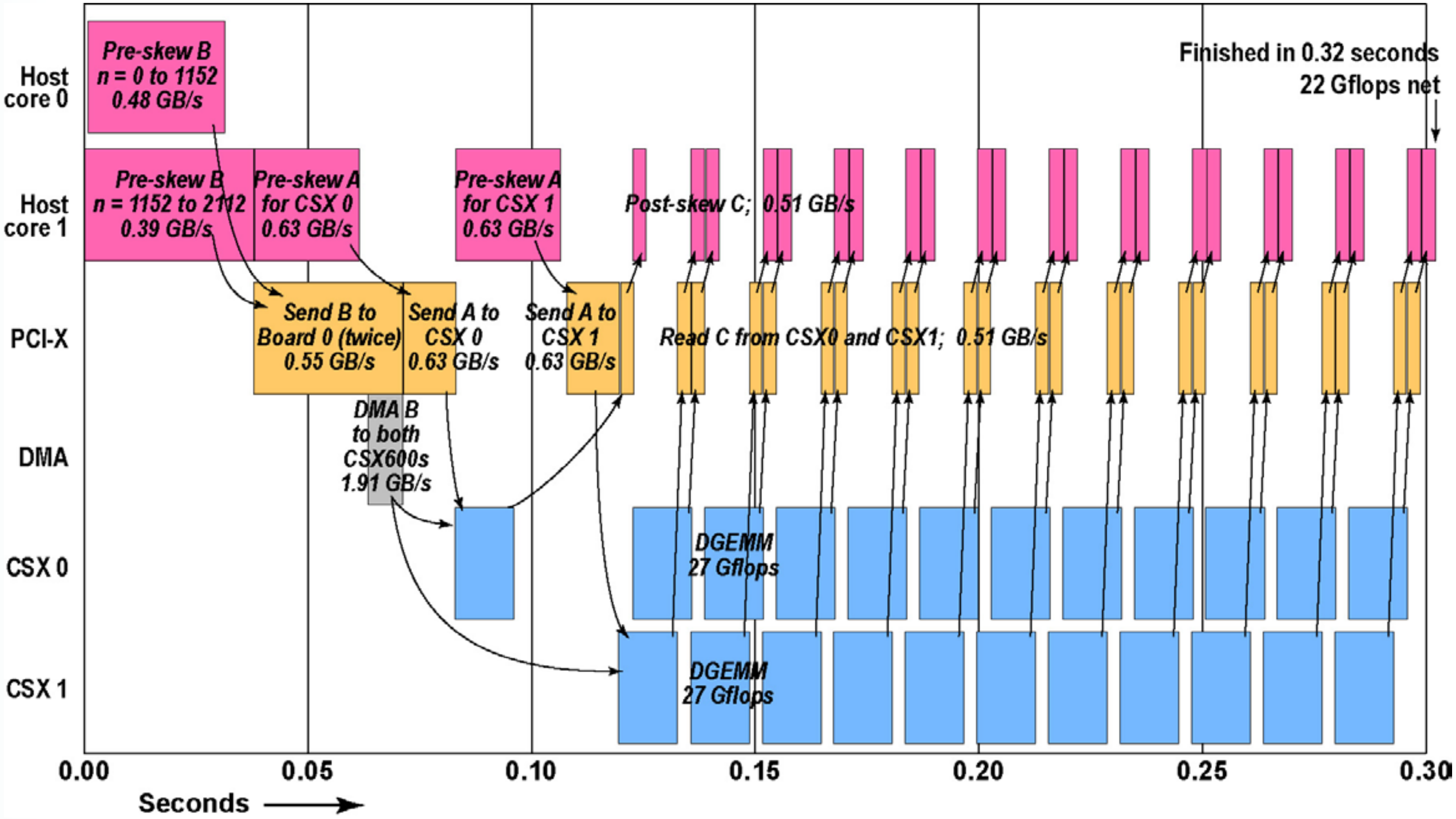
Simple but decent accelerator performance model:

$$\frac{90.2}{1+\left(\frac{400}{M}+\frac{200}{K}+\frac{200}{N}\right)} \text{ GFLOPS.}$$

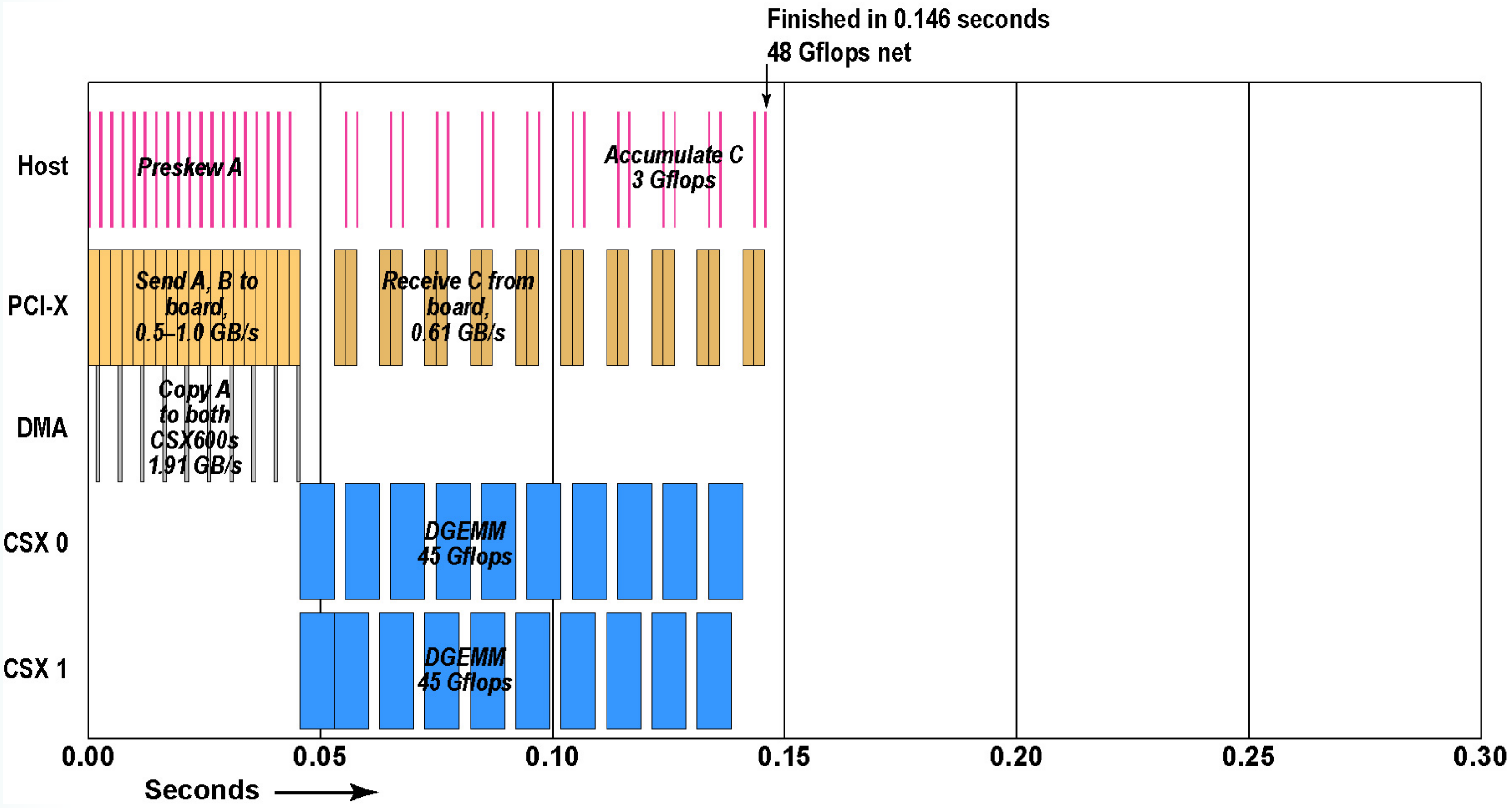
ClearSpeed™

Unoptimized Tier 3 timing

K=960, *M*=1920, *N*=1920

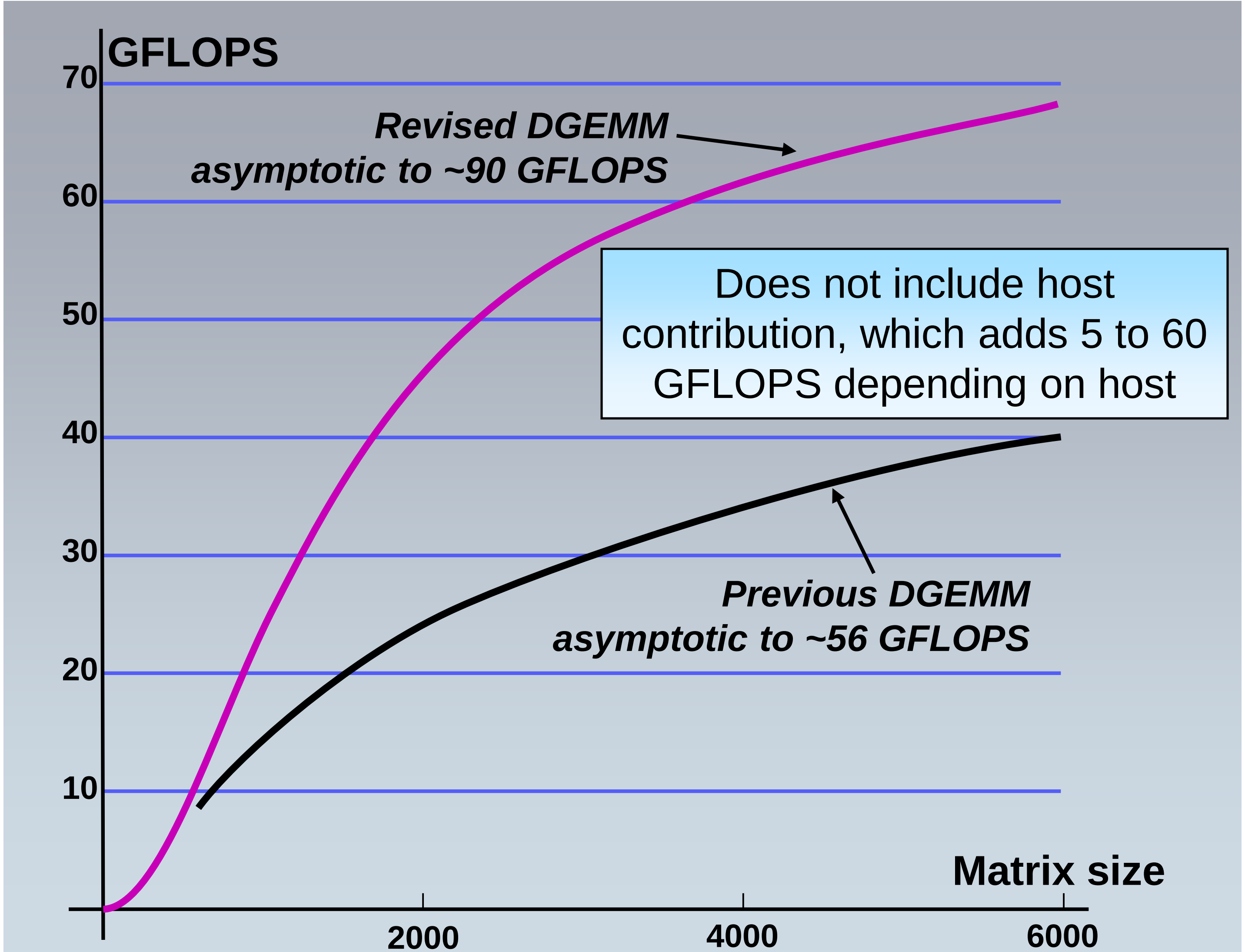


Optimized Tier 3 timing



Can now accelerate matrices as small as N=576

Almost doubled sustained speed



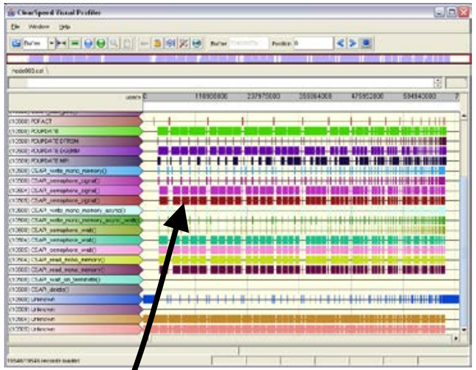
Does not include host contribution, which adds 5 to 60 GFLOPS depending on host

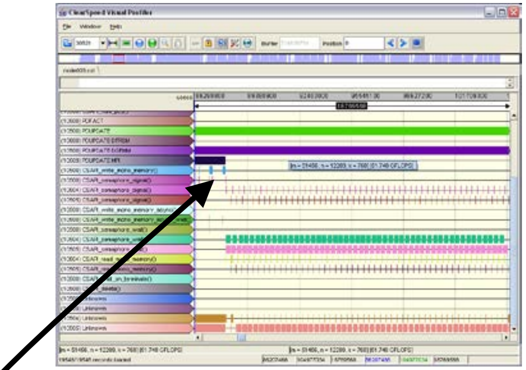
Note: curve only samples integer multiples of vector size

Detailed profiling is essential for accelerator tuning

HOST CODE PROFILING

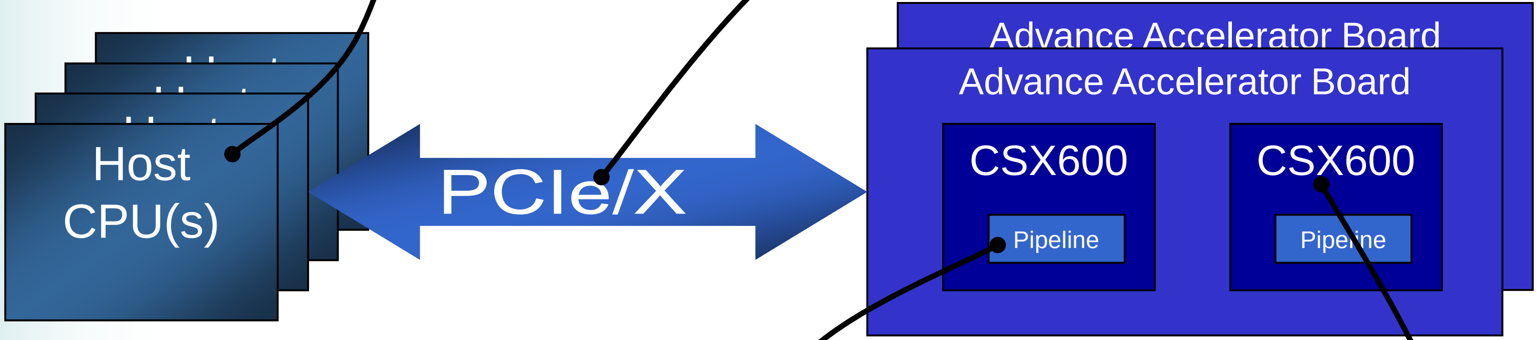
Visually inspect multiple host threads.
Time specific code sections.
Check overlap of host threads





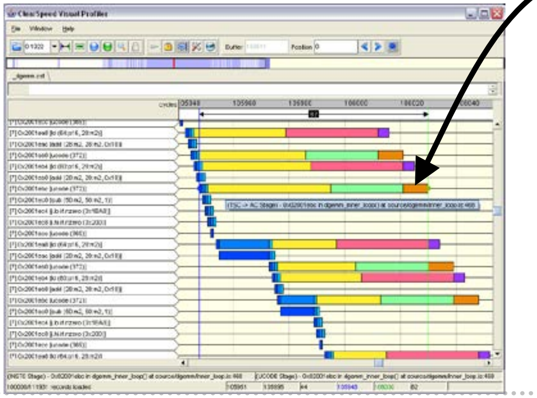
HOST/BOARD INTERACTION

View host/board interactions
Measure transfer bandwidth.
Check overlap of host and board compute



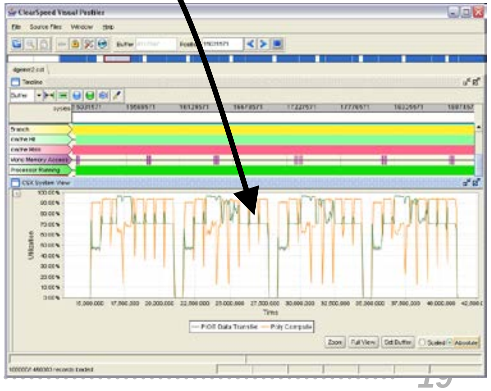
ACCELERATOR PIPE

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

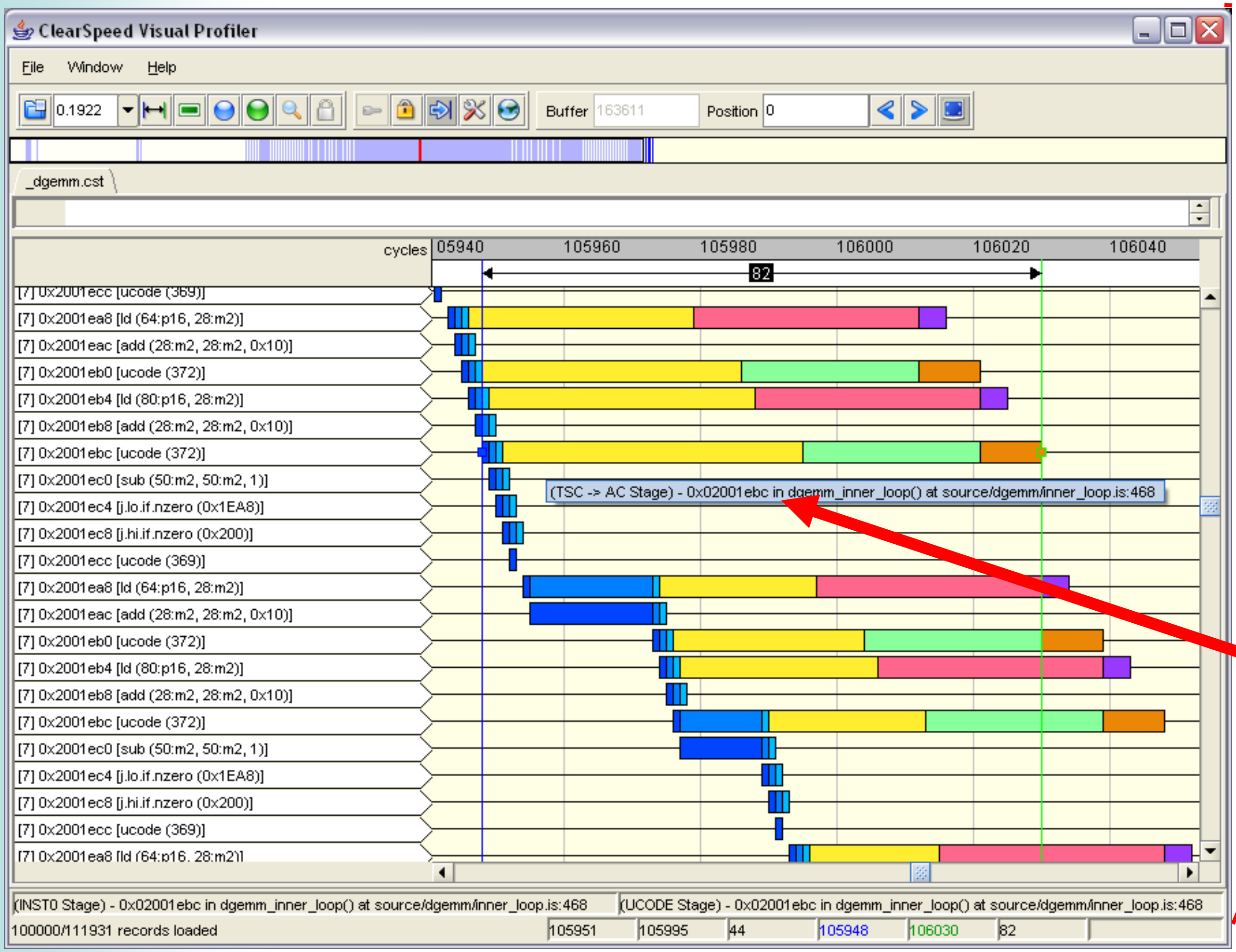


CSX600 SYSTEM

Trace at system level.
Inspect overlap of compute and I/O.
View cache utilization.
Graph performance.



Pipeline view of DGEMM inner loop

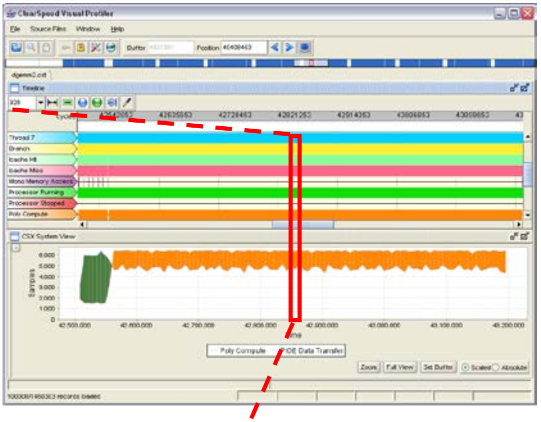


ClearSpeed Visual Profiler

```
[7] 0x2001ecc [ucode (369)]
[7] 0x2001ea8 [ld (64:p16, 28:m2)]
[7] 0x2001eac [add (28:m2, 28:m2, 0x10)]
[7] 0x2001eb0 [ucode (372)]
[7] 0x2001eb4 [ld (80:p16, 28:m2)]
[7] 0x2001eb8 [add (28:m2, 28:m2, 0x10)]
[7] 0x2001ebc [ucode (372)]
[7] 0x2001ec0 [sub (50:m2, 50:m2, 1)]
[7] 0x2001ec4 [j.lo.if.nzero (0x1EA8)]
[7] 0x2001ec8 [j.hi.if.nzero (0x200)]
[7] 0x2001ecc [ucode (369)]
[7] 0x2001ea8 [ld (64:p16, 28:m2)]
[7] 0x2001eac [add (28:m2, 28:m2, 0x10)]
[7] 0x2001eb0 [ucode (372)]
[7] 0x2001eb4 [ld (80:p16, 28:m2)]
[7] 0x2001eb8 [add (28:m2, 28:m2, 0x10)]
[7] 0x2001ebc [ucode (372)]
[7] 0x2001ec0 [sub (50:m2, 50:m2, 1)]
[7] 0x2001ec4 [j.lo.if.nzero (0x1EA8)]
[7] 0x2001ec8 [j.hi.if.nzero (0x200)]
[7] 0x2001ecc [ucode (369)]
[7] 0x2001ea8 [ld (64:p16, 28:m2)]

(TSC -> AC Stage) - 0x02001ebc in dgemm_inner_loop() at source/dgemm/inner_loop.is:468

(INSTO Stage) - 0x02001ebc in dgemm_inner_loop() at source/dgemm/inner_loop.is:468
(UCODE Stage) - 0x02001ebc in dgemm_inner_loop() at source/dgemm/inner_loop.is:468
100000/111931 records loaded    105951    105995    44    105948    106030    82
```

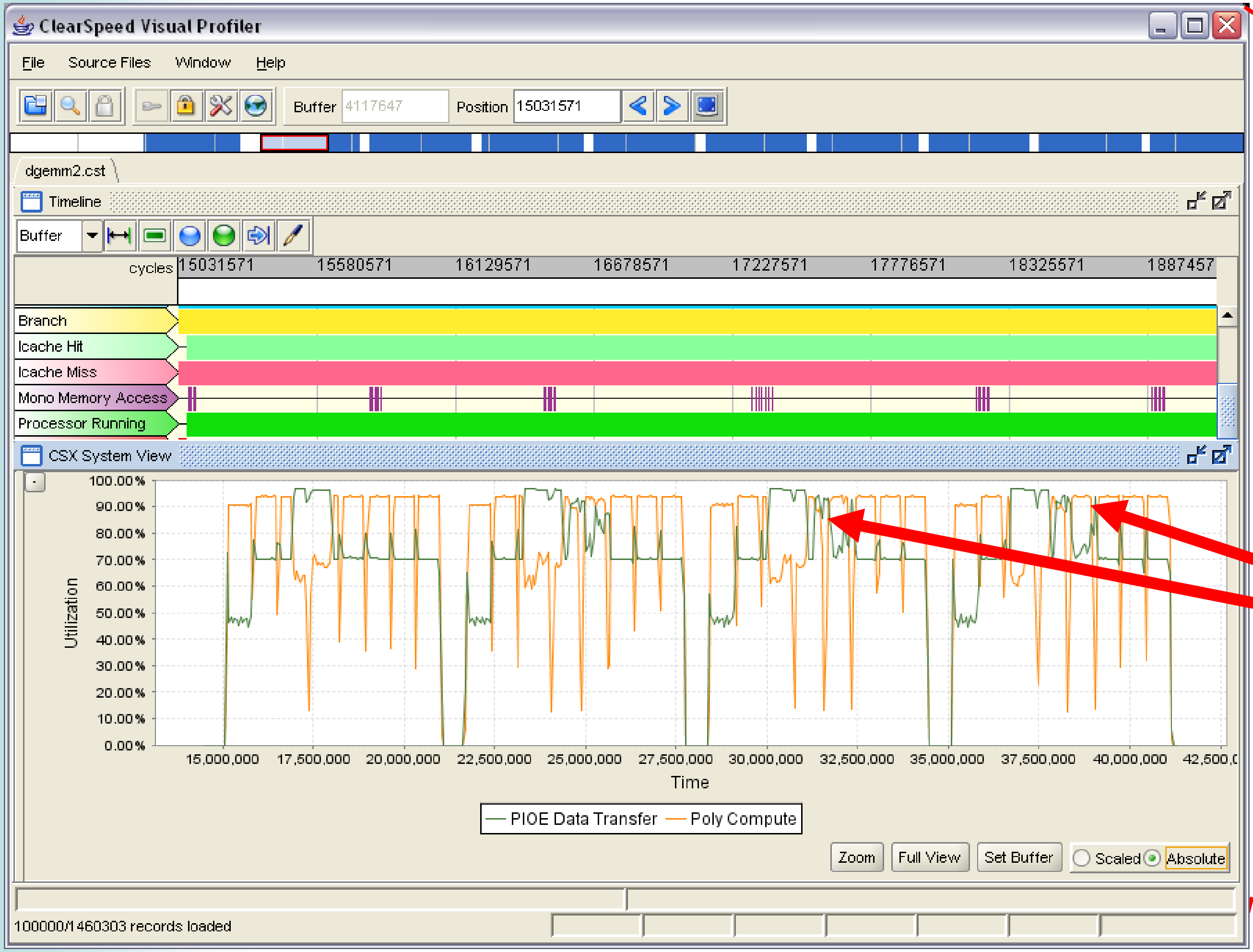


Profile the code running at the instruction level

See the pipeline performance for each instruction

Tune the instruction scheduling for the application code

System level: multiple DGEMM calls

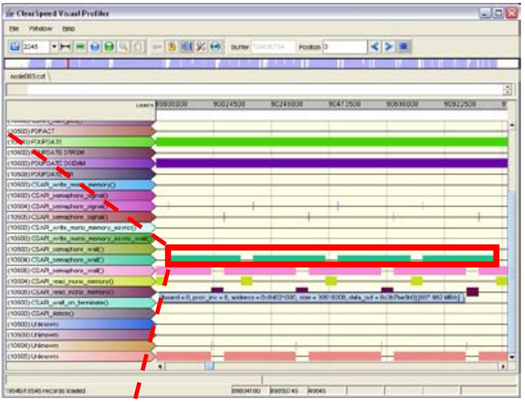


Buffer 4117647 Position 15031571

Buffer

○ Scaled ● Absolute

100000/1460303 records loaded



View the DGEMM calls on host and accelerator

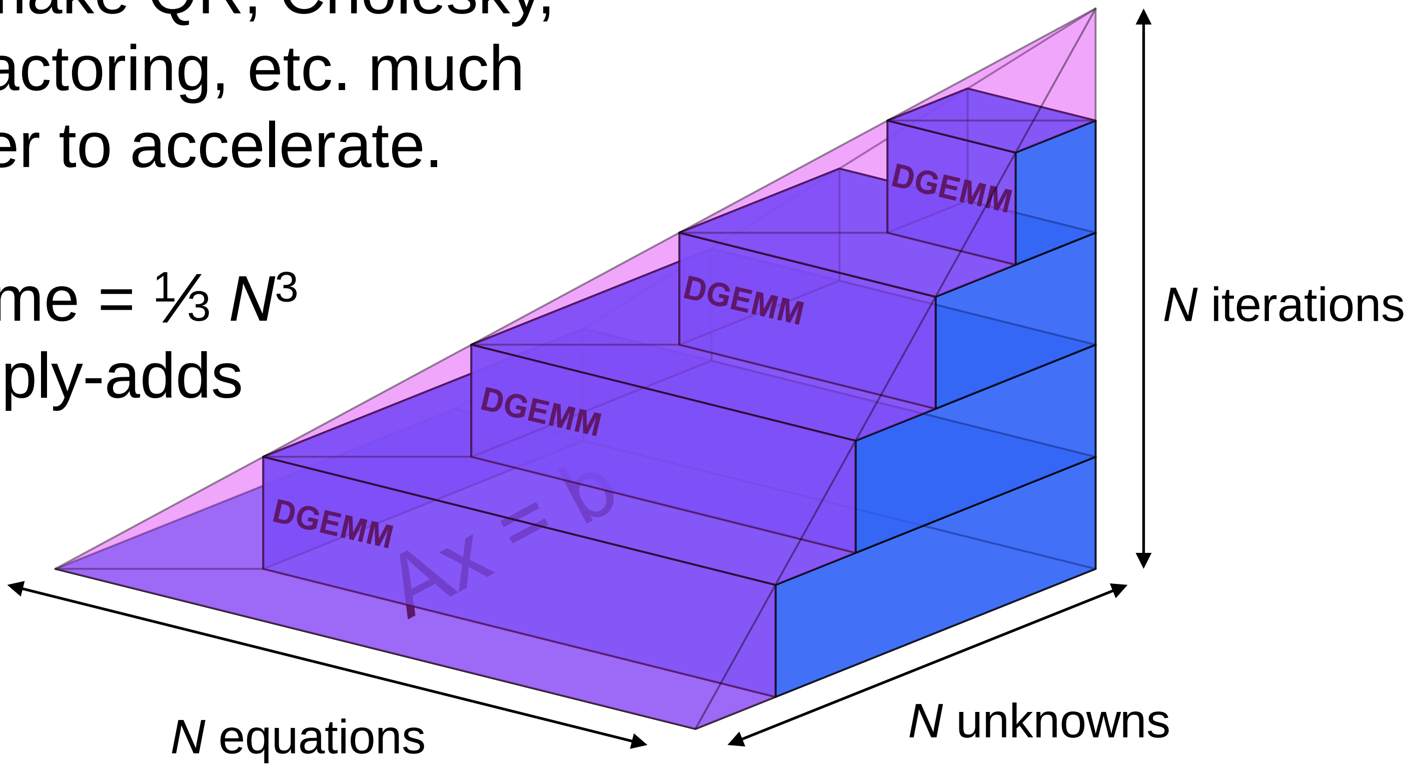
Each call syncs up with the host view of accelerator

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 = ⅓ N^3 multiply-adds



Excellent test of hardware correctness

Accelerated cluster model accurate to about ±5%

$$R_{est} = \frac{1}{\dfrac{1}{PQ\gamma} + \dfrac{3\alpha\left[\left(N_B + 1\right)\left\lceil \lg P \right\rceil + P\right]}{2N^2 N_B} + \dfrac{3\beta\left(3P + Q\right)}{4NPQ}}$$

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)

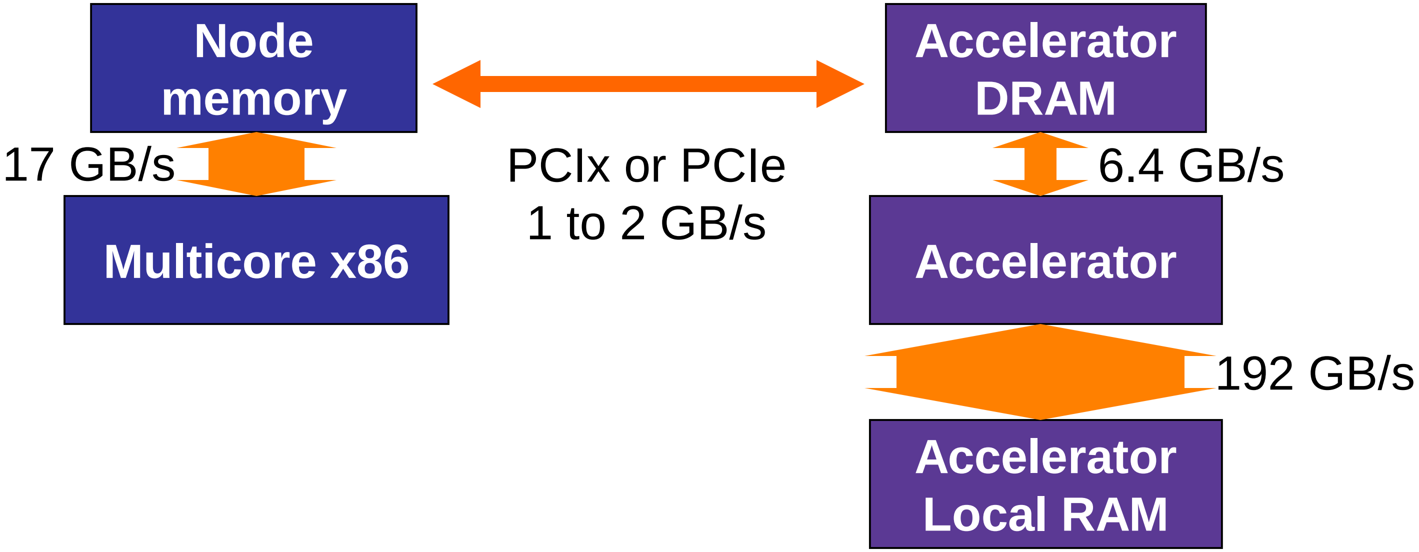
P, Q = Dimensions of the two-dimensional mapping of computational nodes

α = Effective point-to-point latency of MPI broadcast, in seconds

β = Effective point-to-point reciprocal bandwidth of message broadcast, in seconds per datum

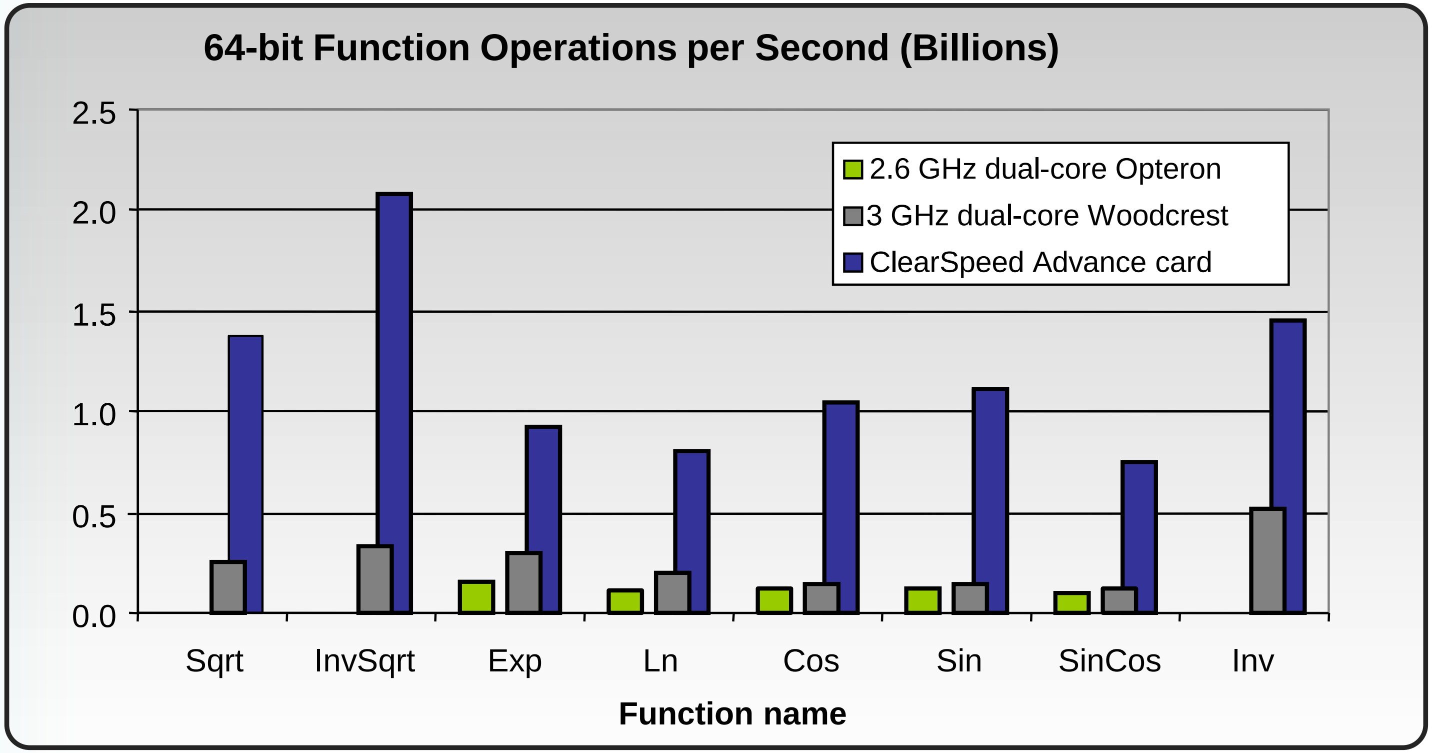
γ = Effective floating-point operations per second of a node independent of MPI operations

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

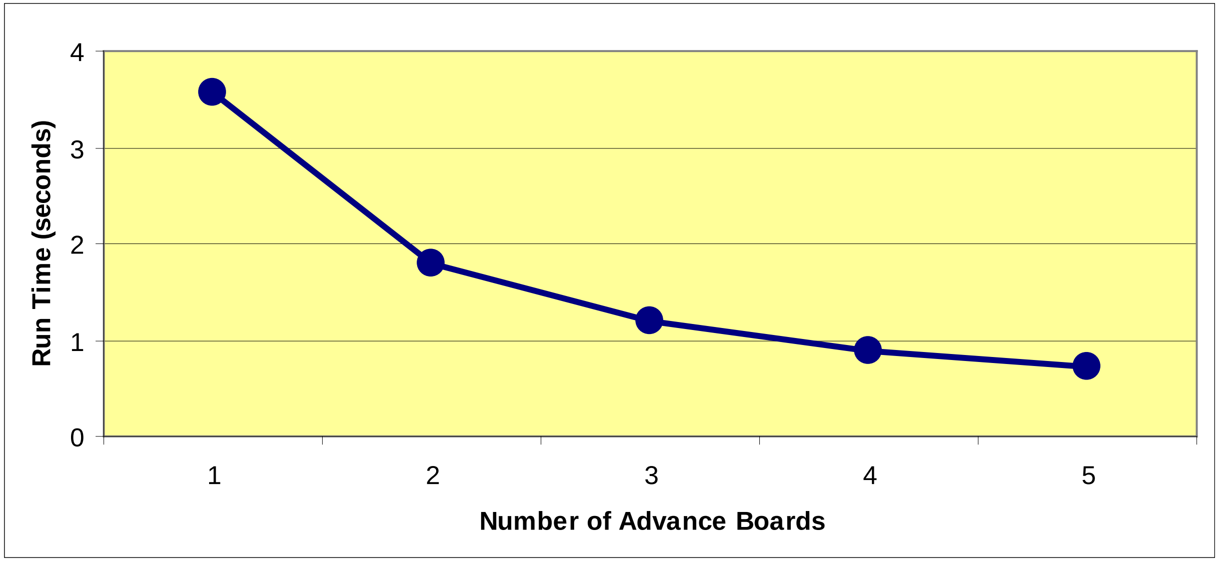


Typical speedup of ~8X over the fastest x86 processors, because math functions stay in the local memory on the card

Monte Carlo PDE methods exploit Tier 1 bandwidth

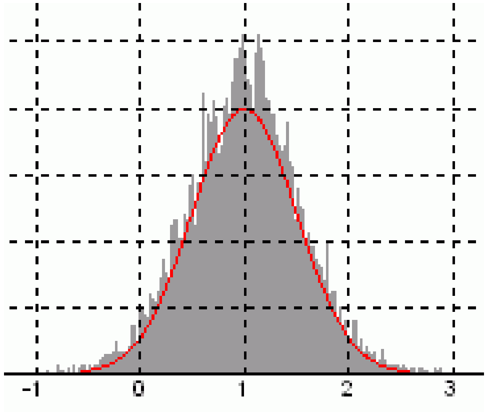
Real apps do work resembling "EP" of NAS Parallel Benchmarks. "Quants" solve PDEs this way for options pricing, Black-Scholes model (a form of the Heat Equation)

- **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!**

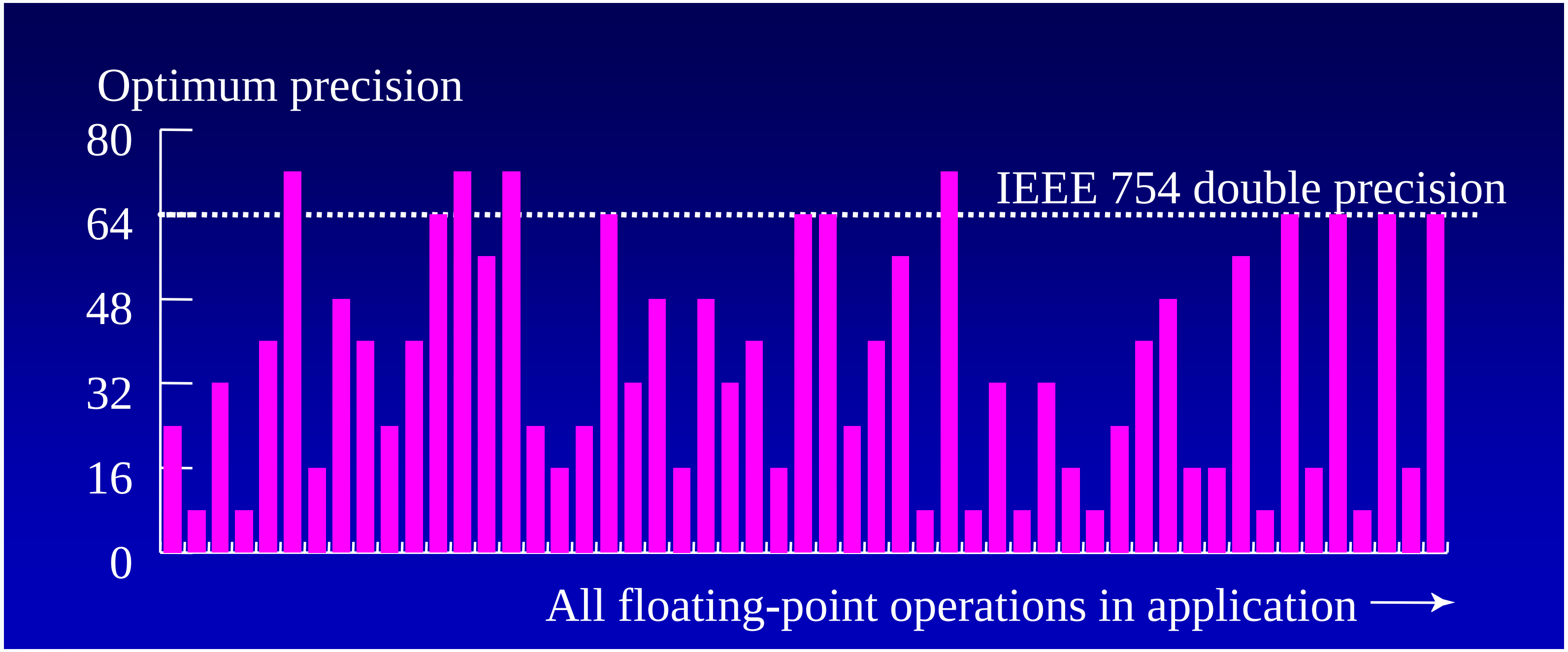


Single precision:
$1.0000 \times 10^8 + 1$
$= 1.0000 \times 10^8$

Double precision:
$1.0000 \times 10^8 + 1$
$= 1.00000001 \times 10^8$

We may need to rethink 64-bit flops…

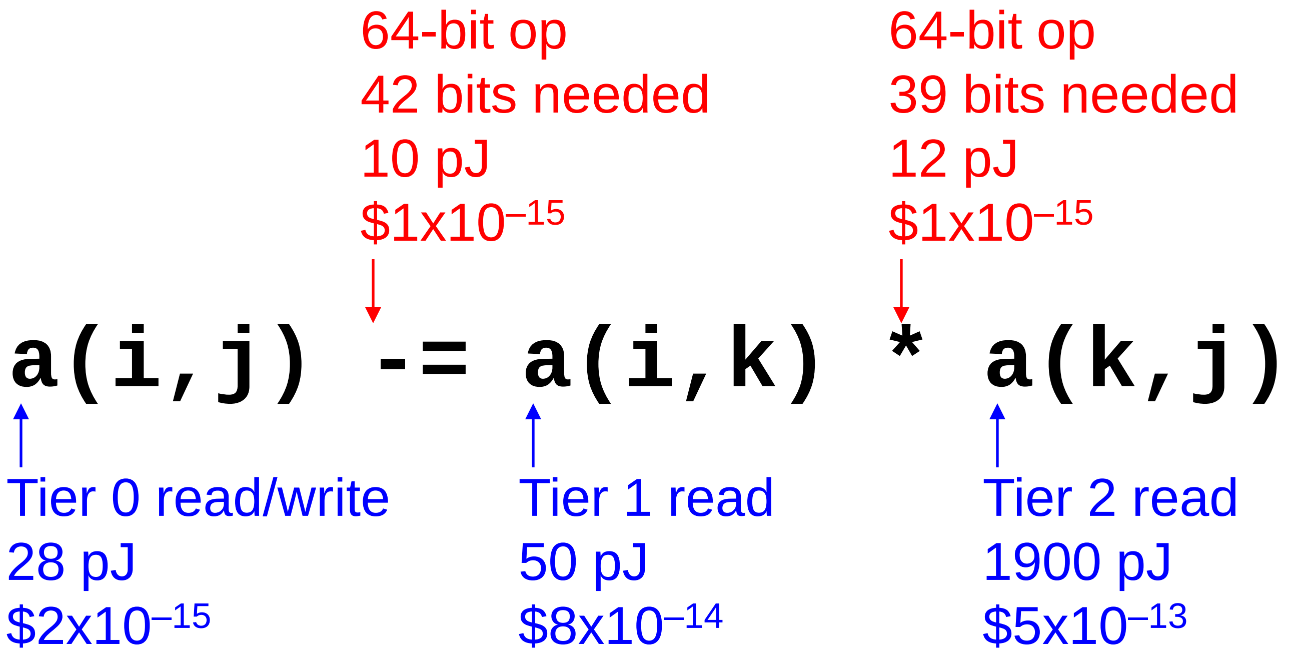
- **Every operation has an optimum number of bits of accuracy**
  - Using too few gives unacceptable errors
  - Using too many wastes memory, bandwidth, joules, dollars.
- **It is unlikely that a code uses *just the right amount* of precision needed.**



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?



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

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

```
a(i,j) -= a(i,k) * a(k,j)
```

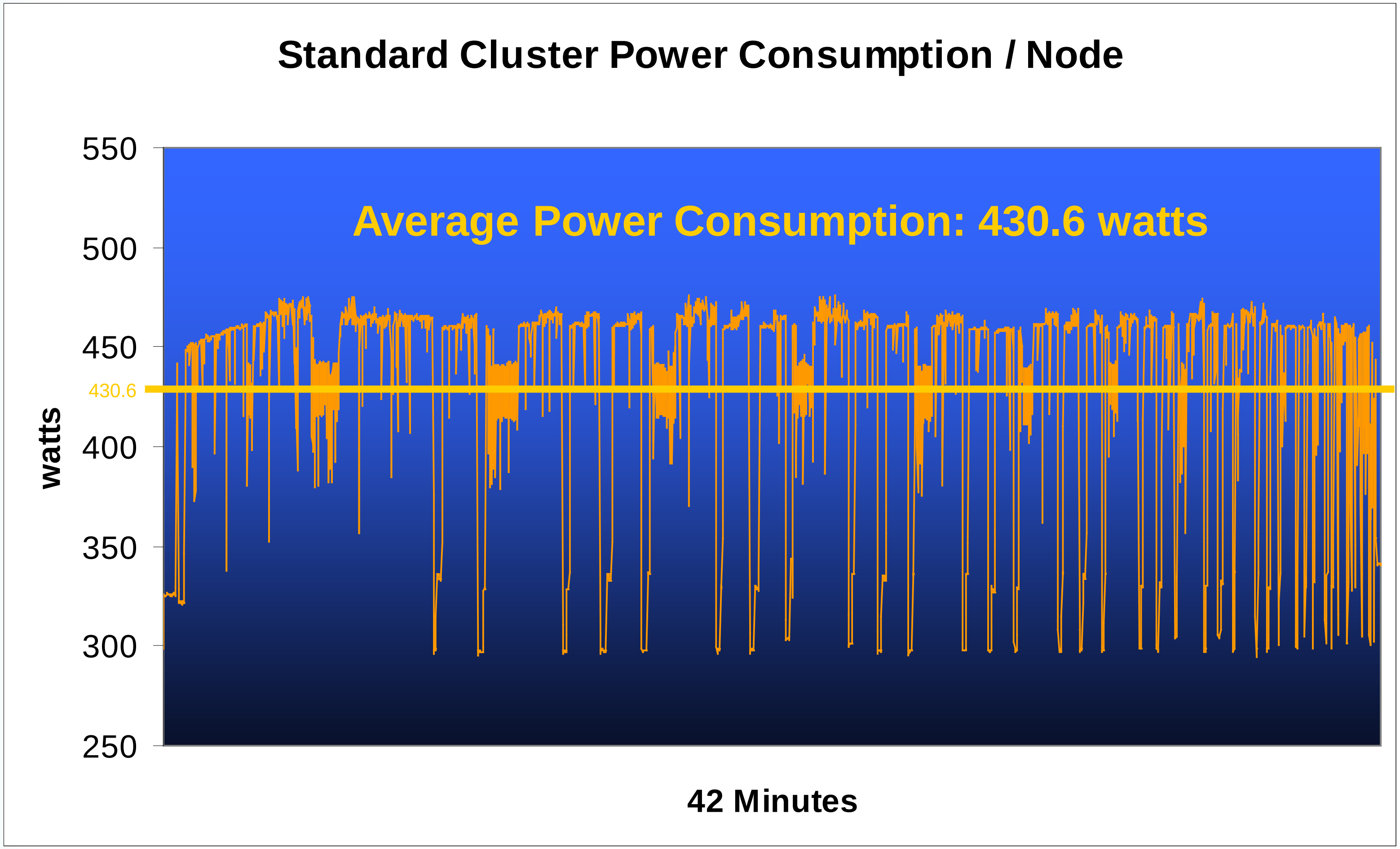
Tier 0 read/write
28 pJ
$\$2\text{x}10^{-15}$

Tier 1 read
50 pJ
$\$8\text{x}10^{-14}$

Tier 2 read
1900 pJ
$\$5\text{x}10^{-13}$

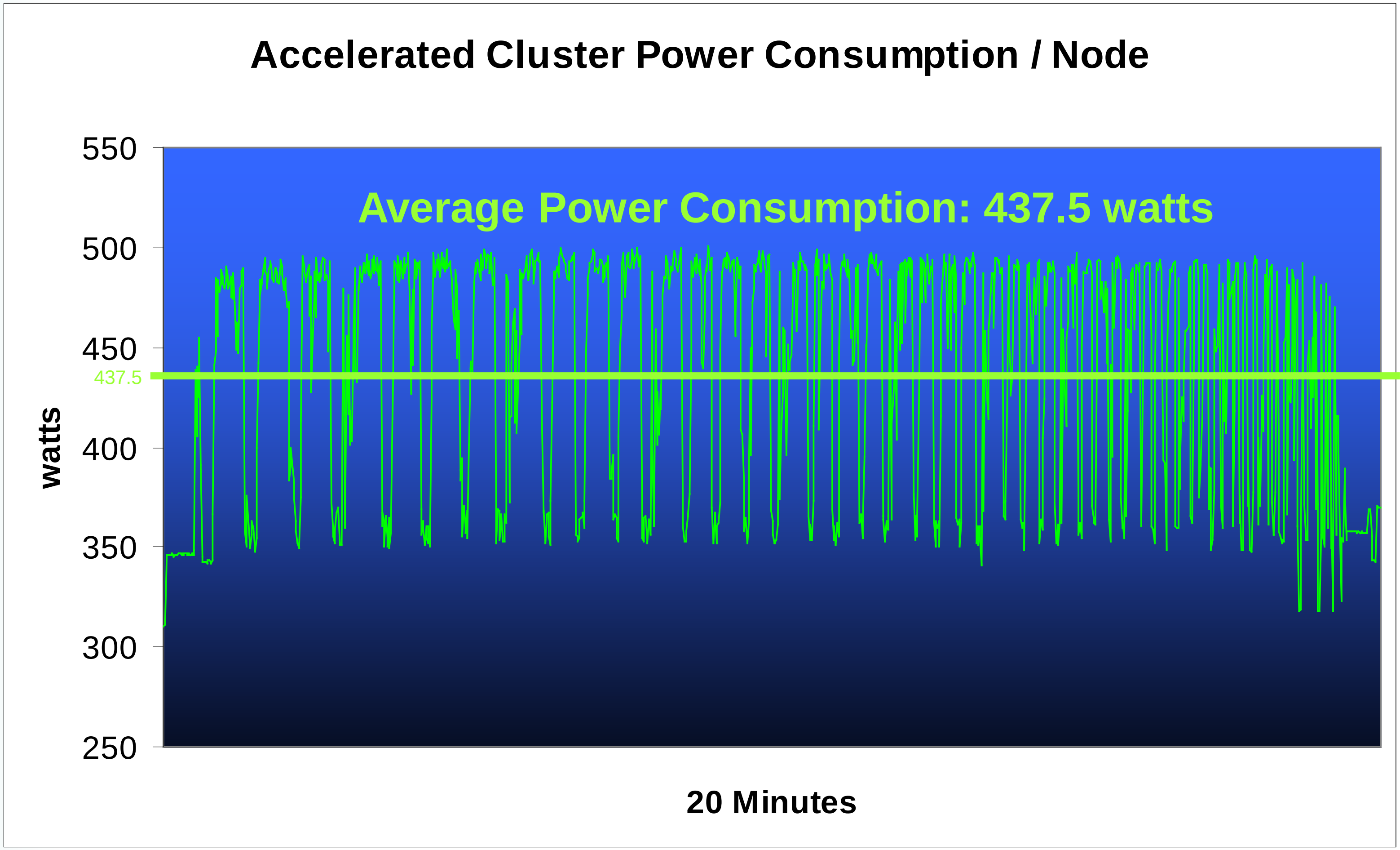
Cost and electrical power and precision are almost as important as timing… why not develop analysis tools for them? You can only optimize what you can measure.

Model/measure power use, not just performance



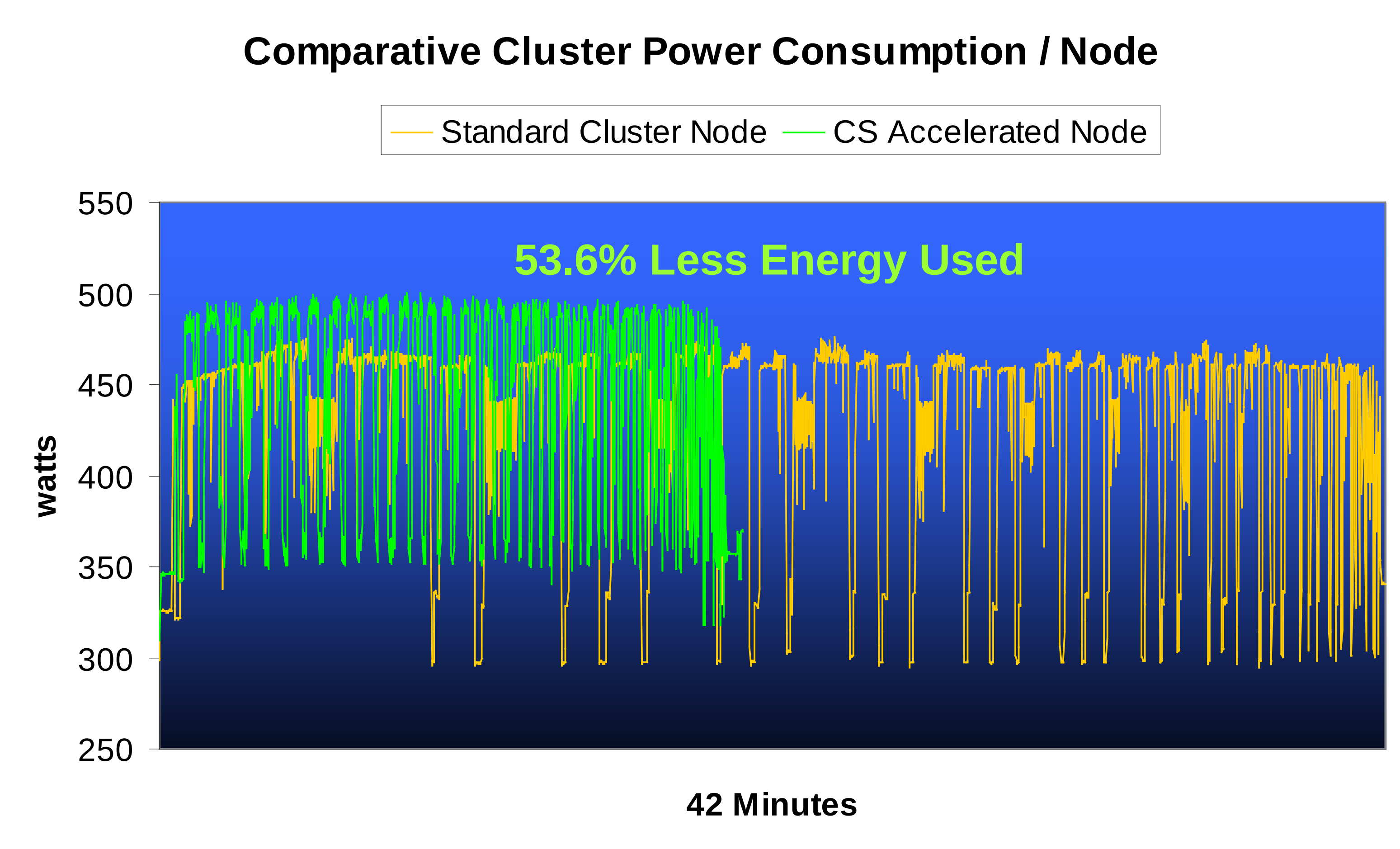
Base System: HP DL380 G5, Intel Xeon 5160 x 2 @ 3GHz, 14GB

Accelerated Cluster Node



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

Energy Usage: Standard vs. Accelerated



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

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 *i j* 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

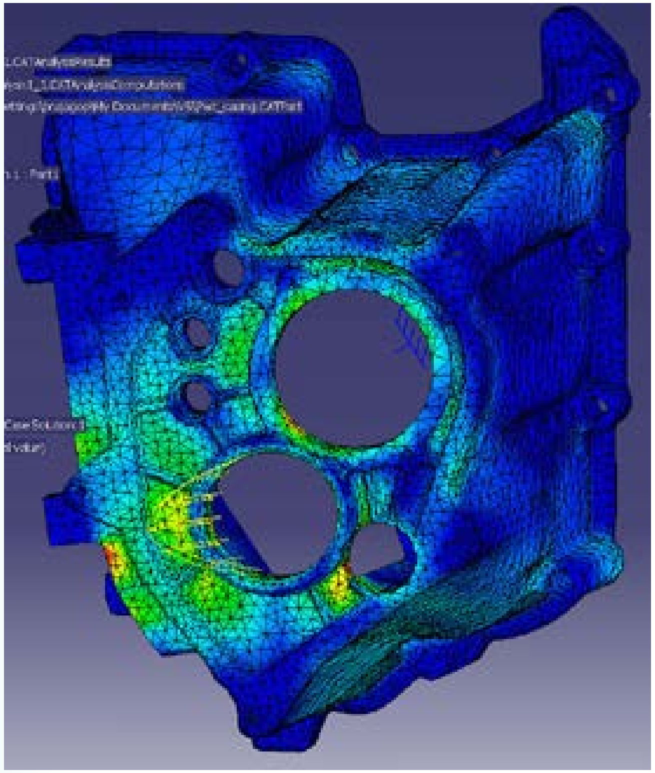
- Newton-Raphson refinement now possible; analytically-computed second derivatives
- 2.6x speedup obtained for this operation in three hours of effort (no source code changes)
- Enables accurate computation of entropy and Gibbs free energy for first time.
- Available now in NAB (Nucleic Acid Builder)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

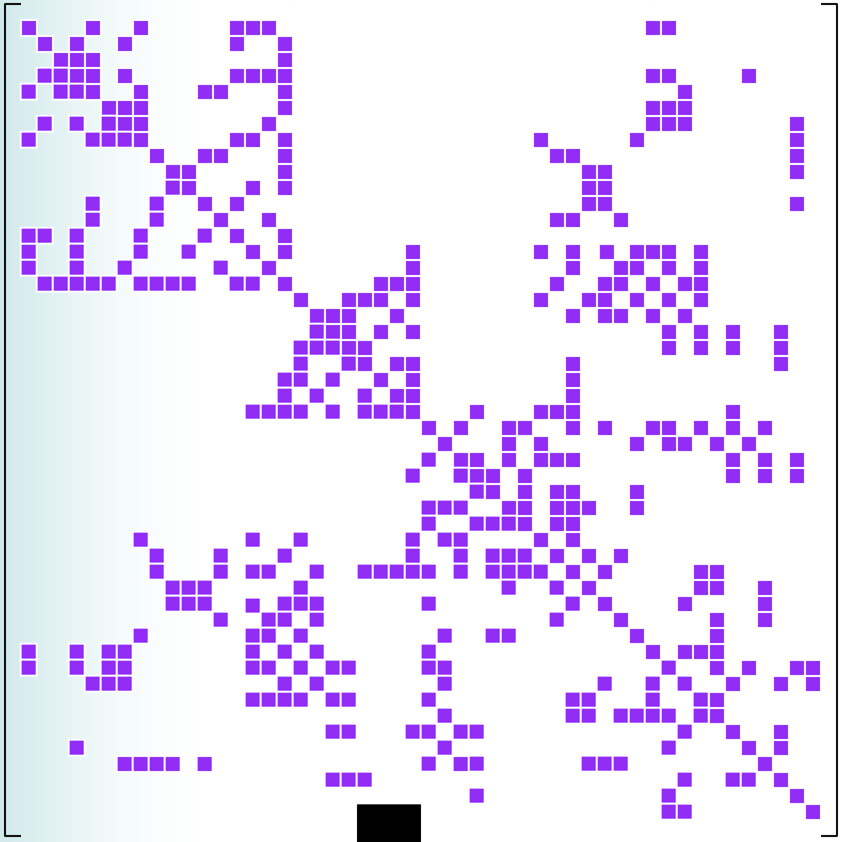
Structural Analysis
ANSYS, LS-DYNA implicit



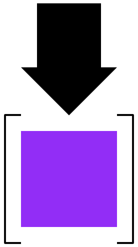
- Each host costs $3,000.
- Software license costs ~$30,000 *per core*, which discourages use of multiple cores.
- MCAE engineer costs over $200,000/year.
- In California, anyway.
- Accelerator card would be cost-effective even with a 7% performance boost. Actual performance boost should be more like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

Accelerating ANSYS, LS-DYNA with Lucas’ solver

10 million degrees of freedom (sparse)

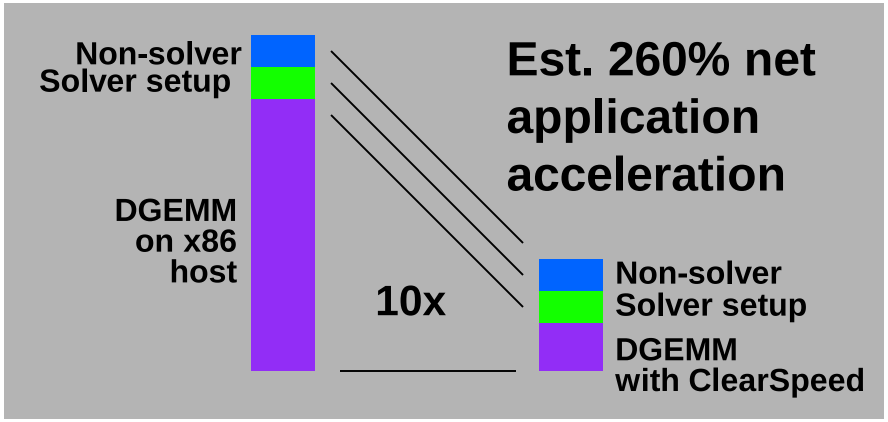


becomes…



50,000 *dense* equivalent

Accelerator can solve at over 50 GFLOPS



Est. 260% net application acceleration

- **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^TA variant**
- **Could enable some CFD acceleration (for codes based on finite elements, low Reynolds numbers`)**

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.