Tuning sparse matrix kernels for multicore

U.C. Berkeley / LBNL

→ Sam Williams, Lenny Oliker, John Shalf, James Demmel, Katherine Yelick

Georgia Tech

→ Richard Vuduc

CSE/CS 8803 PNA: Parallel Numerical Algorithms

[L.25] Tuesday, April 15, 2008

bebop
Berkeley Benchmarking and OPtimization Group
bebop.cs.berkeley.edu

BIPS

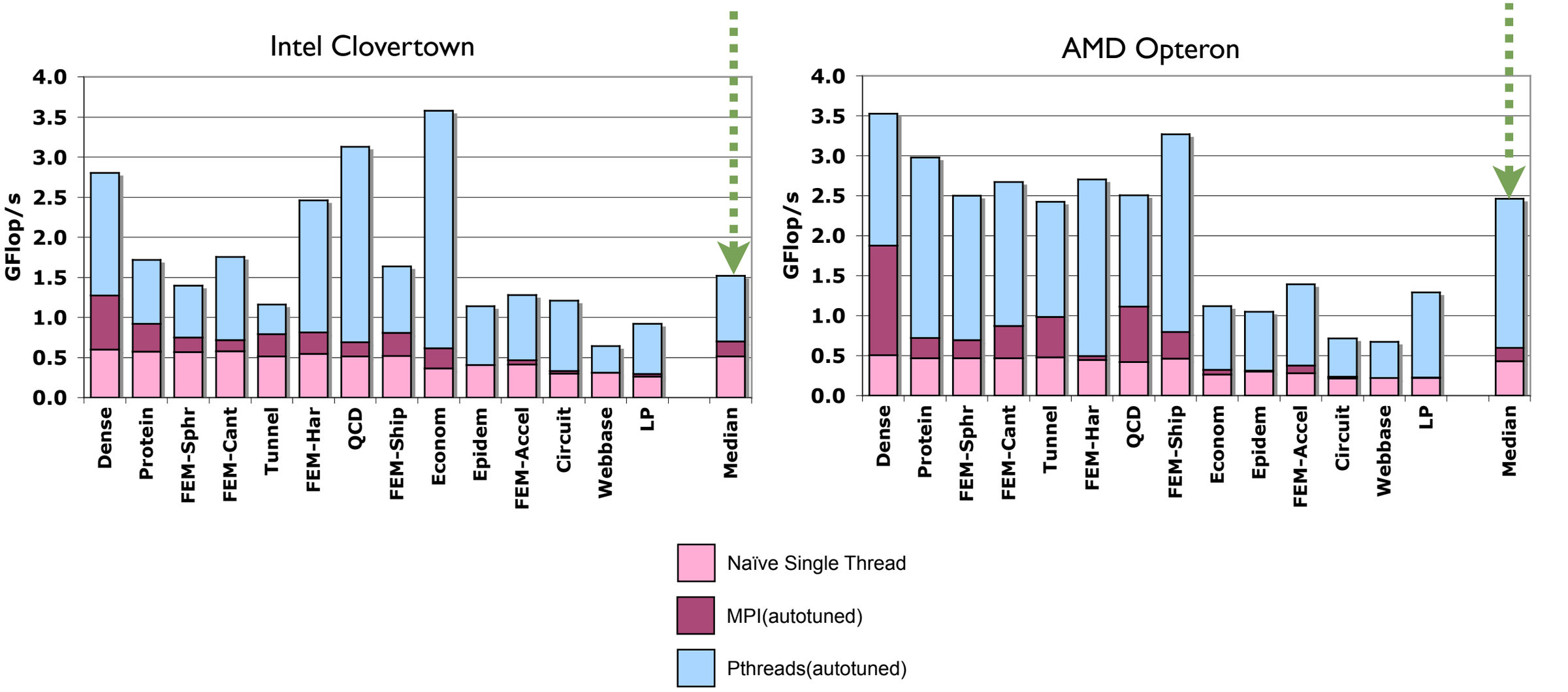
Berkeley Lab

Office of Science
U.S. Department of Energy

Georgia Tech College of Computing
Computational Science and Engineering

~2x on 2x4 cores

~4x on 2x2 cores



Multicore-specific vs. "Off-the-shelf"

Sparse matrix-vector multiply (SpMV)

PETSc 2.3.0+MPICH 1.2.7+OSKI 1.0.1h

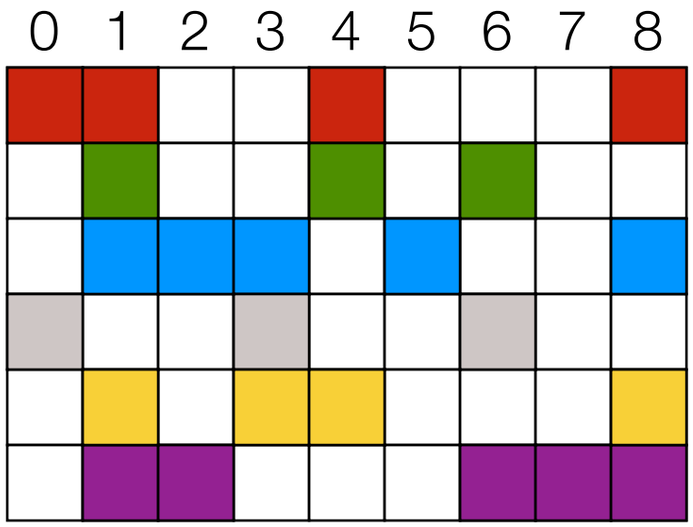
Preview

- Comparisons
  - Intel 4-core, AMD 2-core, Sun Niagara2, STI Cell
  - Multicore-specific vs. "Off-the-shelf" MPI
- Key conclusions
  - Machine with the most complex architecture needs most tuning & achieves lowest performance
  - 50–60% DRAM bandwidth vs. 90% on Cell
  - Place, Prefetch, Compress, Block—and tune!

- **The need for automatic tuning**

  (A 20-year retrospective)

Compressed sparse row (CSR) format

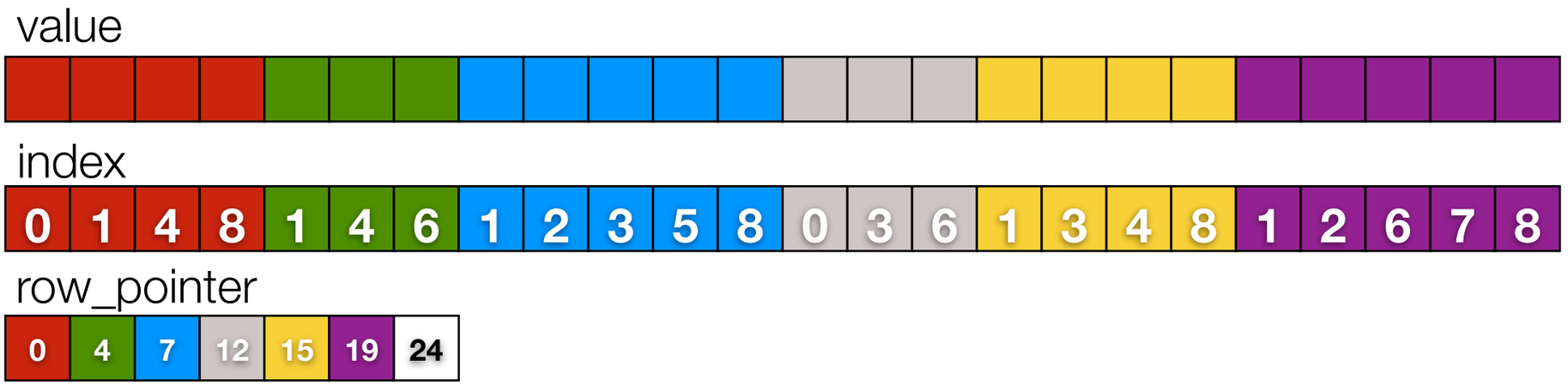


value

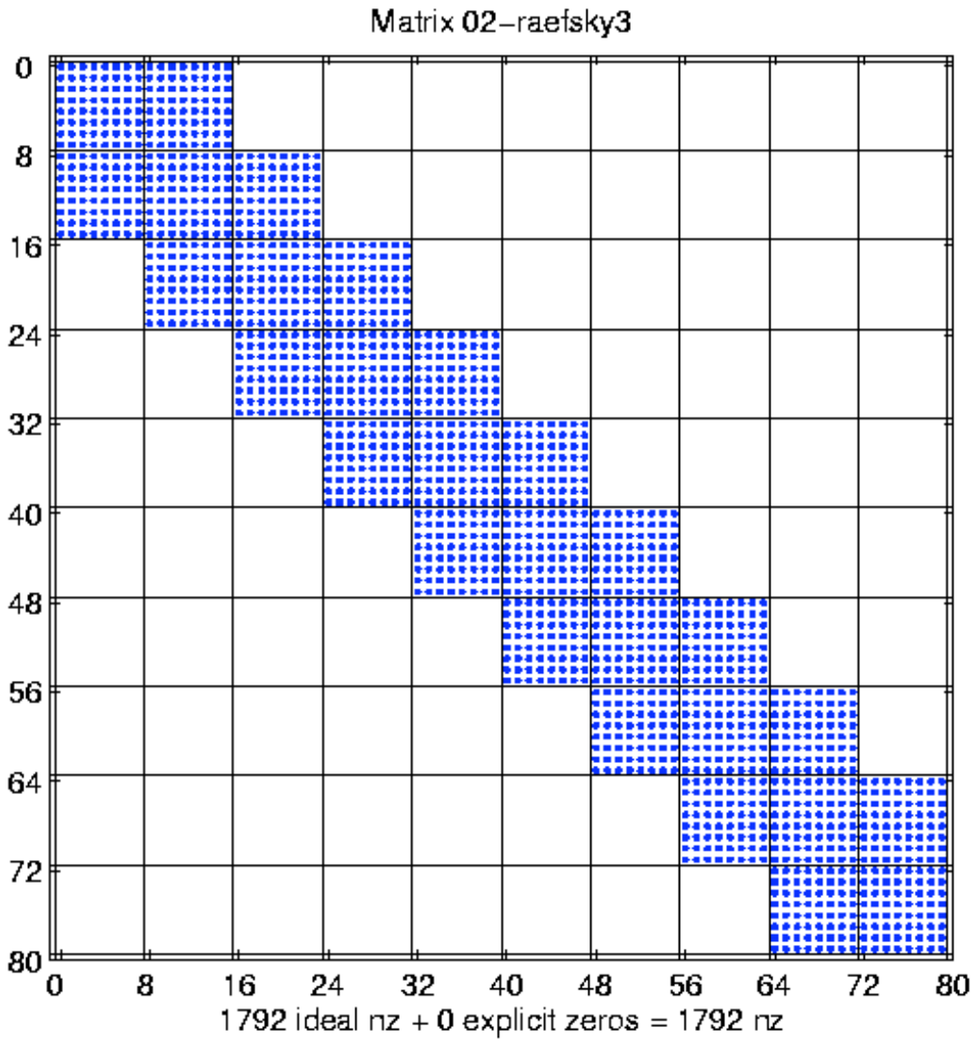
index

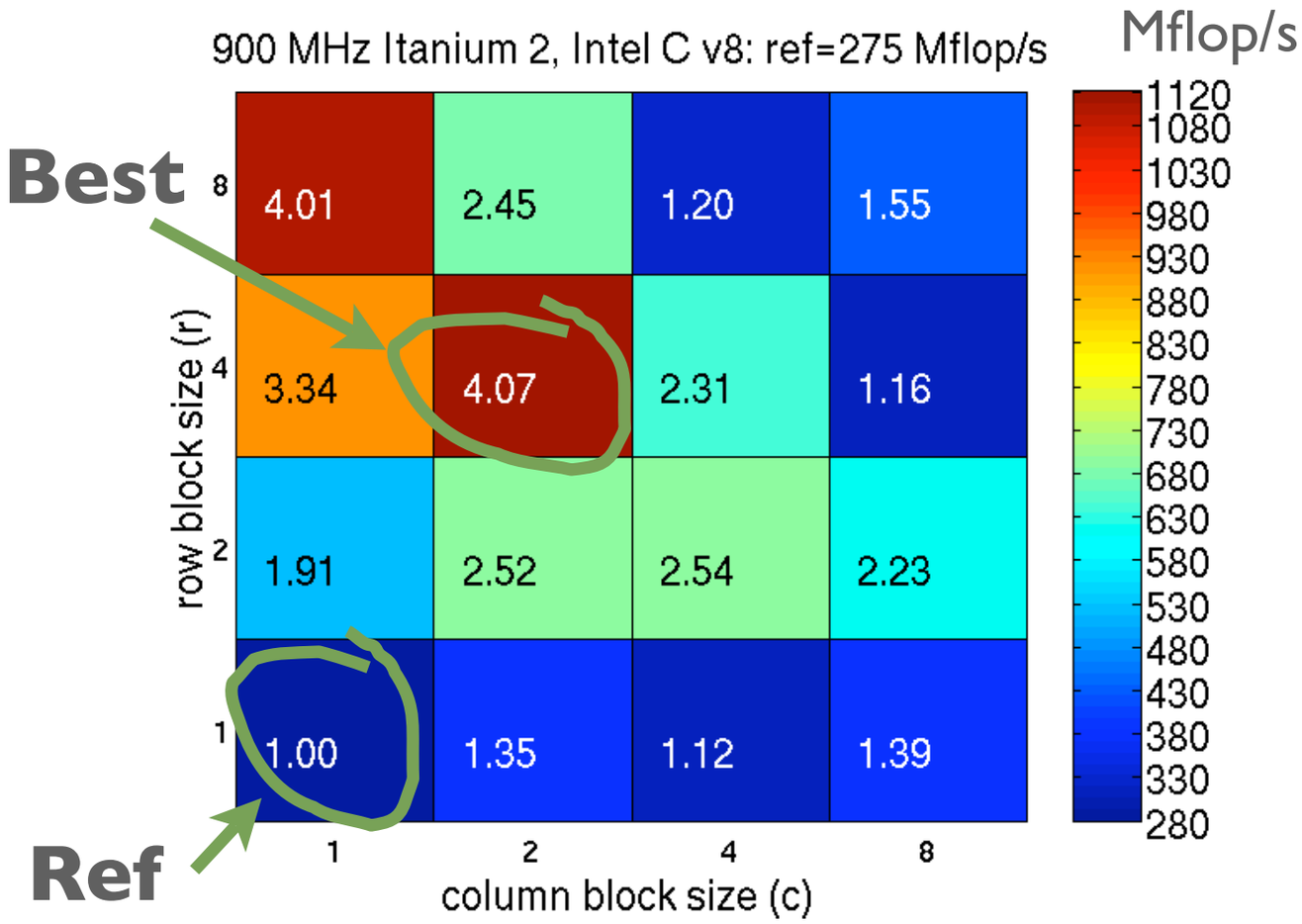
row_pointer

SpMV performance: Rules of thumb

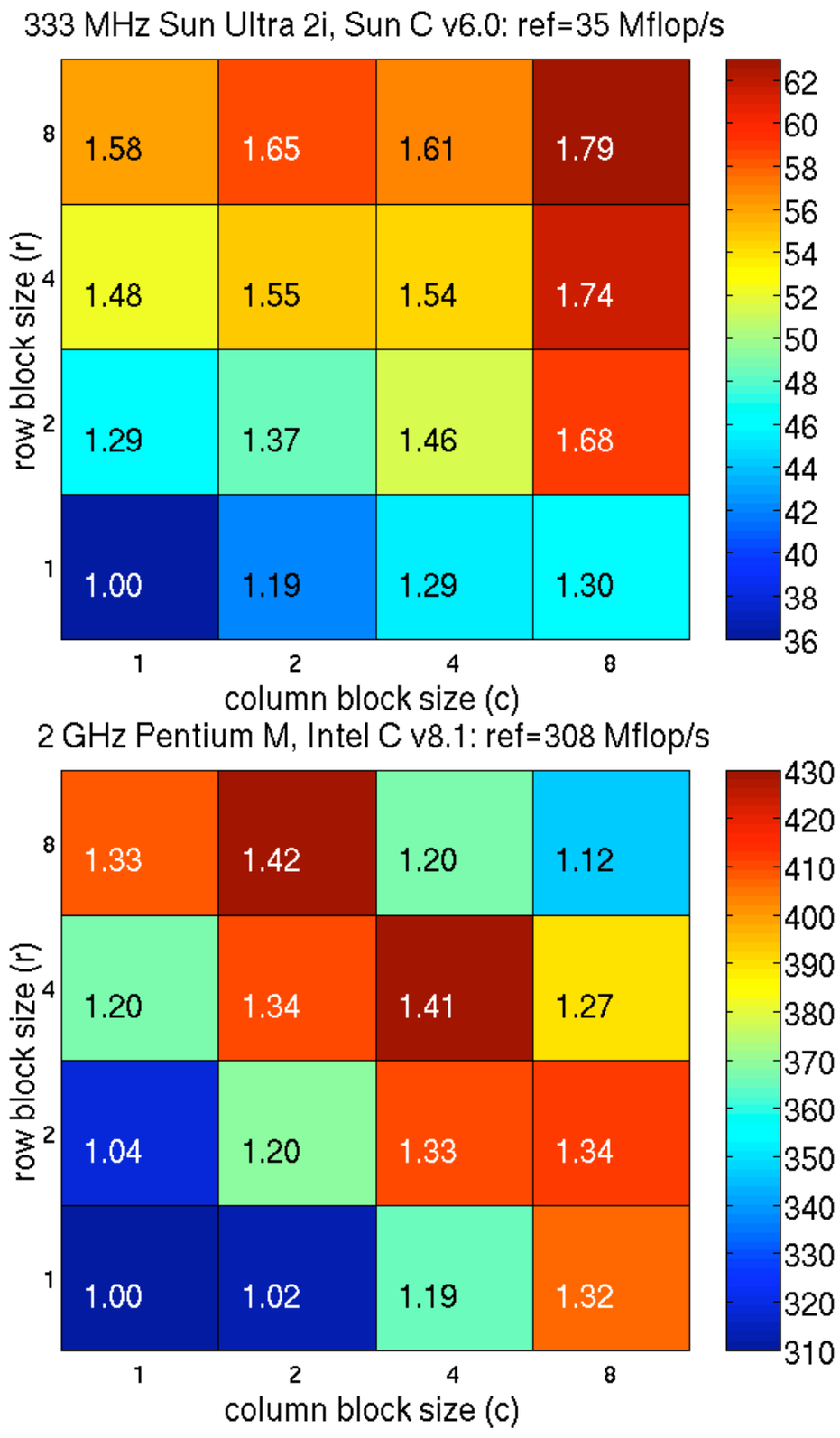


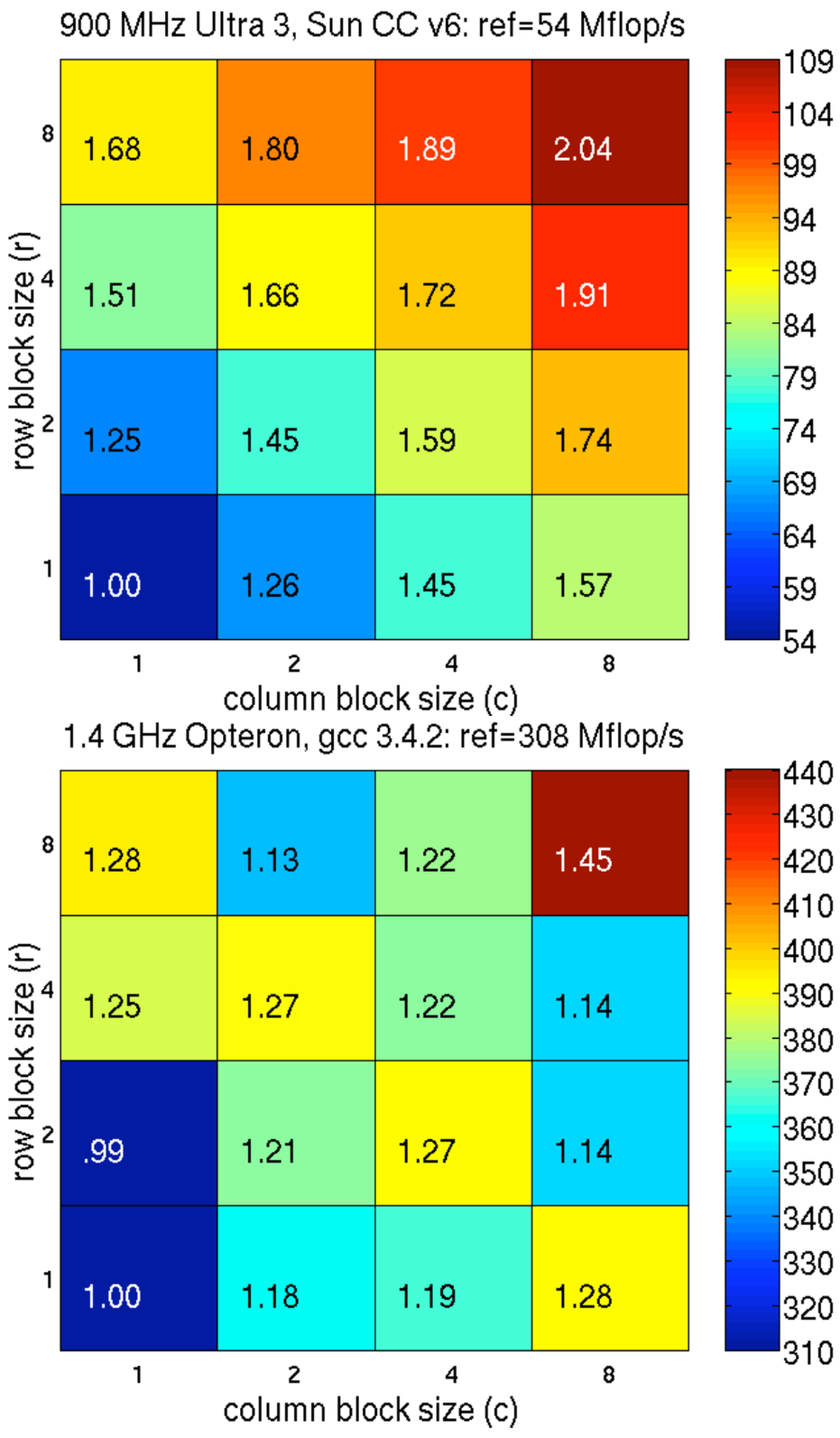
- Low computational intensity, vs. dense case
- Serial performance ~ 10% peak or less
- Bandwidth limited → compress
- Expect 1.5x speedup max, 32b ints + 64b vals

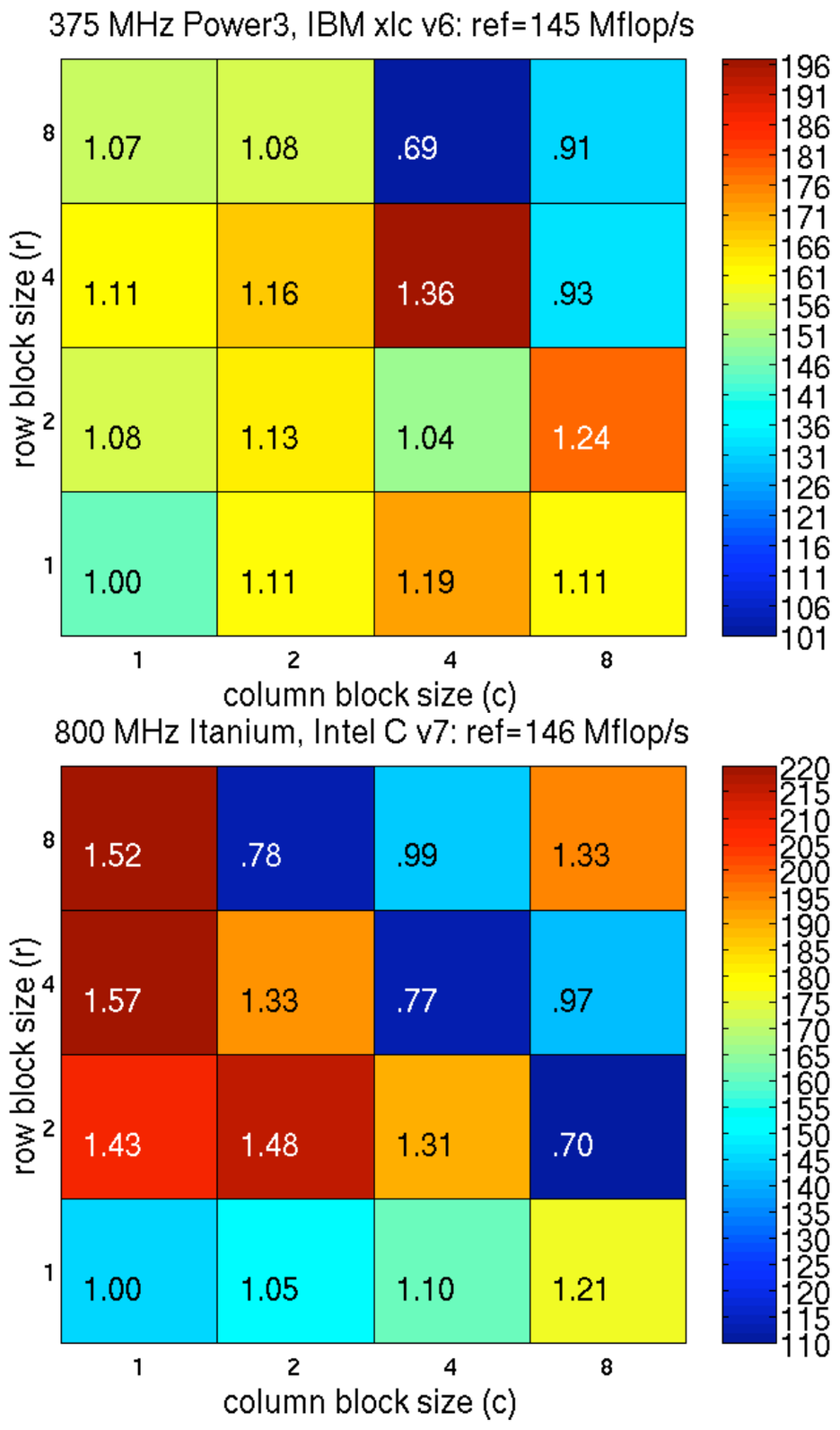


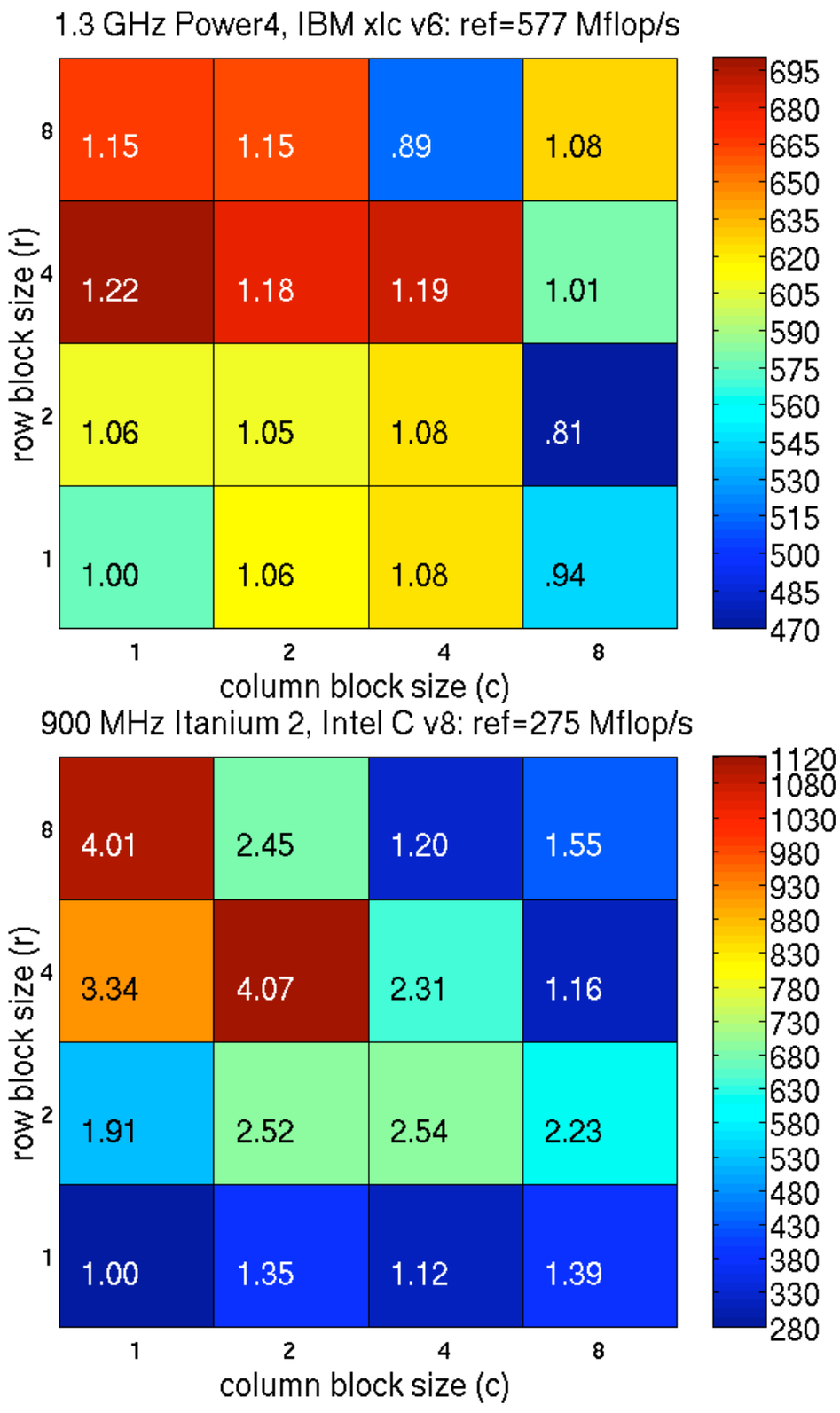


Optimal block size not obvious


333 MHz Sun Ultra 2i, Sun C v6.0: ref=35 Mflop/s
1.58
1.65
1.61
1.79
1.48
1.55
1.54
1.74
1.29
1.37
1.46
1.68
1.00
1.19
1.29
1.30
62
60
58
56
54
52
50
48
46
44
42
40
38
36
row block size (r)
column block size (c)
2 GHz Pentium M, Intel C v8.1: ref=308 Mflop/s
1.33
1.42
1.20
1.12
1.20
1.34
1.41
1.27
1.04
1.20
1.33
1.34
1.00
1.02
1.19
1.32
430
420
410
400
390
380
370
360
350
340
330
320
310
row block size (r)
column block size (c)


900 MHz Ultra 3, Sun CC v6: ref=54 Mflop/s
1.68
1.80
1.89
2.04
1.51
1.66
1.72
1.91
1.25
1.45
1.59
1.74
1.00
1.26
1.45
1.57
109
104
99
94
89
84
79
74
69
64
59
54
row block size (r)
column block size (c)
1.4 GHz Opteron, gcc 3.4.2: ref=308 Mflop/s
1.28
1.13
1.22
1.45
1.25
1.27
1.22
1.14
.99
1.21
1.27
1.14
1.00
1.18
1.19
1.28
440
430
420
410
400
390
380
370
360
350
340
330
320
310
row block size (r)
column block size (c)

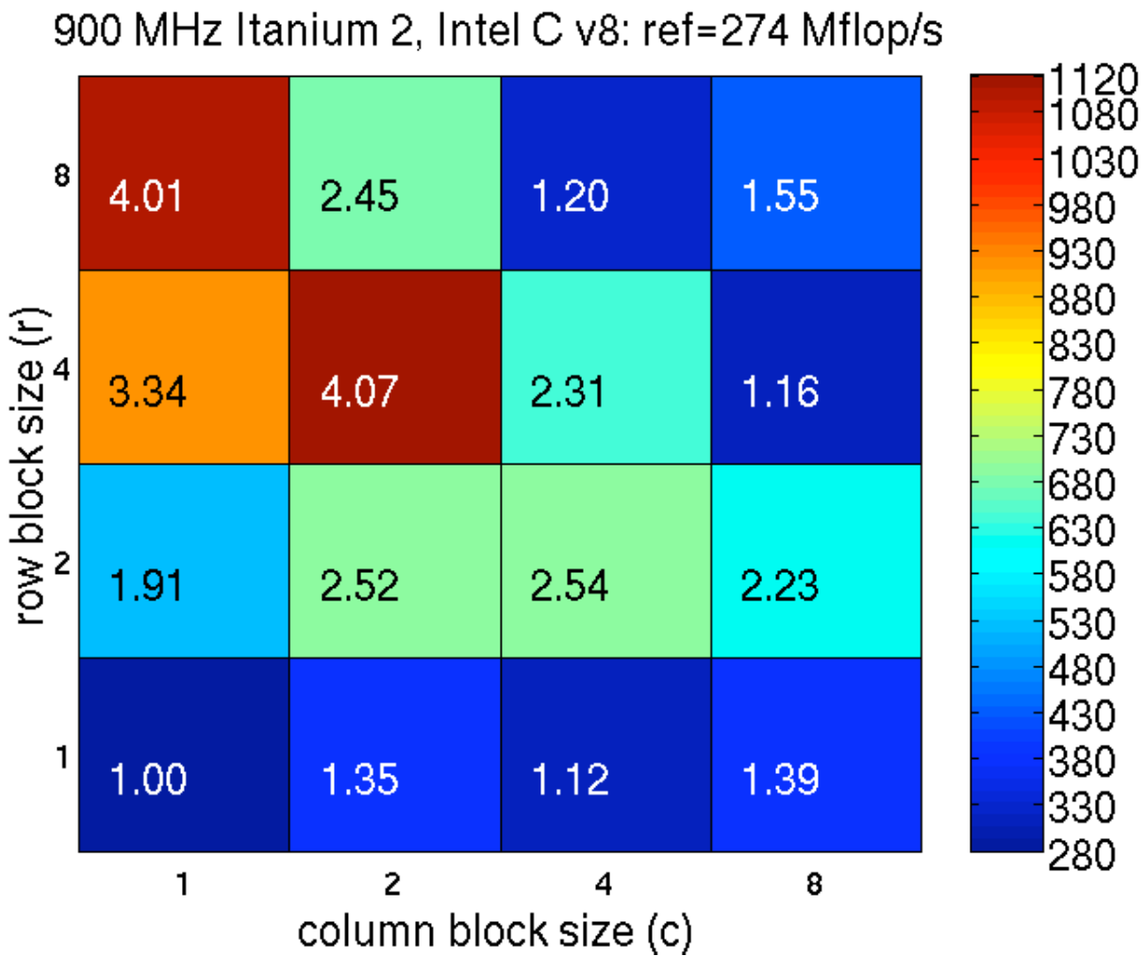

375 MHz Power3, IBM xlc v6: ref=145 Mflop/s
row block size (r)
column block size (c)
1.07
1.08
.69
.91
1.11
1.16
1.36
.93
1.08
1.13
1.04
1.24
1.00
1.11
1.19
1.11
196
191
186
181
176
171
166
161
156
151
146
141
136
131
126
121
116
111
106
101

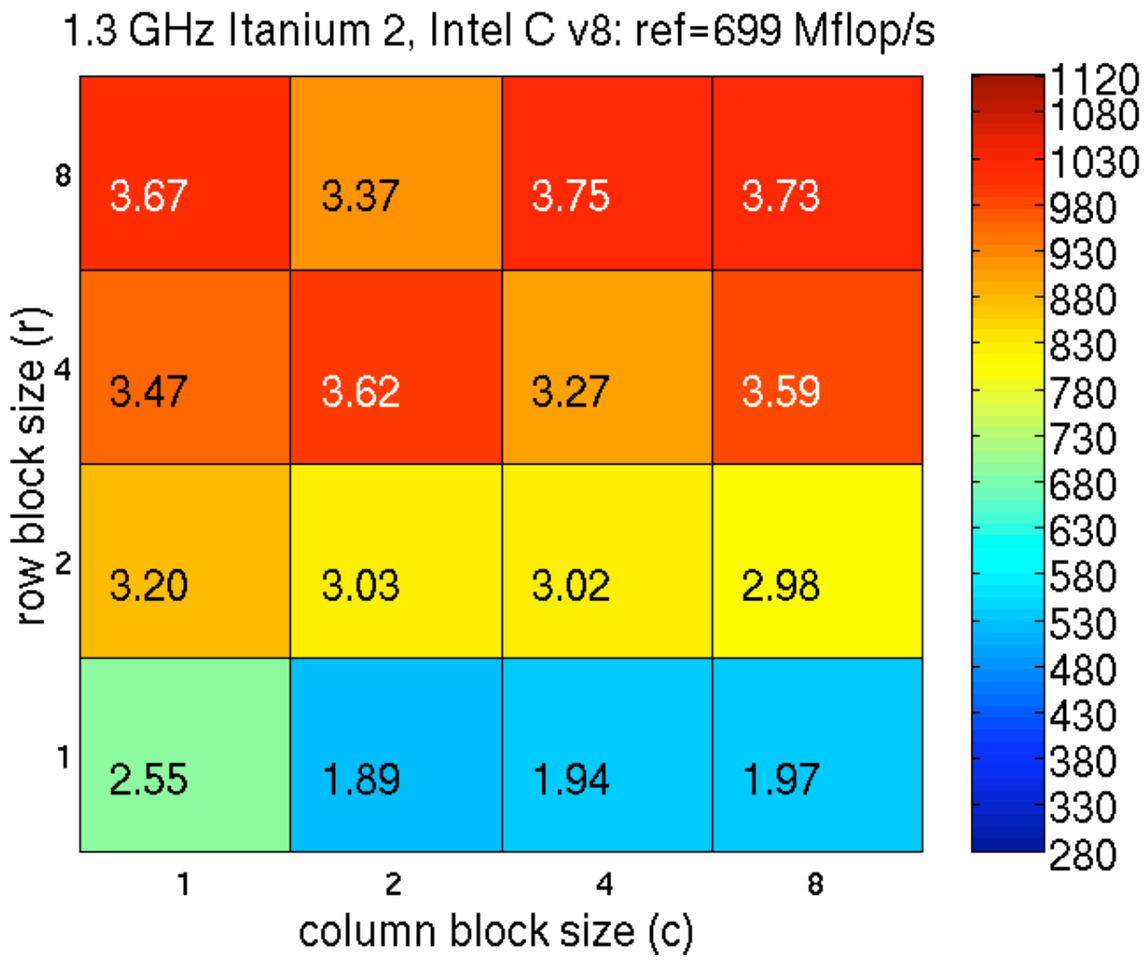

1.3 GHz Power4, IBM xlc v6: ref=577 Mflop/s
row block size (r)
column block size (c)
1.15
1.15
.89
1.08
1.22
1.18
1.19
1.01
1.06
1.05
1.08
.81
1.00
1.06
1.08
.94
695
680
665
650
635
620
605
590
575
560
545
530
515
500
485
470

800 MHz Itanium, Intel C v7: ref=146 Mflop/s
row block size (r)
column block size (c)
1.52
.78
.99
1.33
1.57
1.33
.77
.97
1.43
1.48
1.31
.70
1.00
1.05
1.10
1.21
220
215
210
205
200
195
190
185
180
175
170
165
160
155
150
145
140
135
130
125
120
115
110

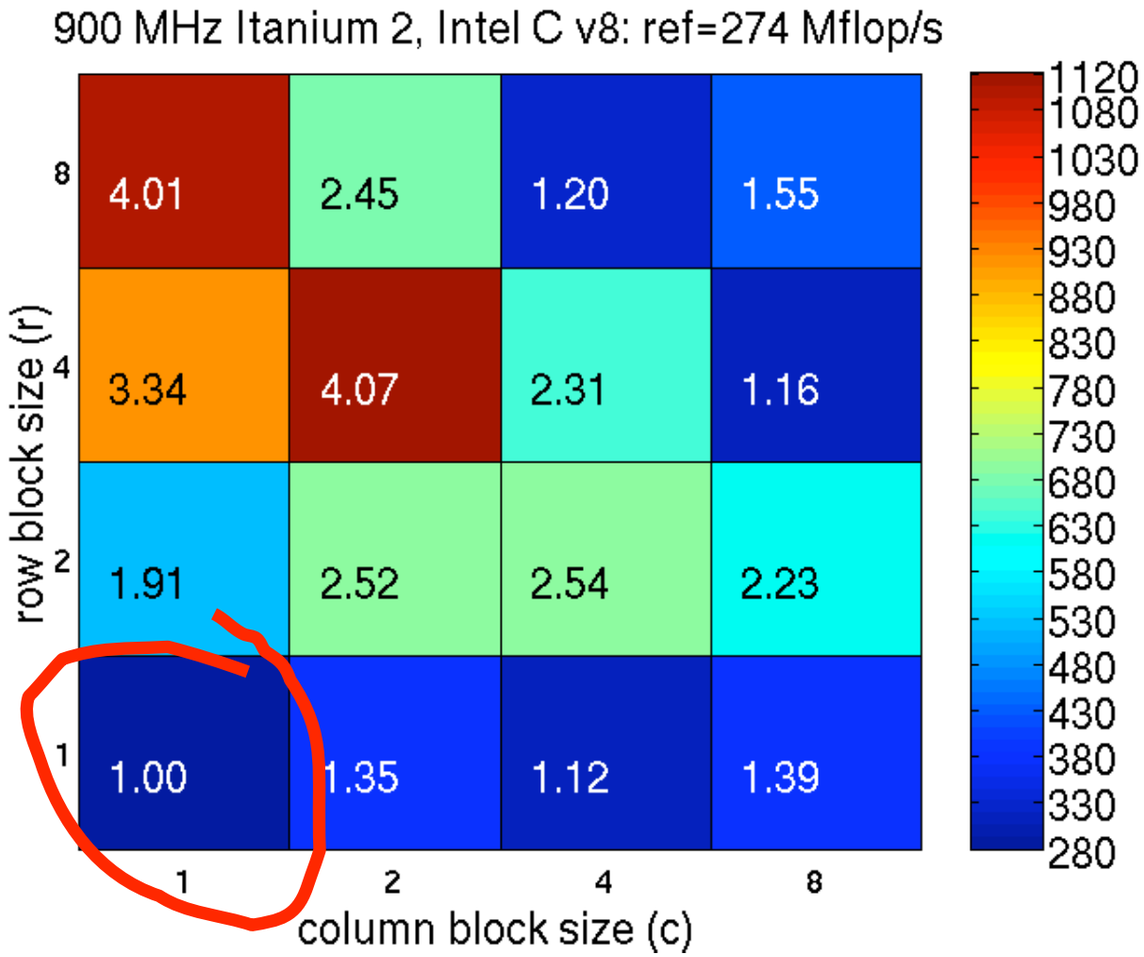
900 MHz Itanium 2, Intel C v8: ref=275 Mflop/s
row block size (r)
column block size (c)
4.01
2.45
1.20
1.55
3.34
4.07
2.31
1.16
1.91
2.52
2.54
2.23
1.00
1.35
1.12
1.39
1120
1080
1030
980
930
880
830
780
730
680
630
580
530
480
430
380
330
280

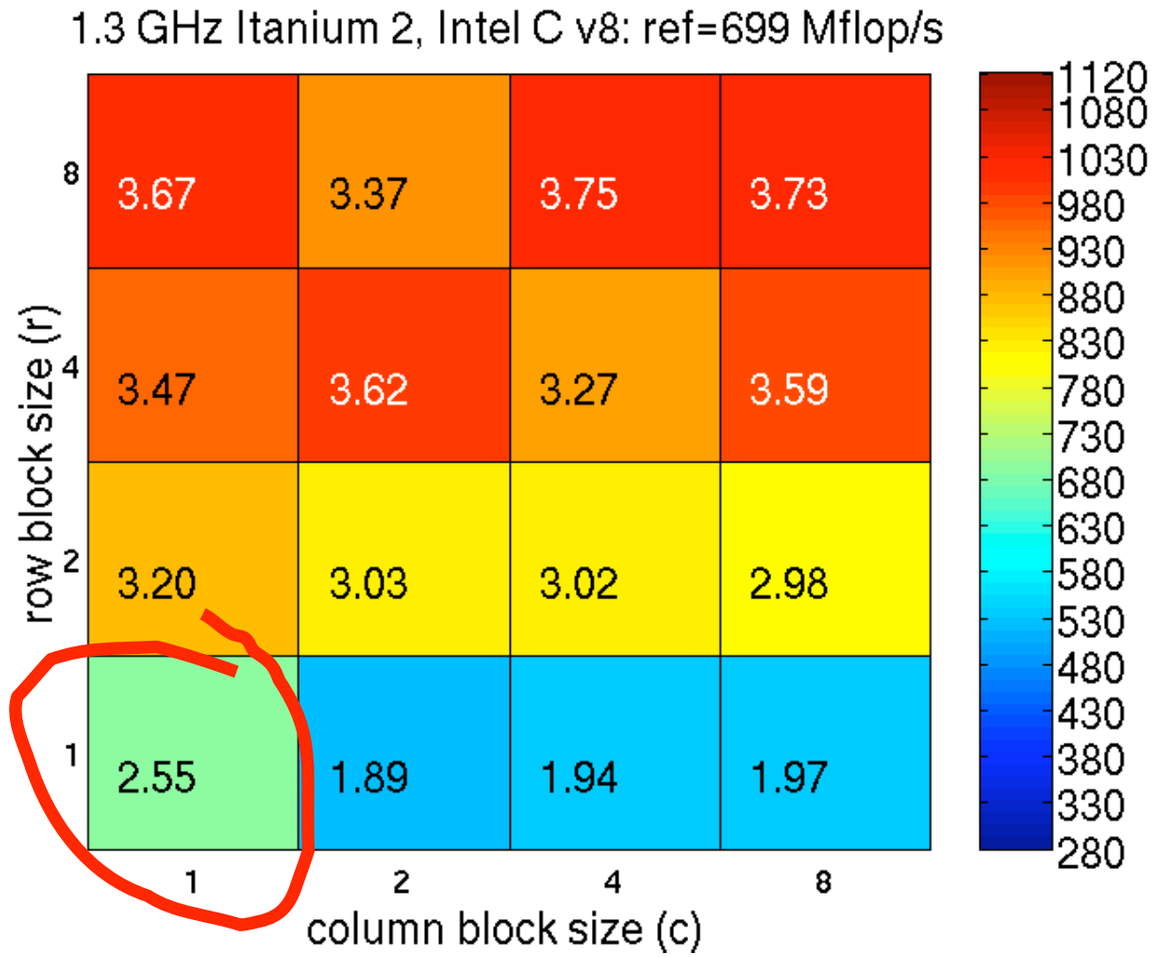
Better, worse, or about the same?



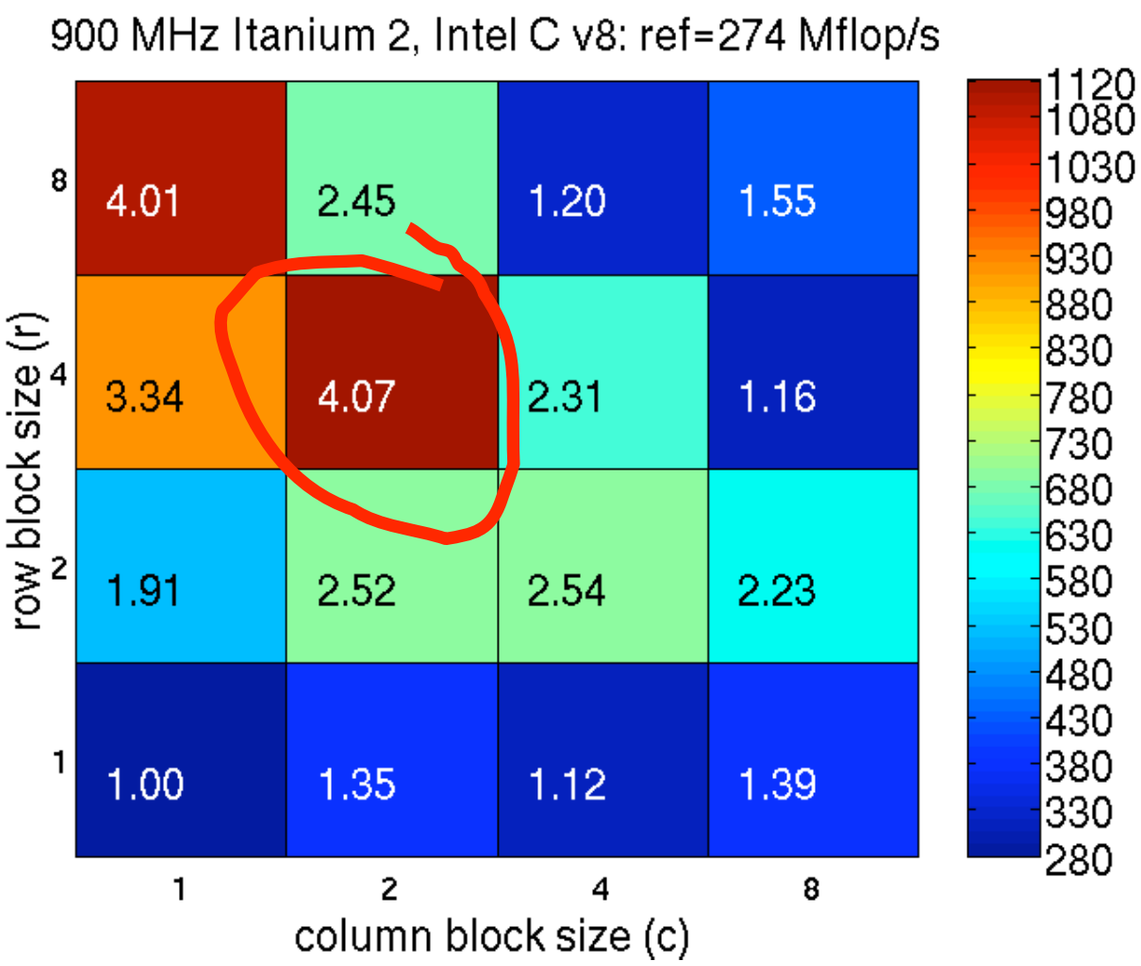


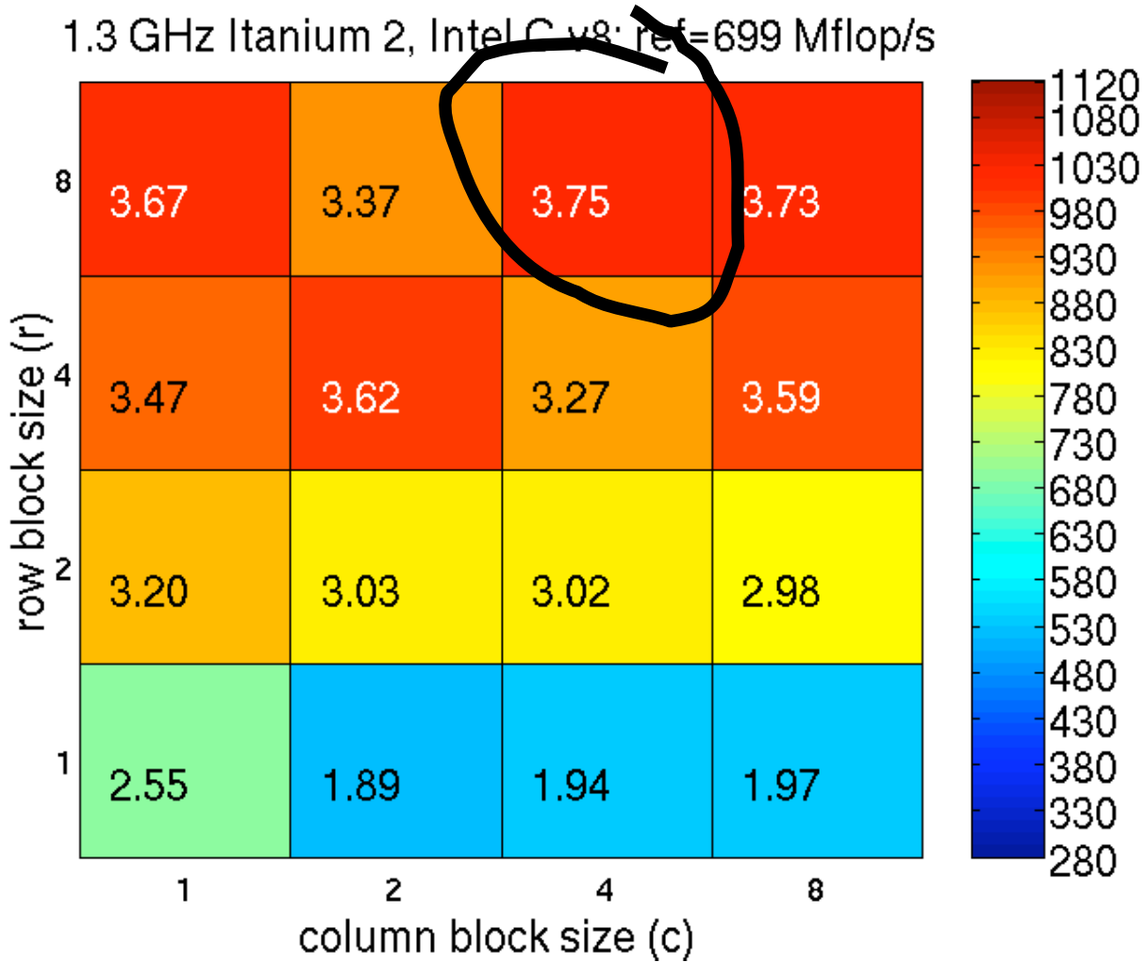
Example 1: Processor upgrade.



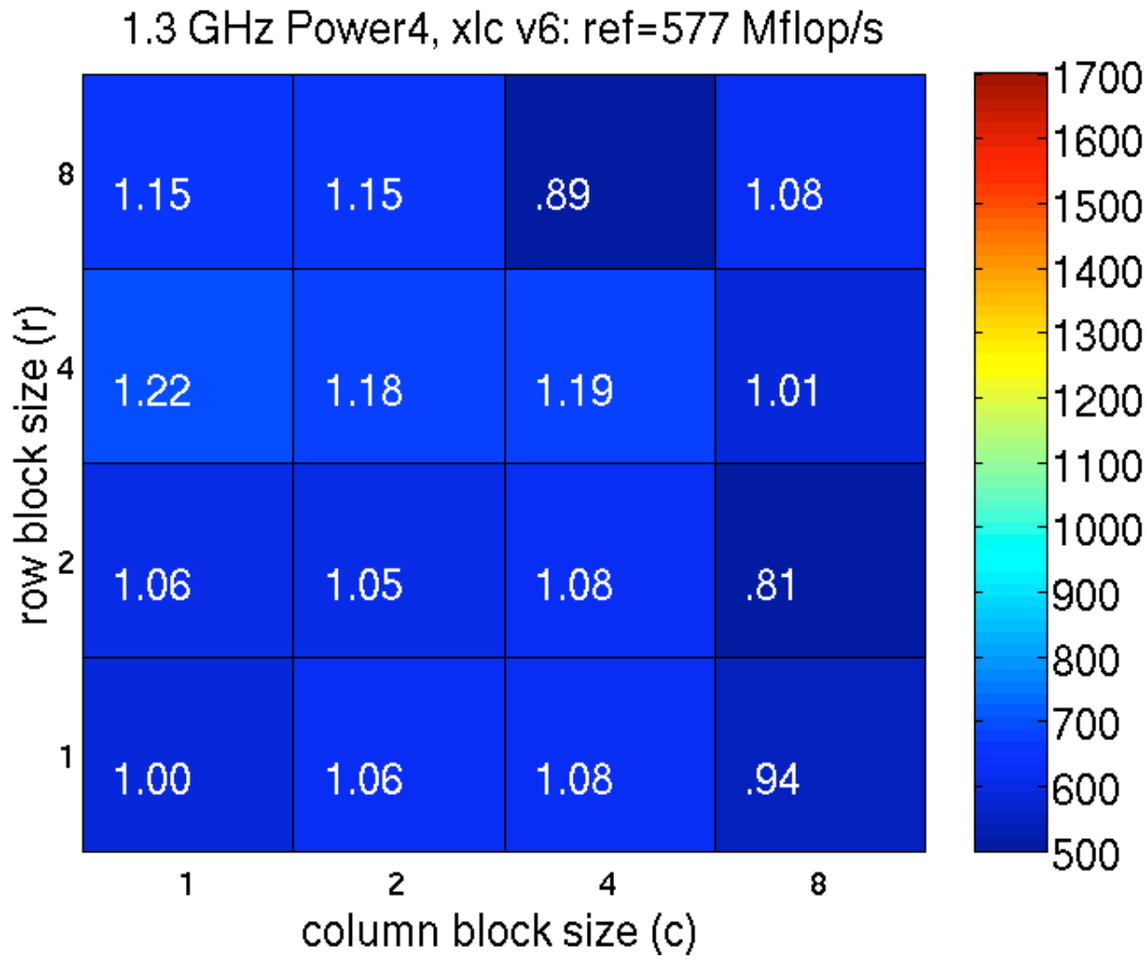


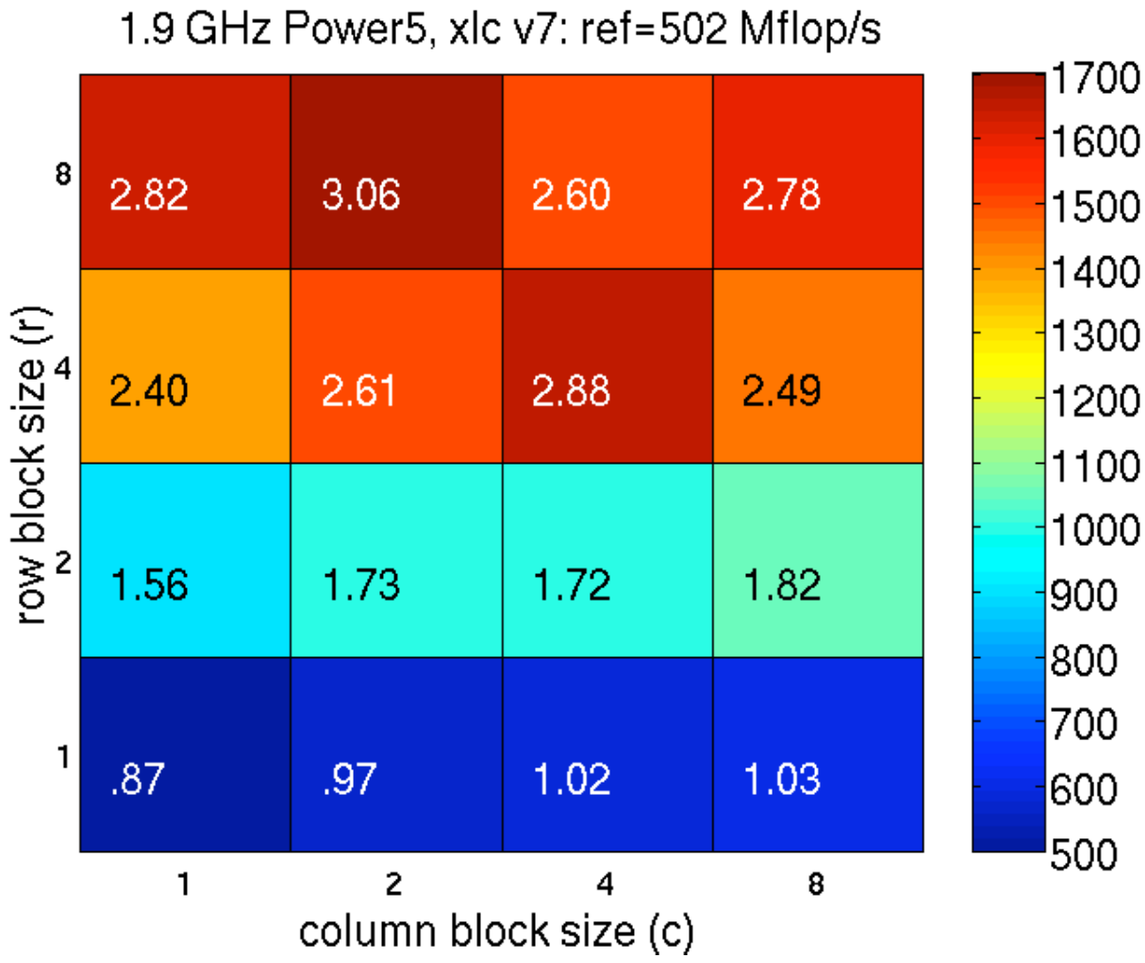
Reference improves…



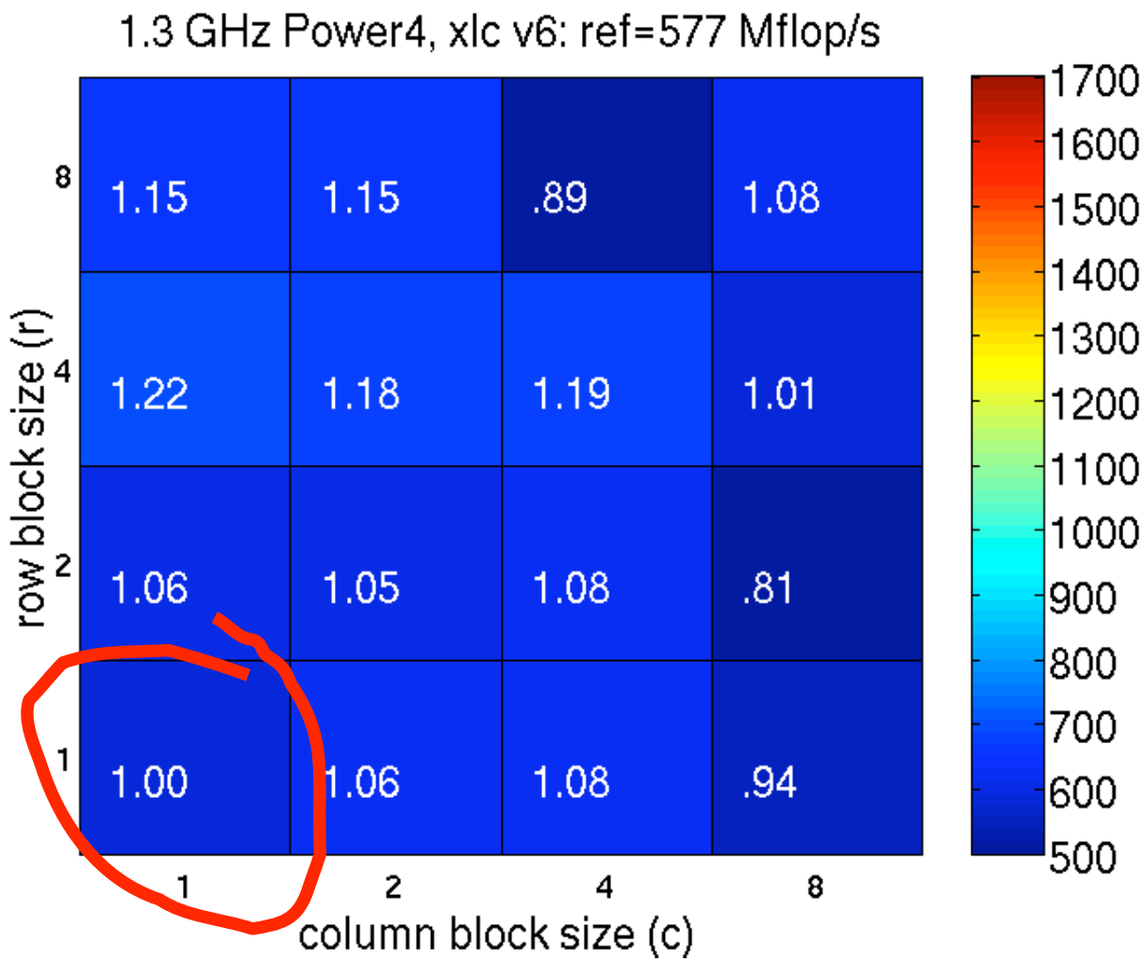


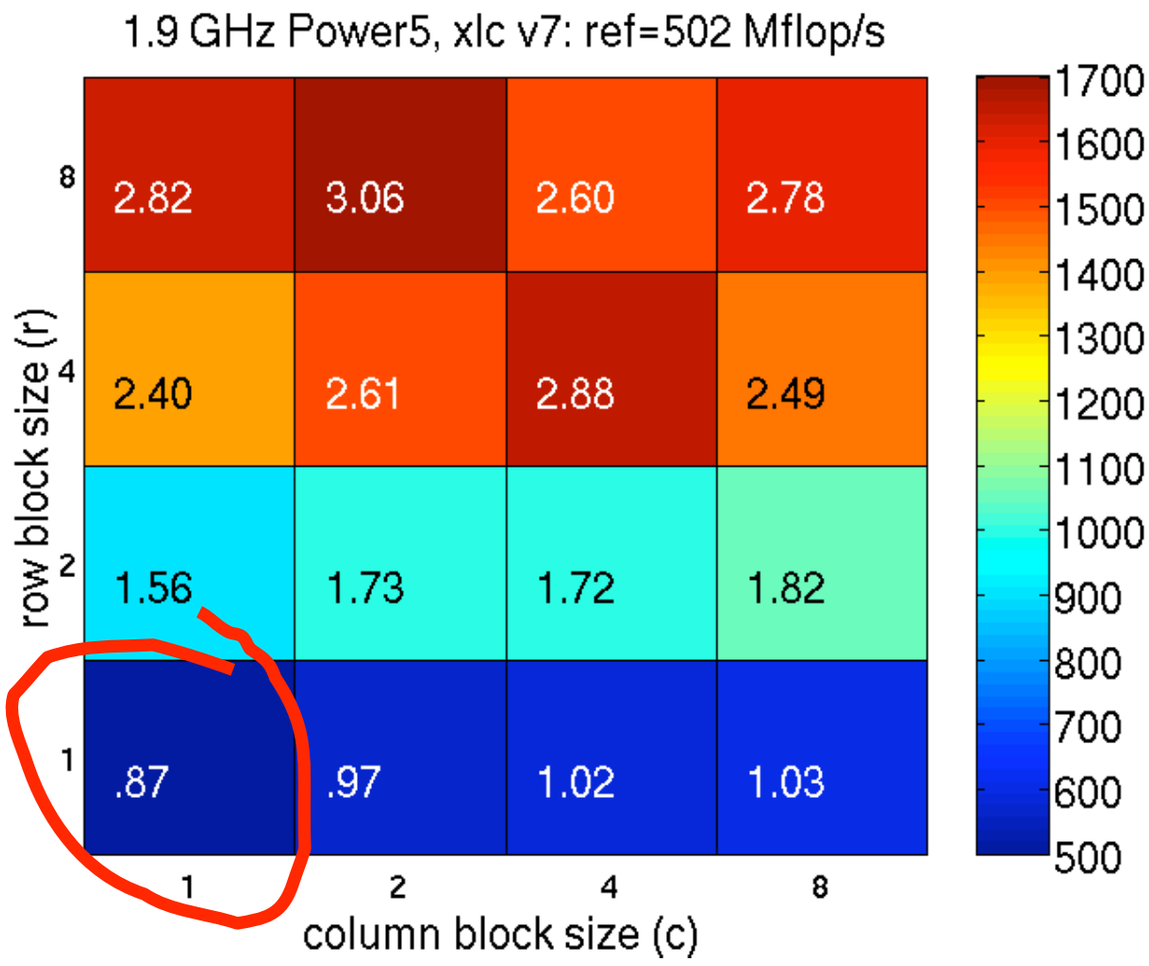
… and best possible worsens?



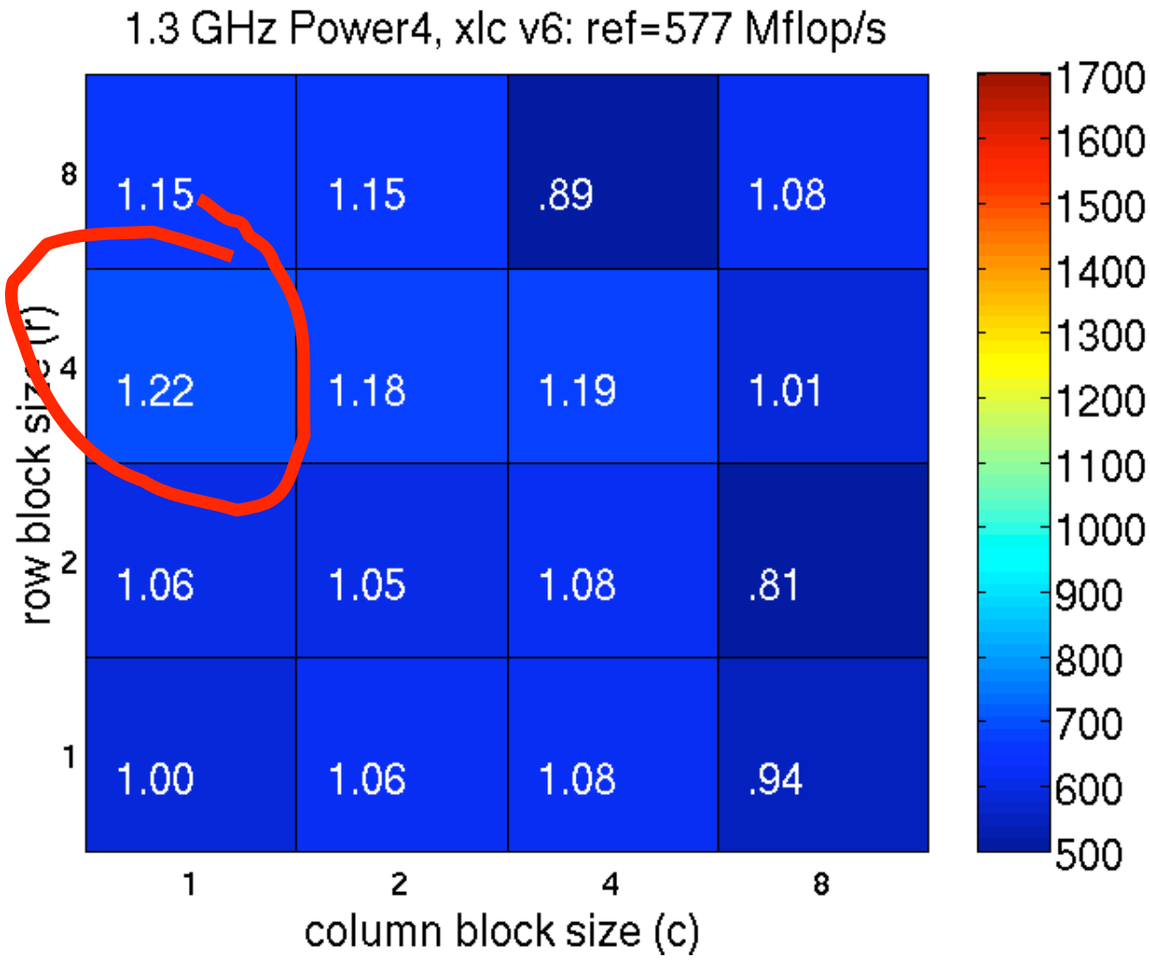


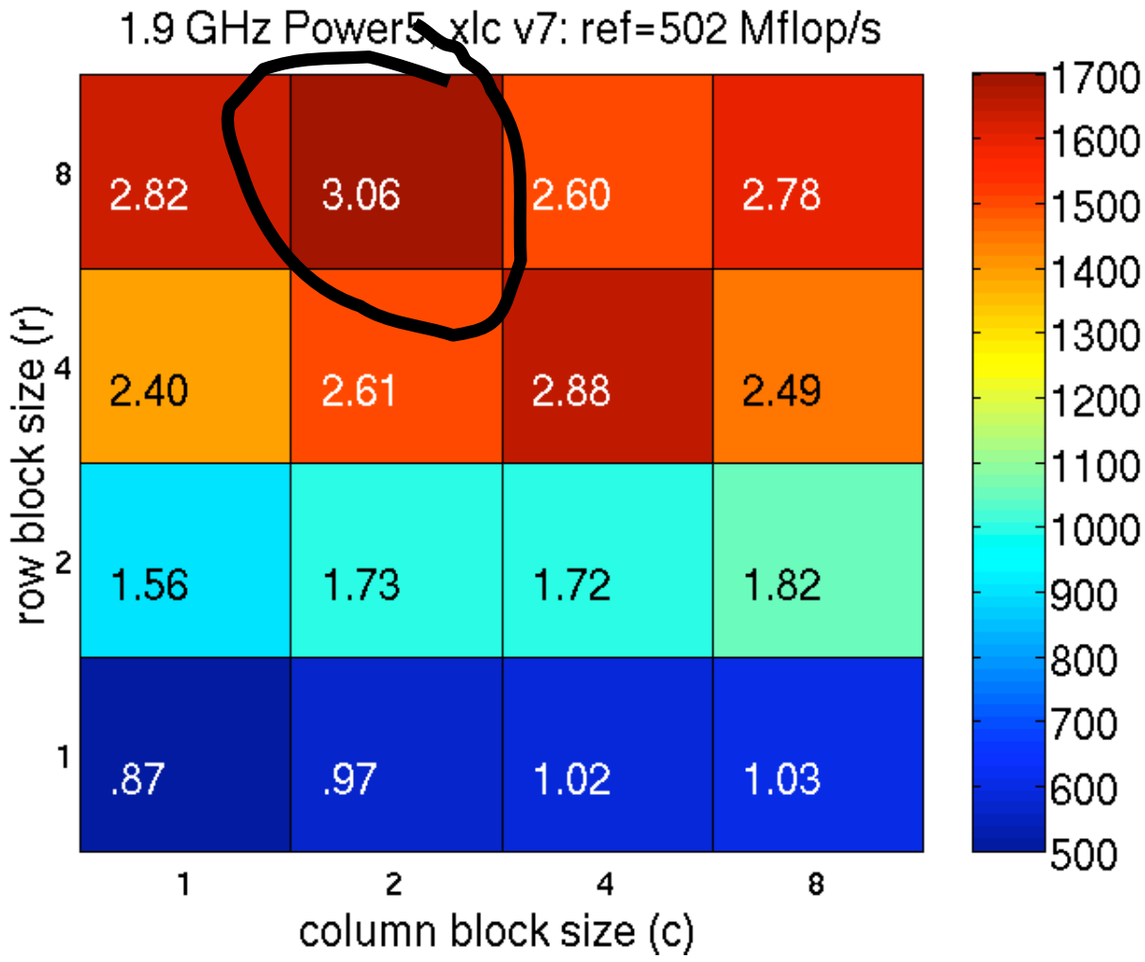
Example 2: Power4 ⇒ Power5.



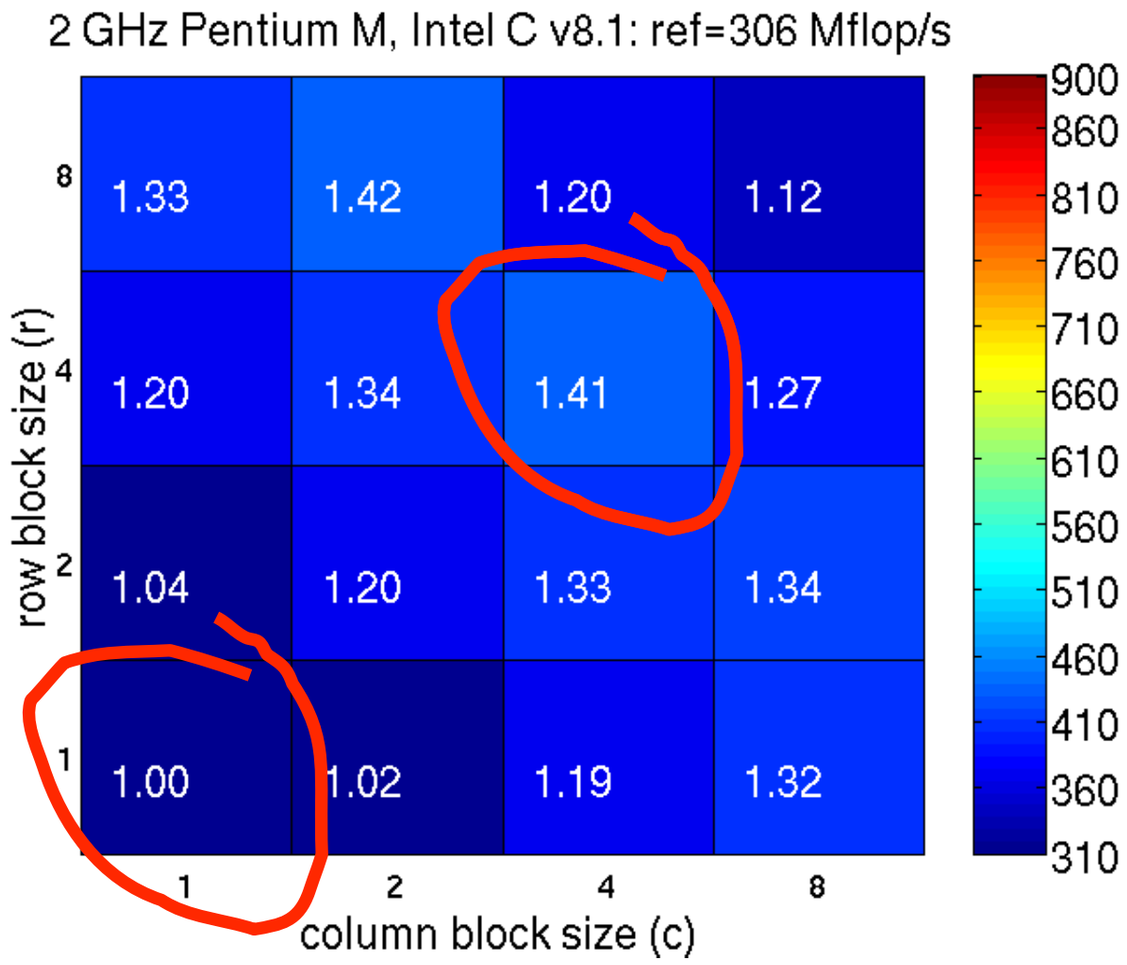


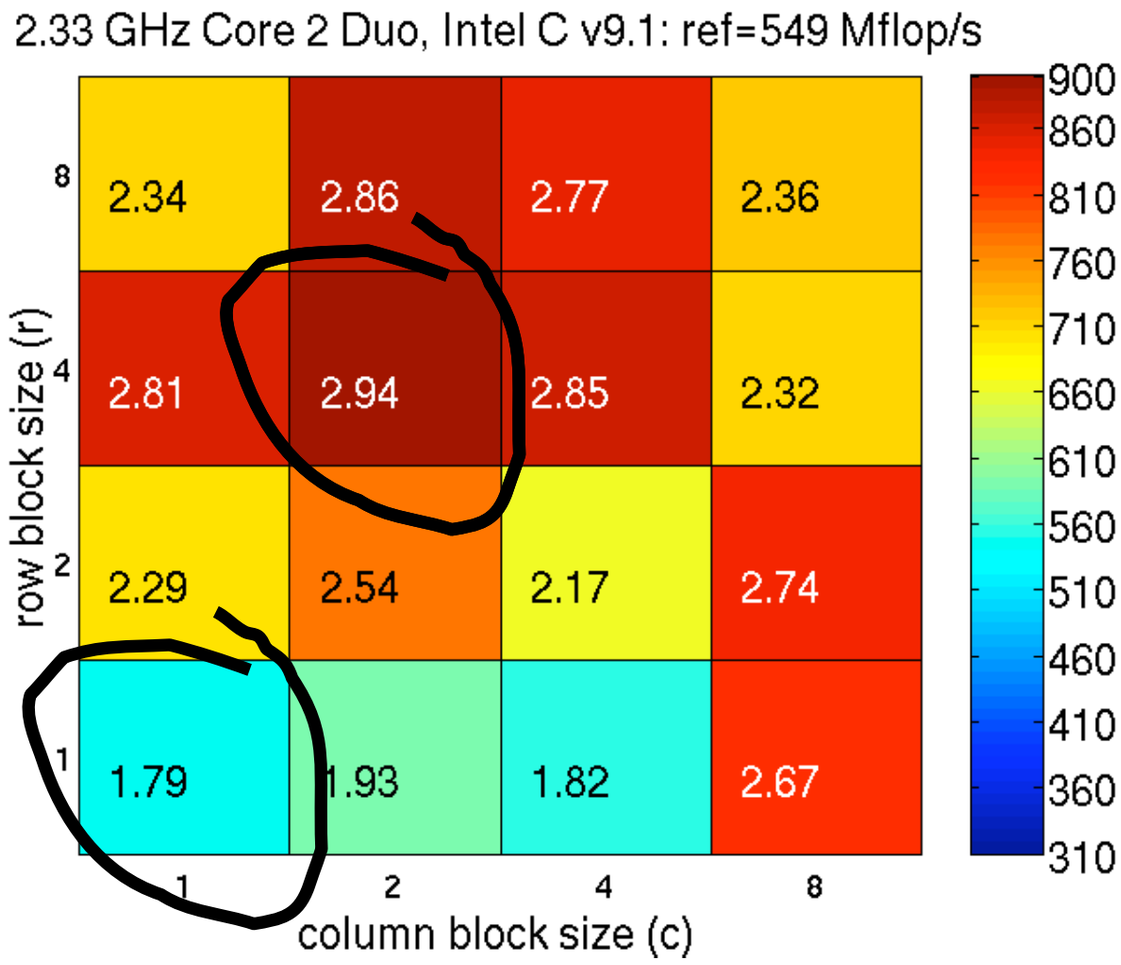
Reference gets worse!



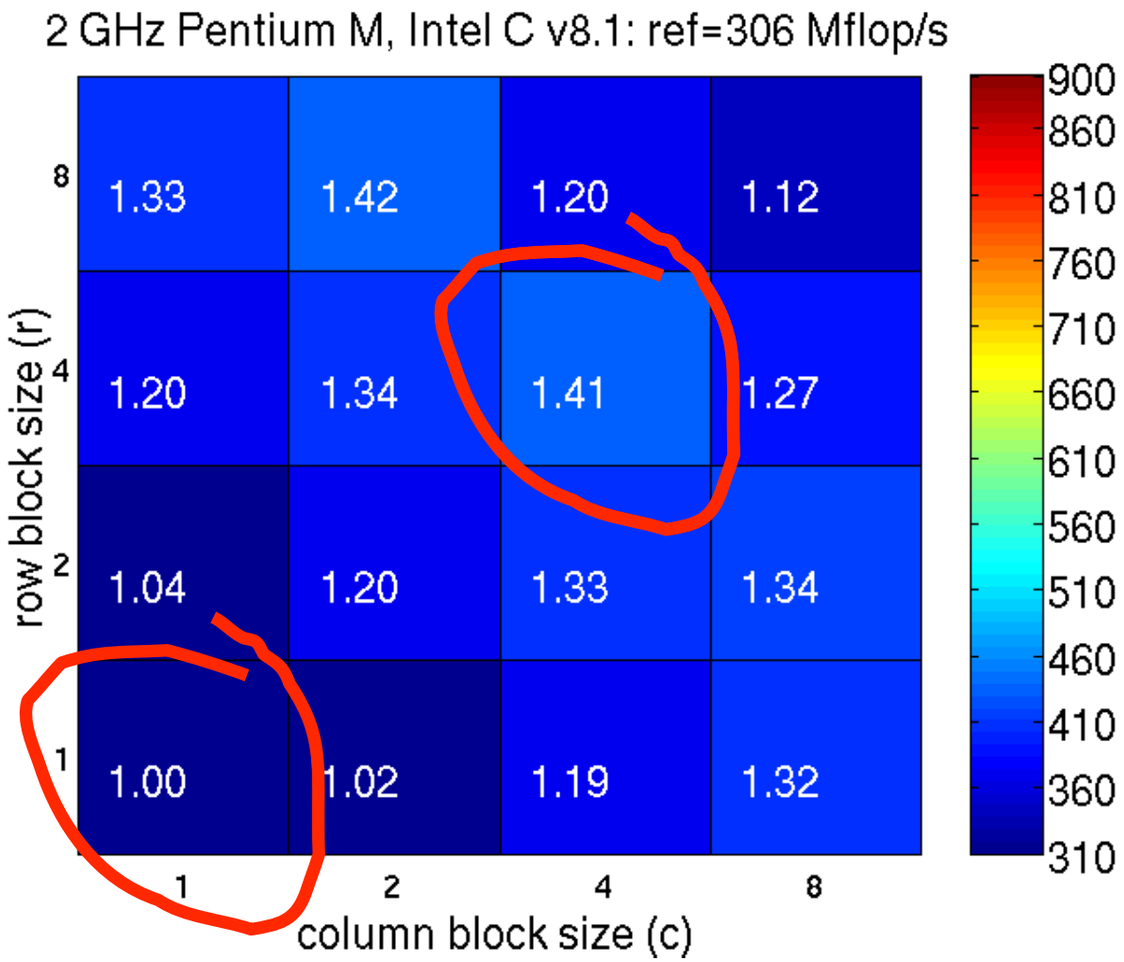


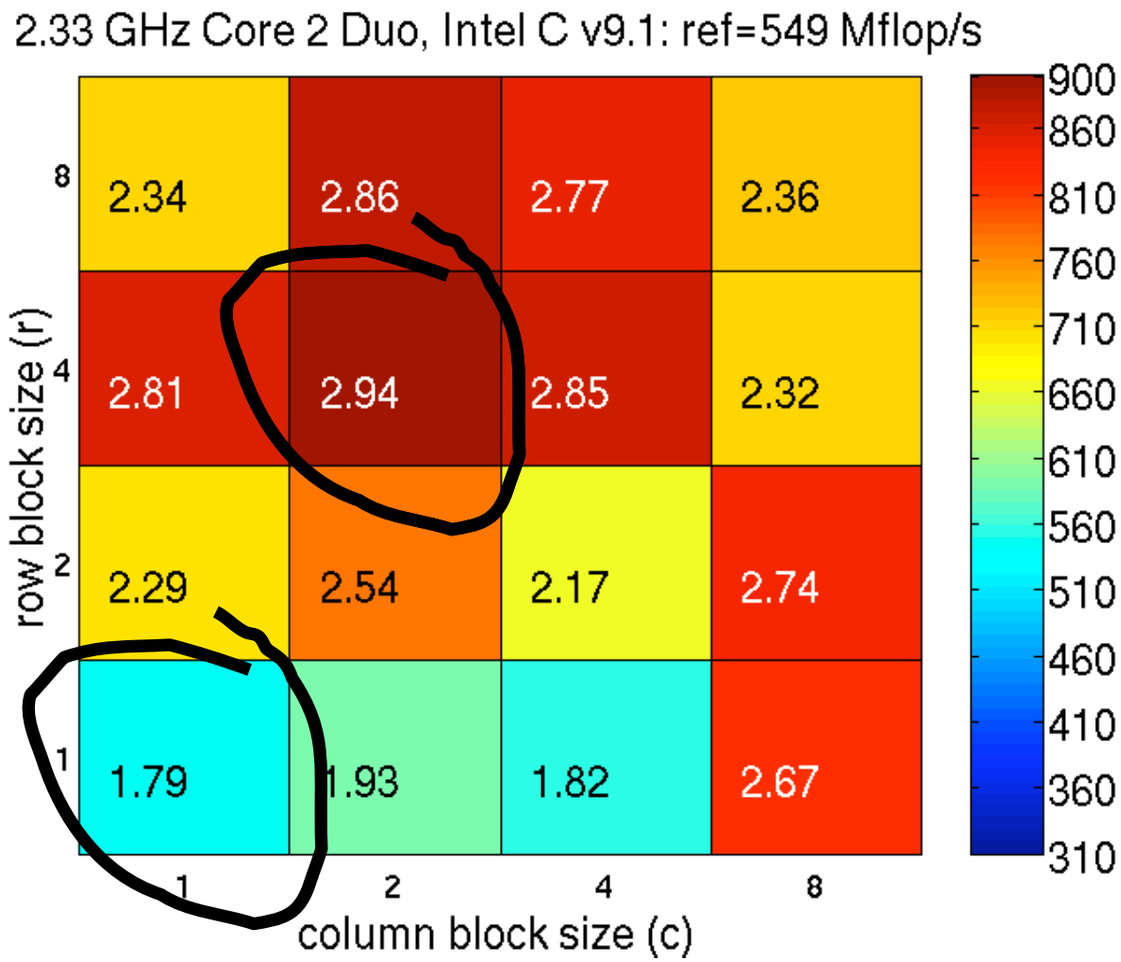
Importance of tuning increases.



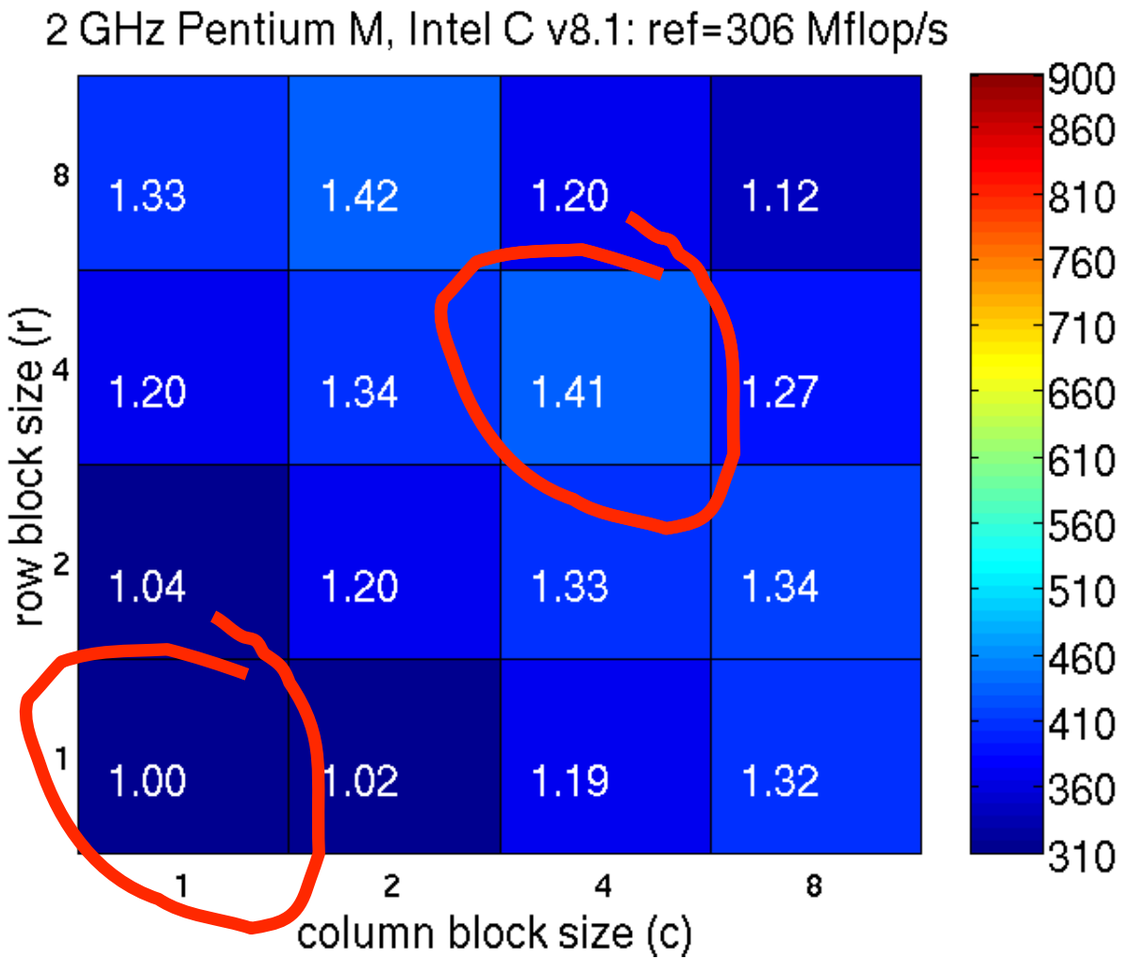


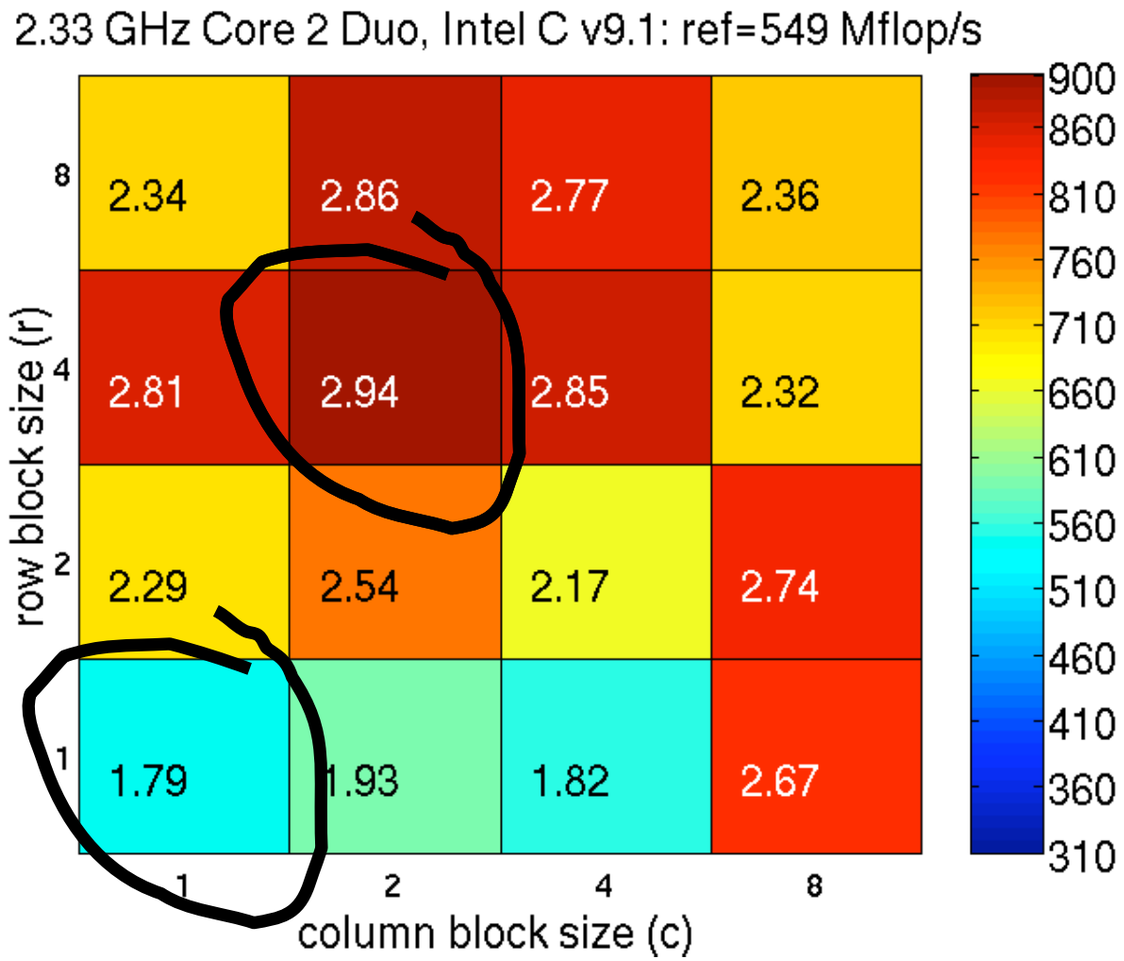
Example 3: Single- to dual-core.





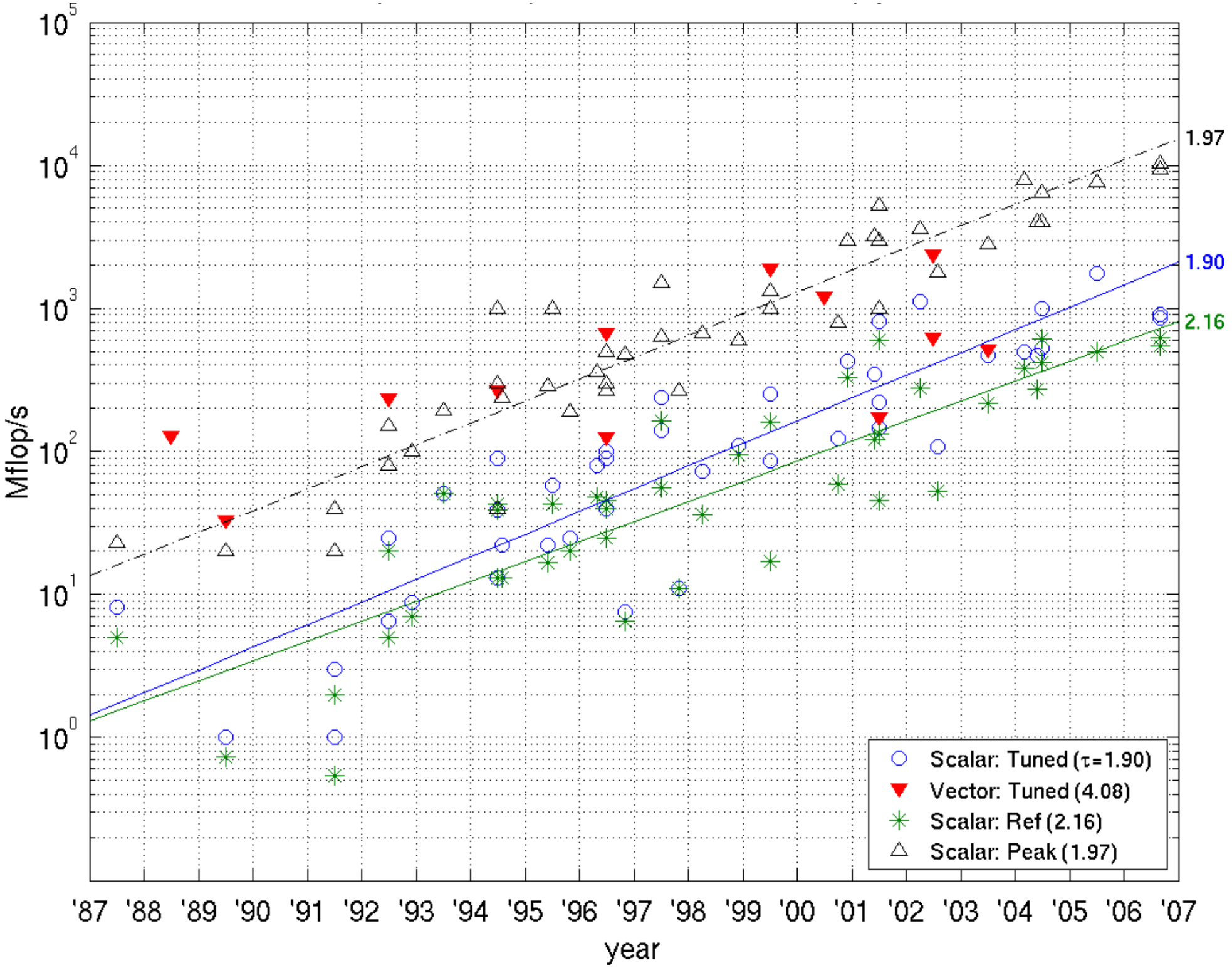
Reference & best improve…



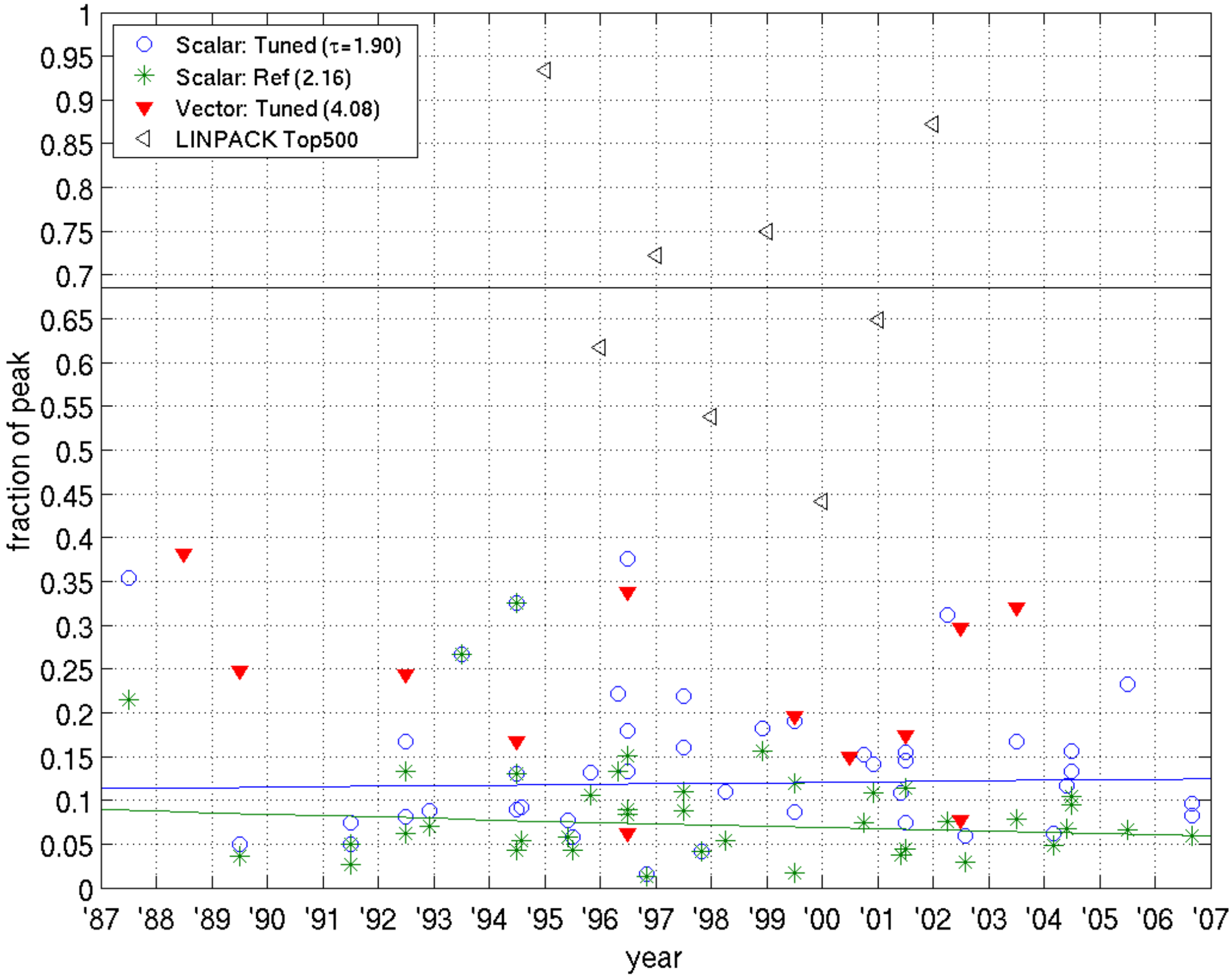


Fraction of peak: 11% → 9.5%

20 years of single-core SpMV performance



20 years of single-core SpMV performance



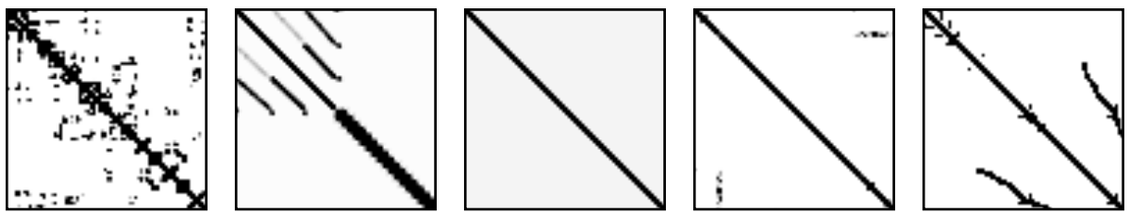
- **Tuning for multicore:**

  Matrices and machines

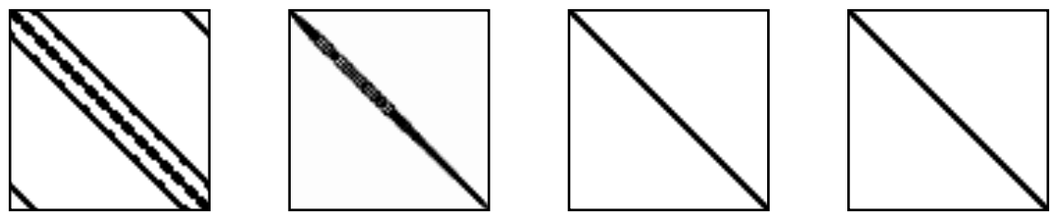
2K x 2K Dense matrix stored in sparse format

Dense

Well Structured
(sorted by nonzeros/row)



Protein | FEM / Spheres | FEM / Cantilever | Wind Tunnel | FEM / Harbor



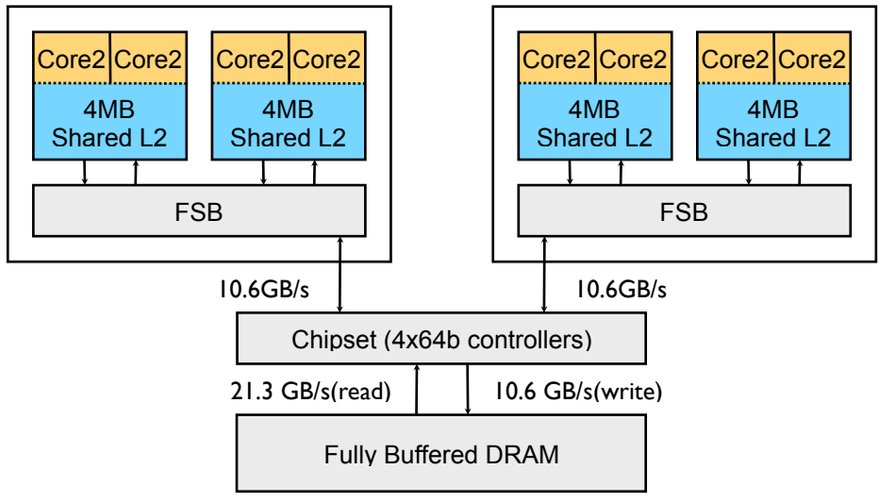
QCD | FEM / Ship | Economics | Epidemiology

Poorly Structured Mix

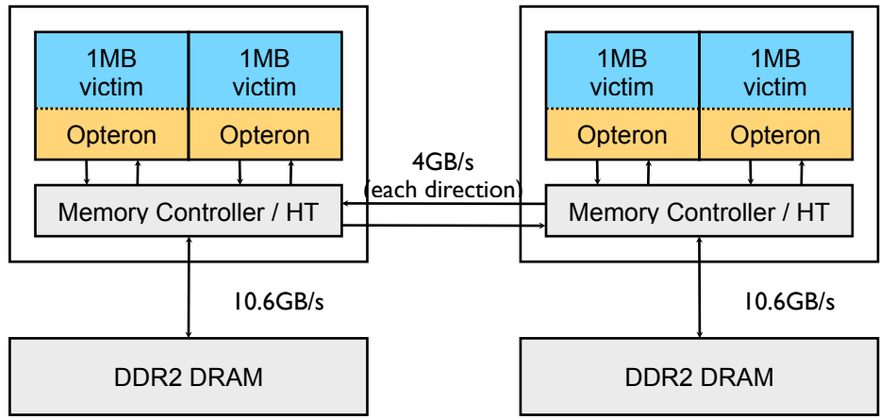
FEM / Accelerator | Circuit | webbase

Extreme Aspect Ratio (linear programming)

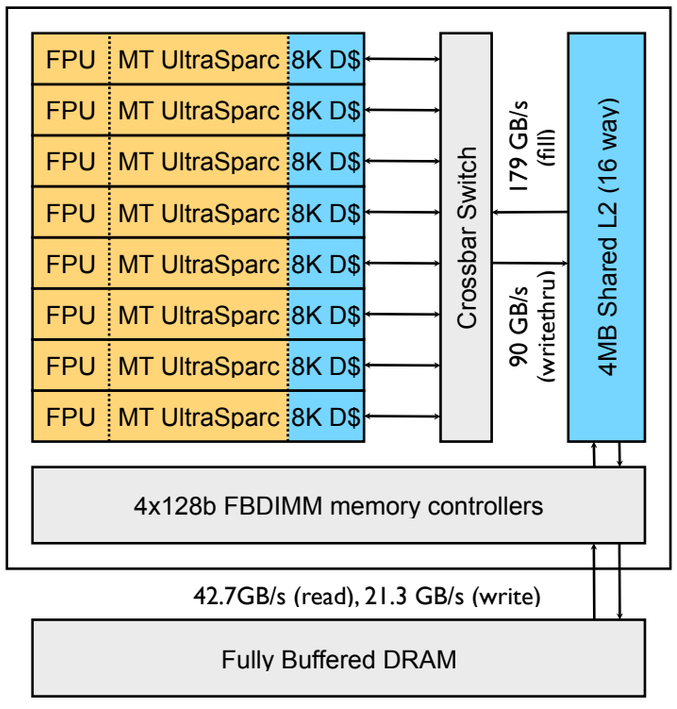
n ~ 2k to 1 M
nnz ~ 2 to 12M
nnz/row ~ 3 to 3k



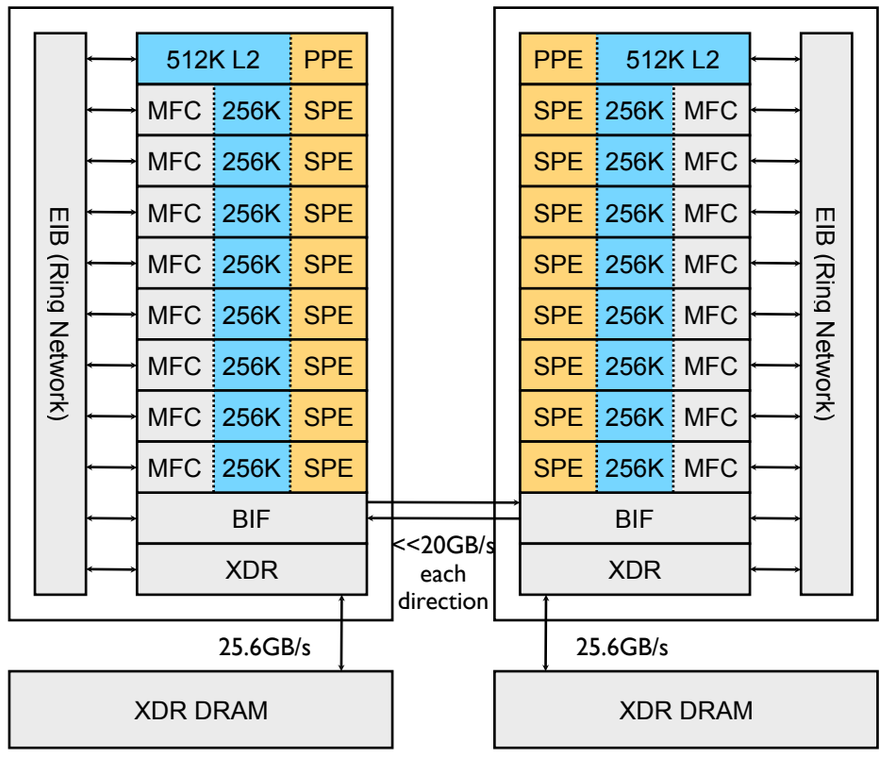
Intel Clovertown (2 x 4)



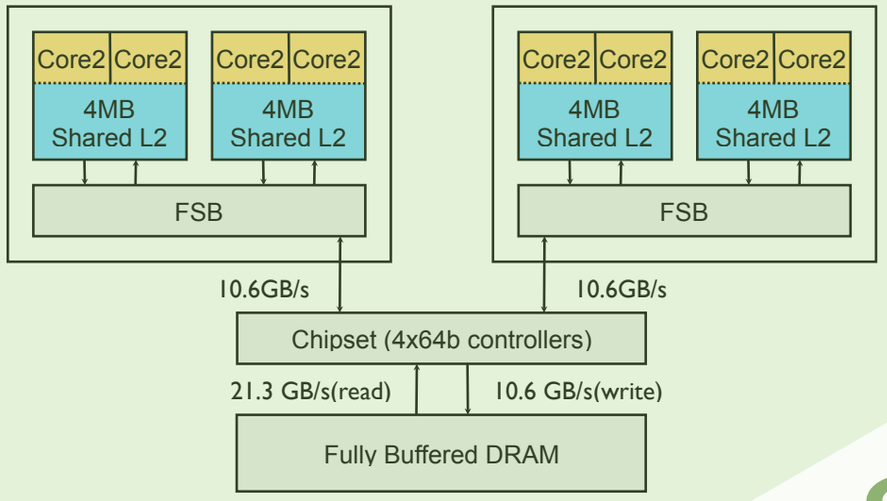
AMD Opteron X2 (2 x 2)



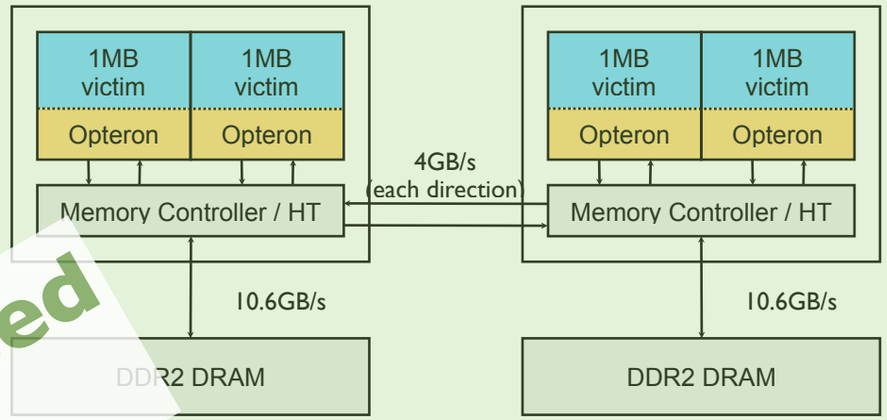
Sun Niagara2 (1 x 8 x 8)



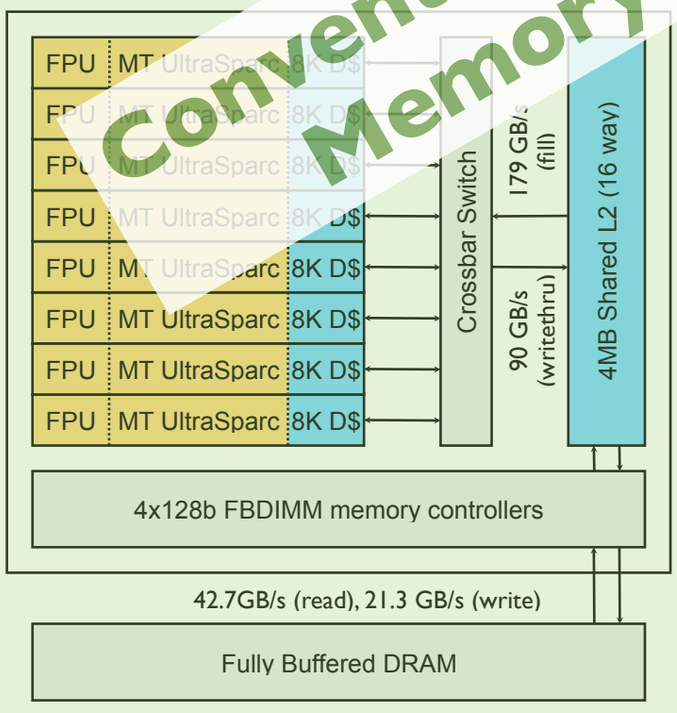
STI Cell Blade (2 x 8)



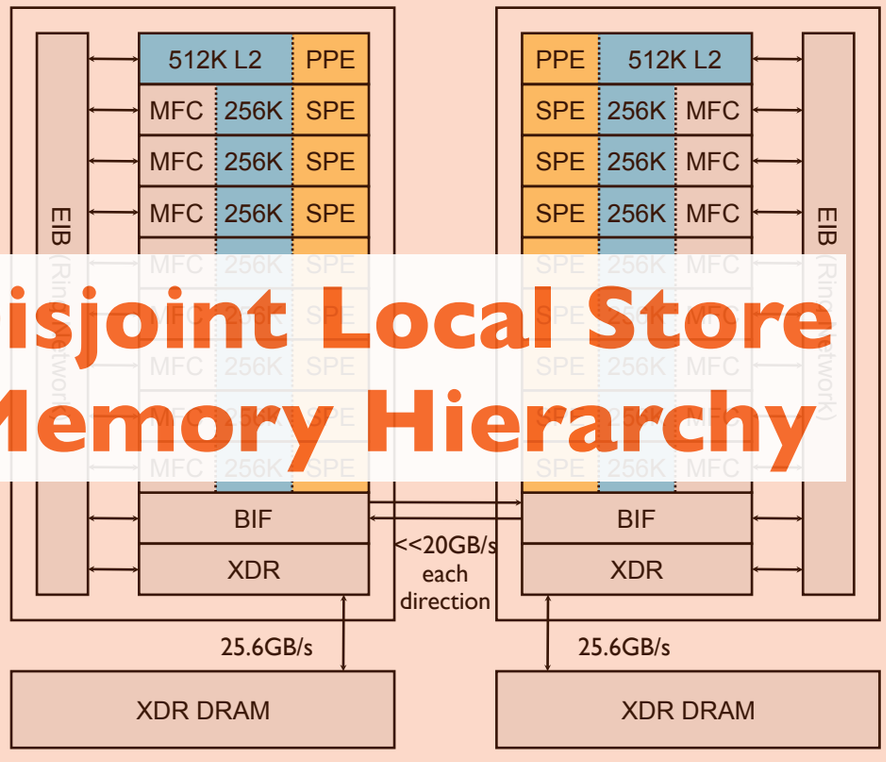
Intel Clovertown



AMD Opteron



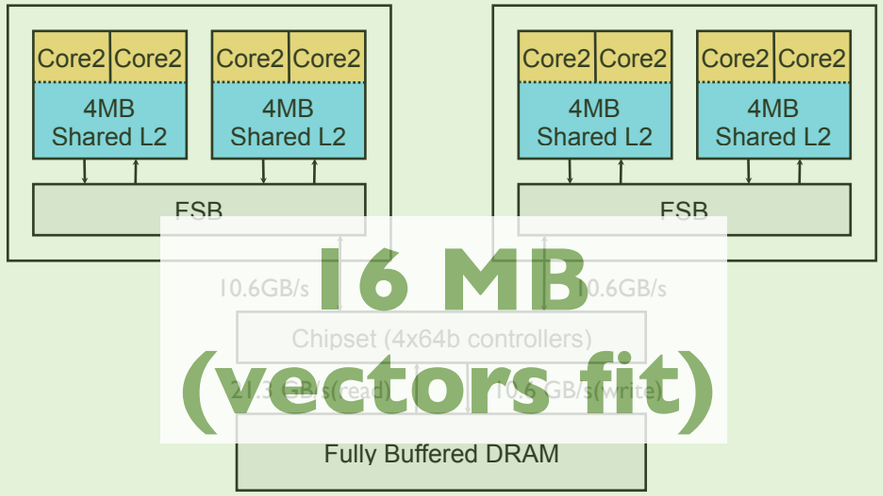
Sun Niagara2



IBM Cell Blade

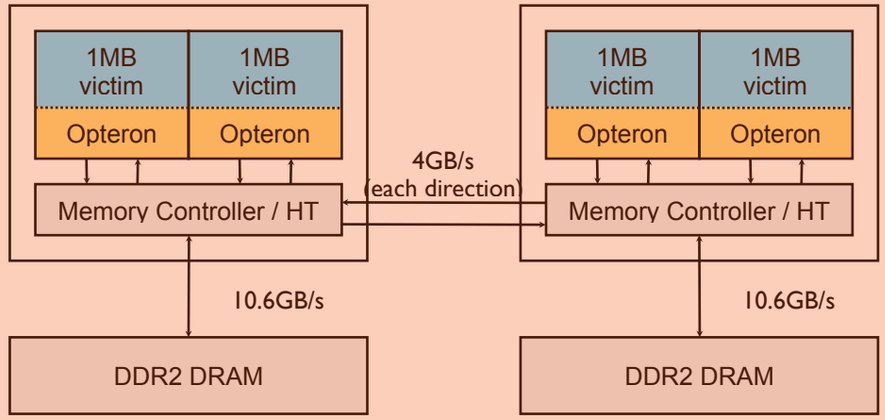
Conventional Cache-based Memory Hierarchy

Disjoint Local Store Memory Hierarchy

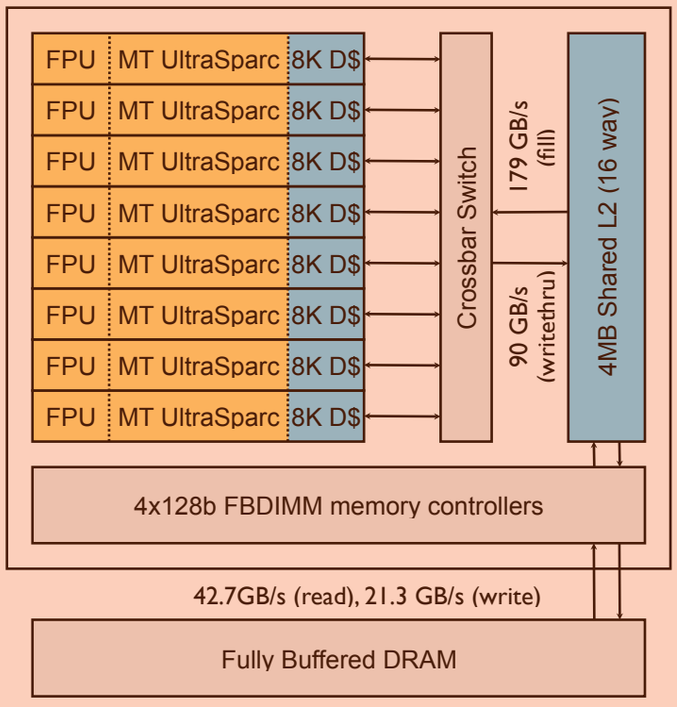


16 MB (vectors fit)

Intel Clovertown

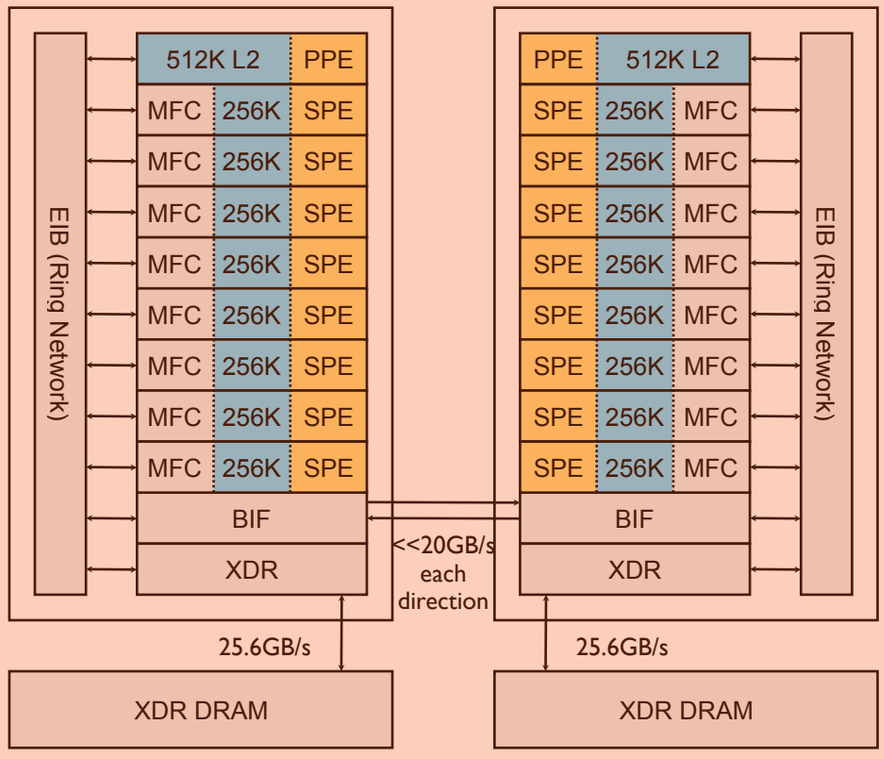


AMD Opteron

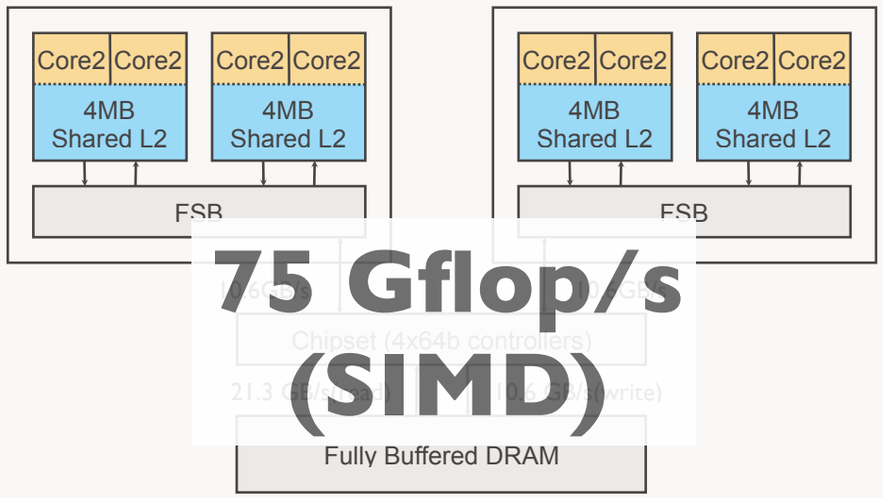


4 MB

Sun Niagara2

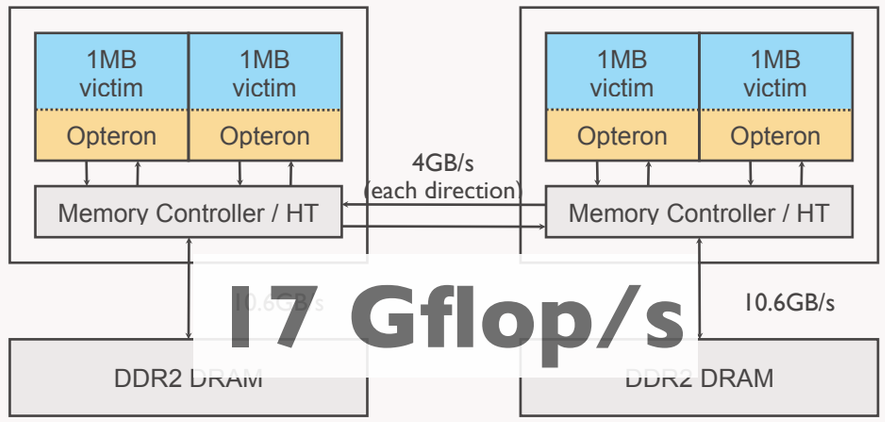


IBM Cell Blade



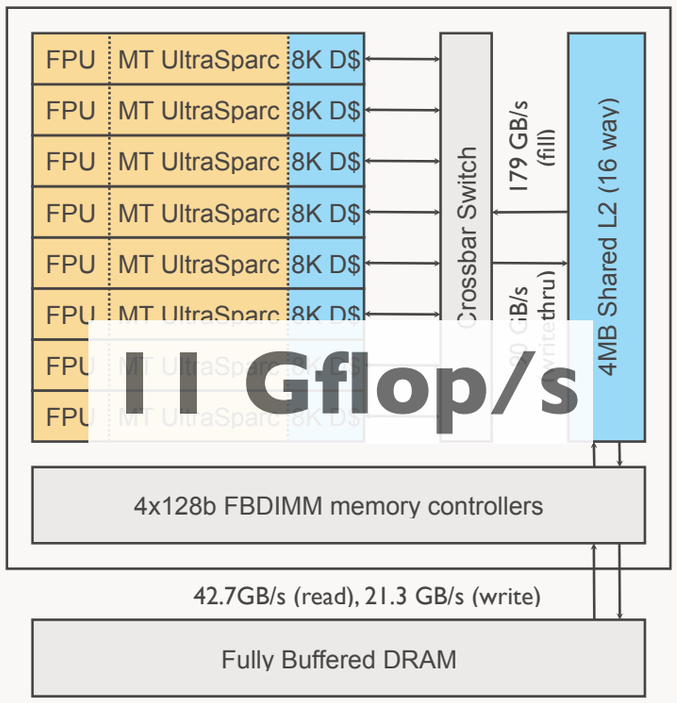
75 Gflop/s (SIMD)

Intel Clovertown



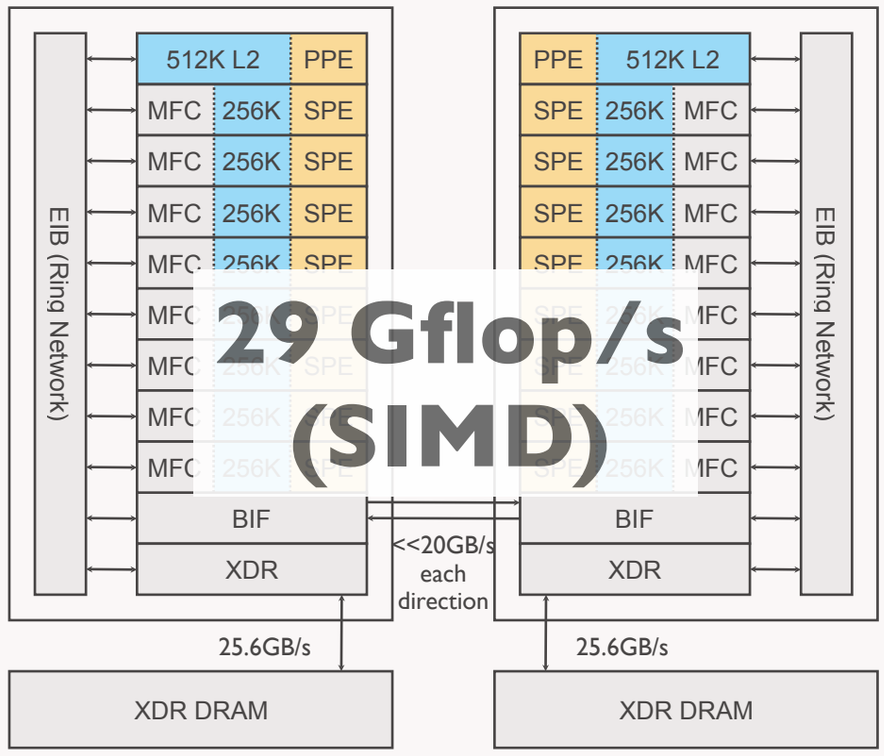
17 Gflop/s

AMD Opteron



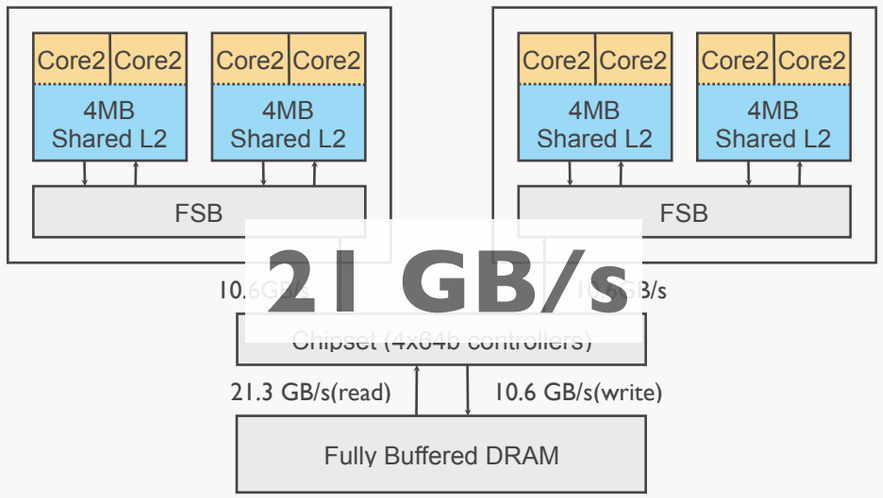
11 Gflop/s

Sun Niagara2

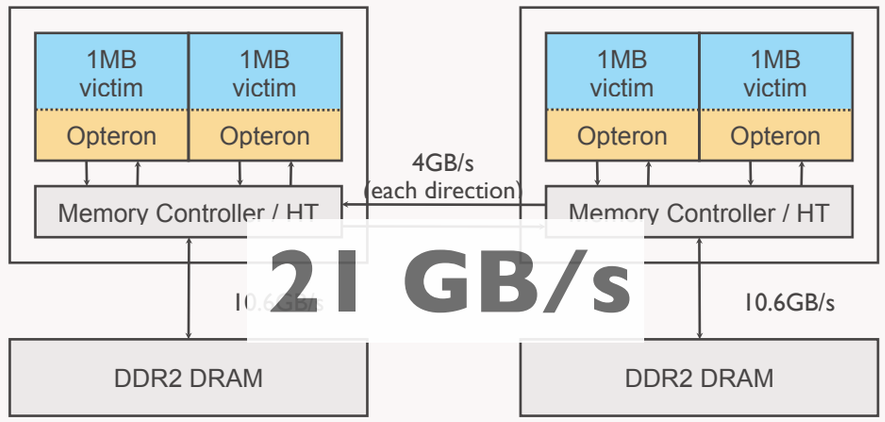


29 Gflop/s (SIMD)

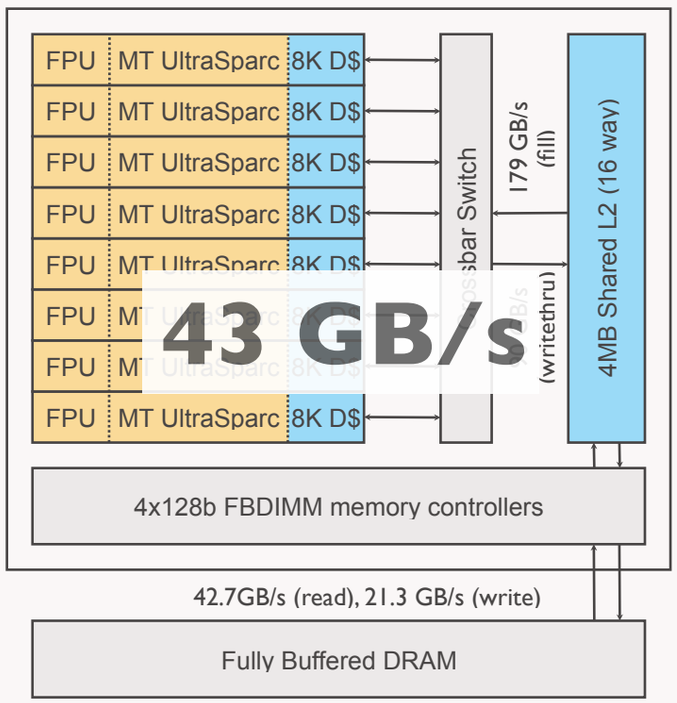
IBM Cell Blade



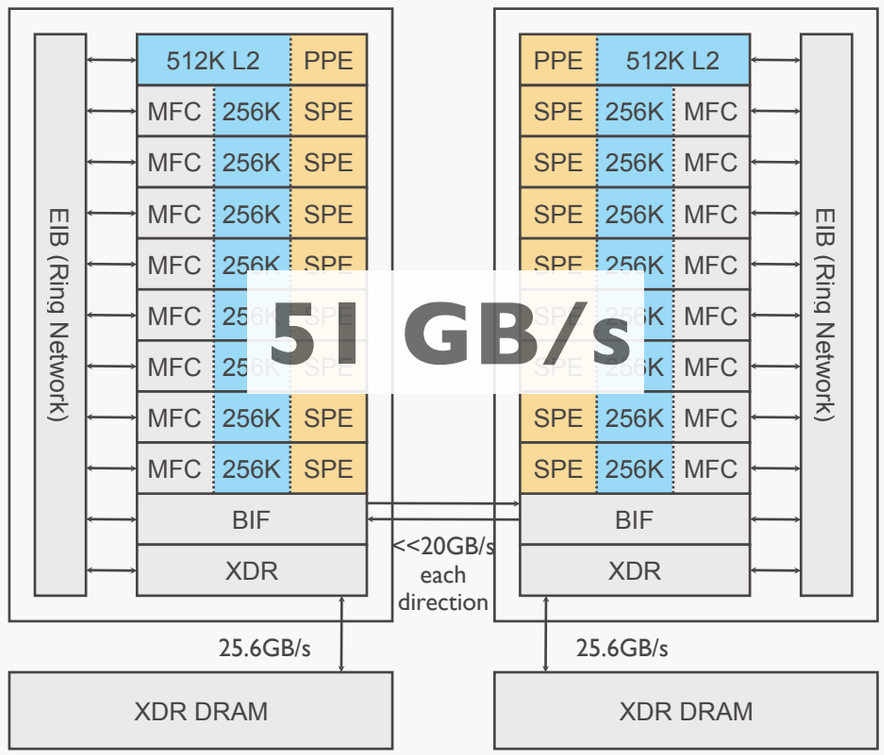
Intel Clovertown



AMD Opteron

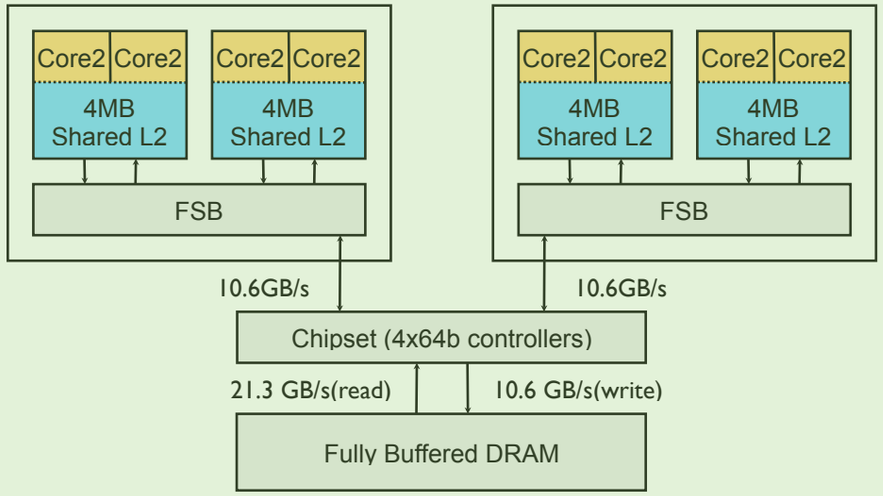


Sun Niagara2



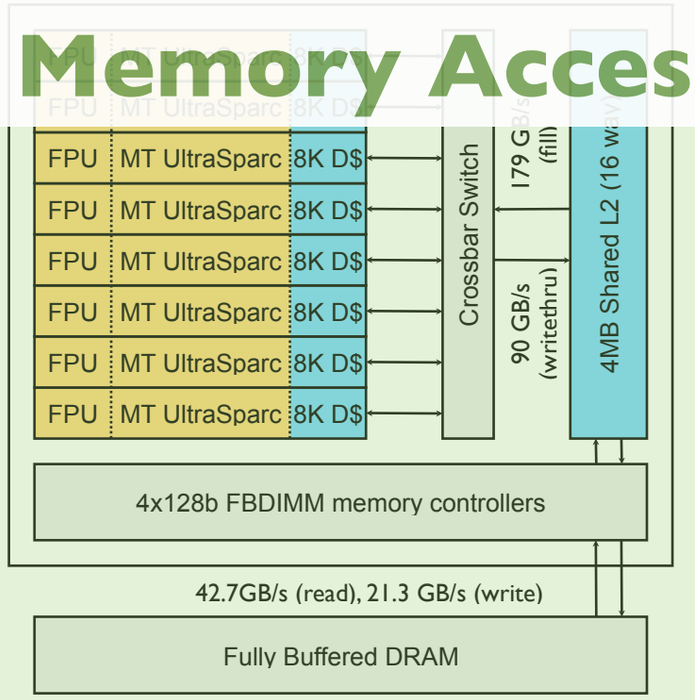
IBM Cell Blade

(Peak read bandwidth)

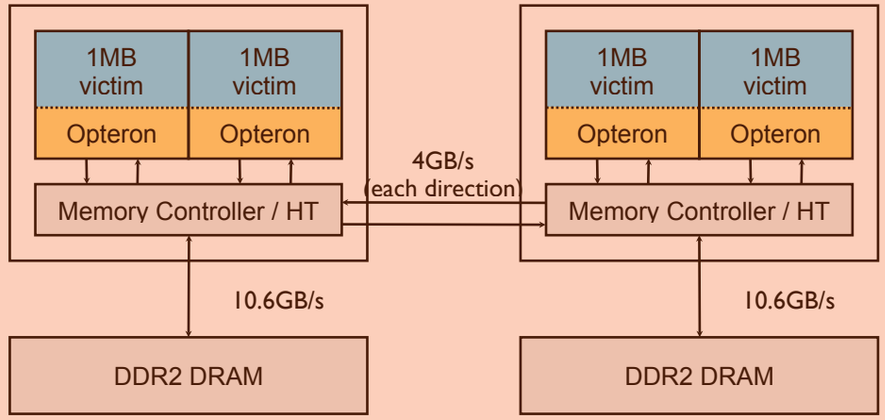


Intel Clovertown

Uniform Memory Access

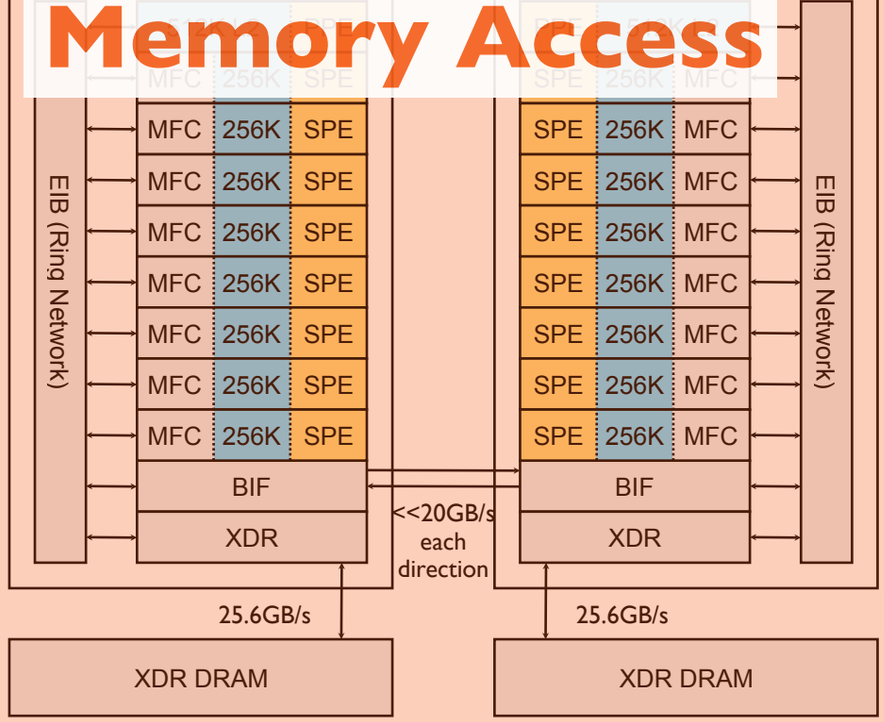


Sun Niagara2



AMD Opteron

Non-uniform Memory Access

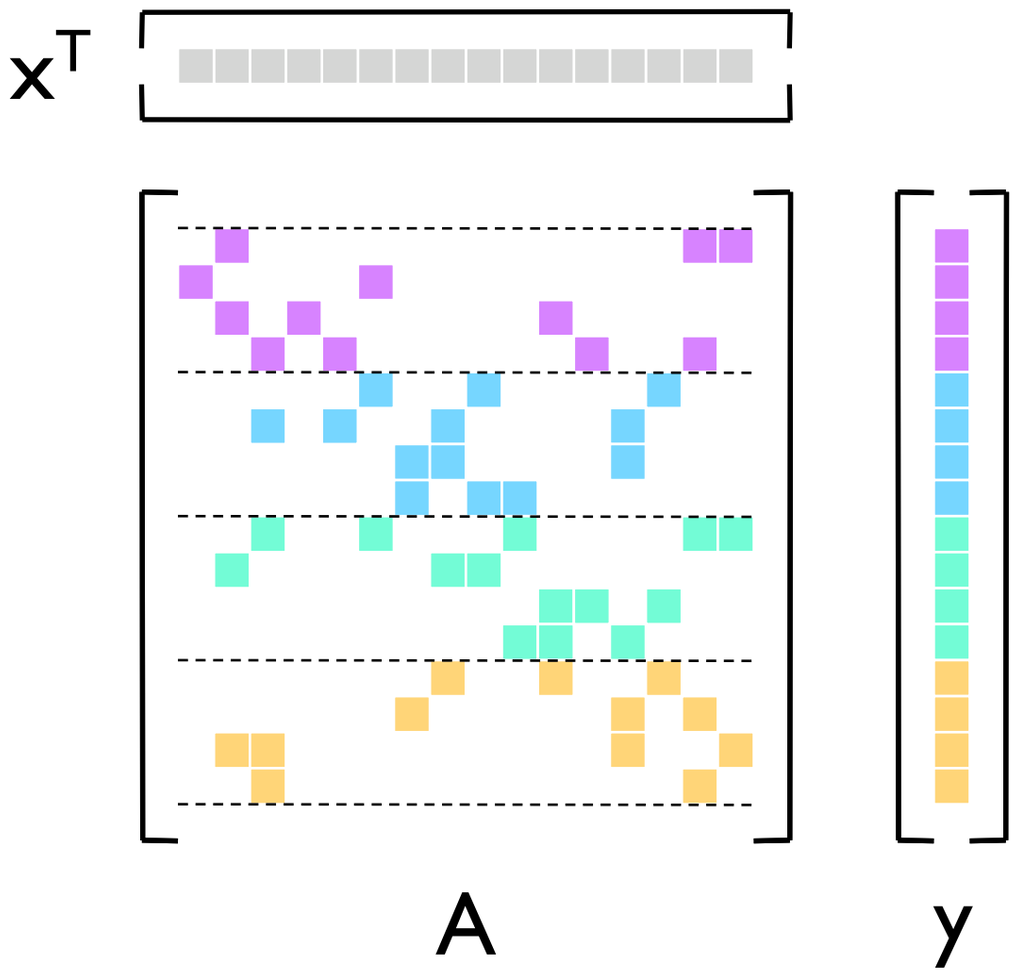


IBM Cell Blade

(Peak read bandwidth)

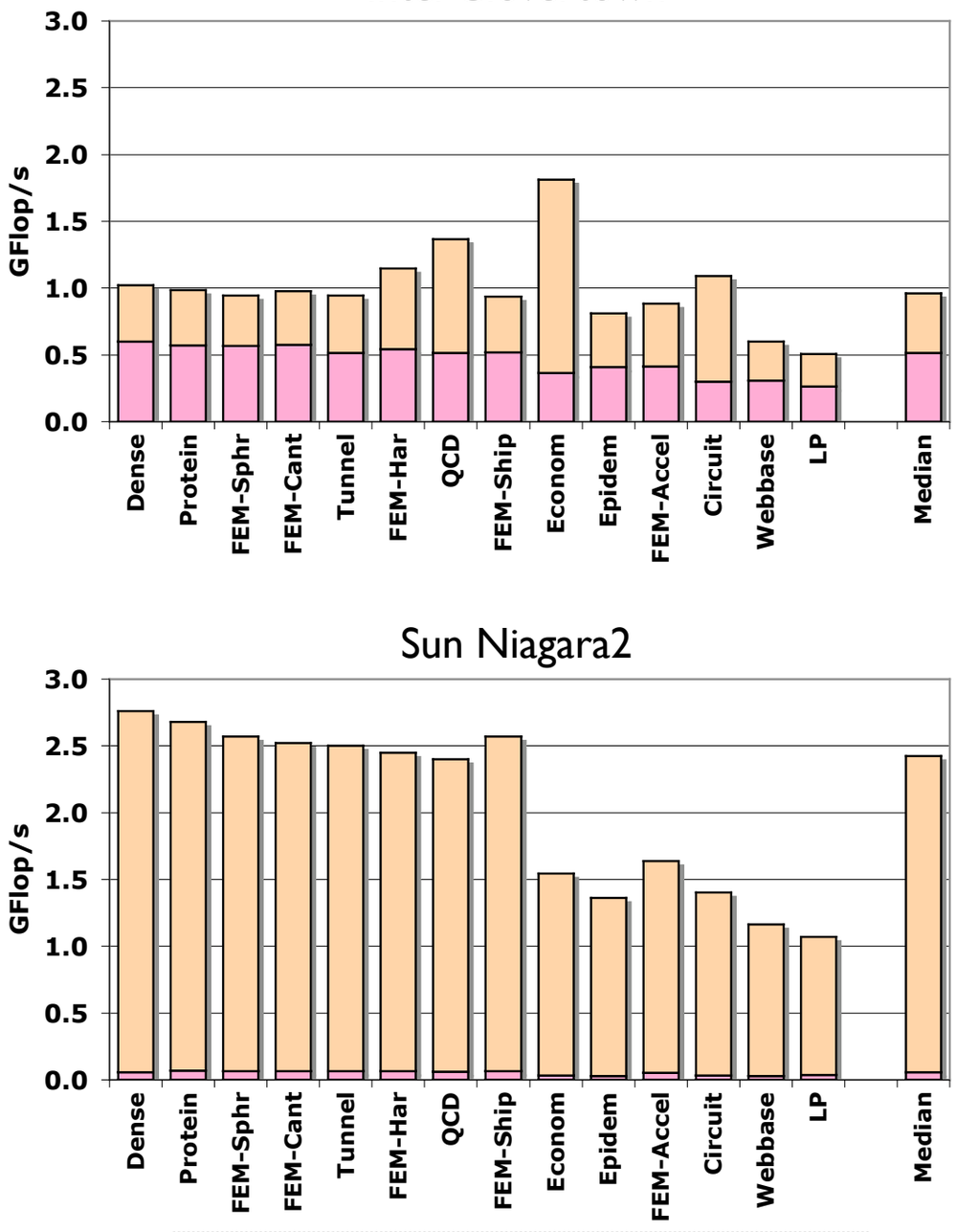
- **Naïve codes on cache-based machines**
  - Serial
    - Compressed sparse row (CSR) storage
    - "Vanilla" C
    - "Best" compiler options, with some search
  - PThreads
    - 1-D block rows, in CSR
    - Roughly equal nnz / block
    - Barrier before & after SpMV

Parallelization

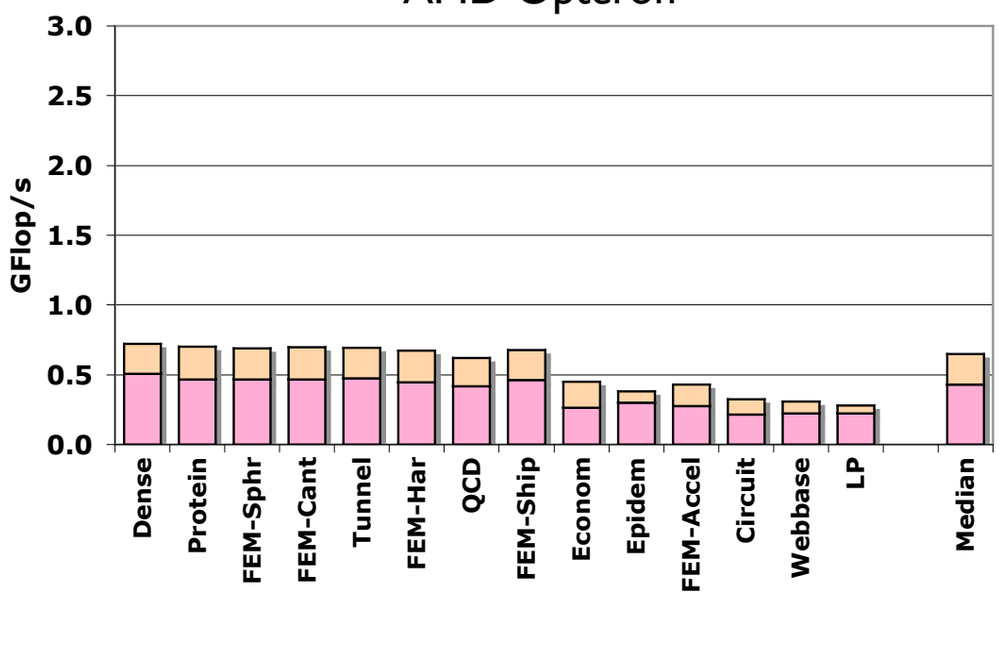


- 1-D block rows, in CSR
- Roughly equal nnz / block
- Barrier before & after SpMV

8 cores ⇒ 1.9x



4 cores ⇒ 1.5x

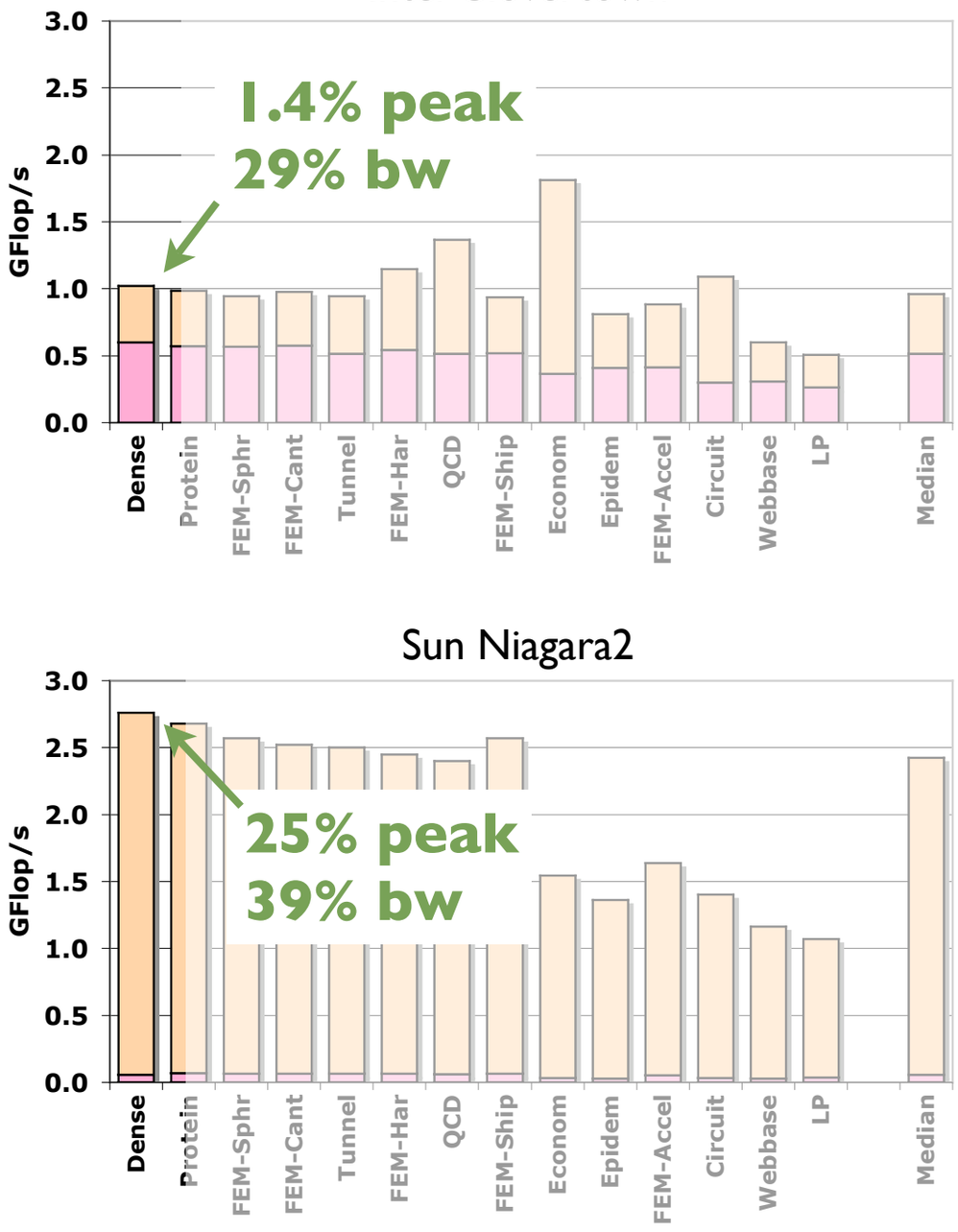


64 threads ⇒ 41x

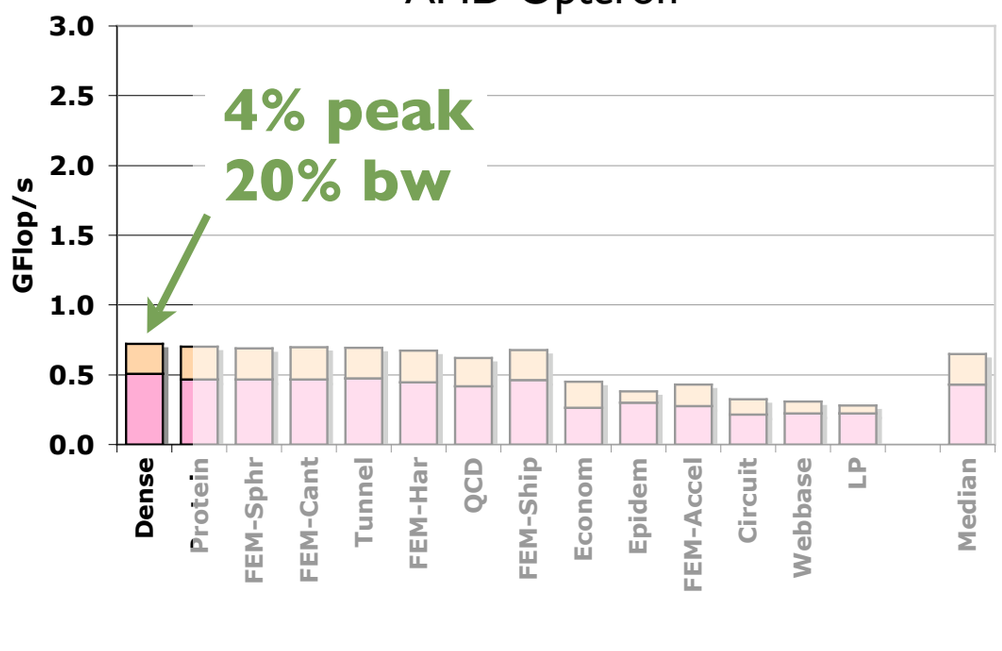
Naïve Pthreads

Naïve Single Thread

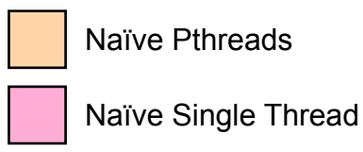
8 cores ⇒ 1.9x



4 cores ⇒ 1.5x

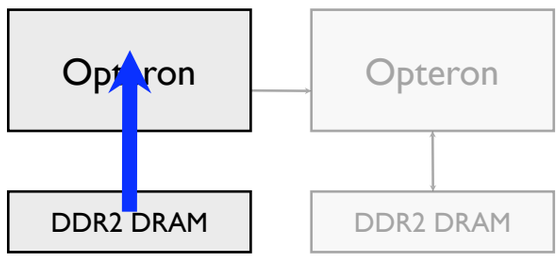


64 threads ⇒ 41x

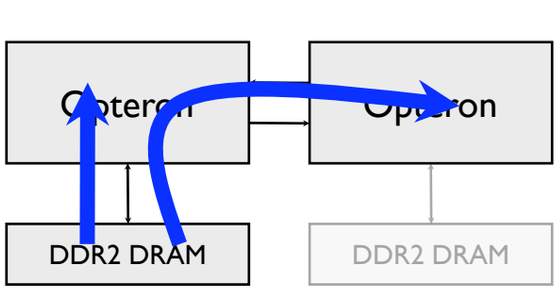


- **Multicore optimizations**
  - Bind
  - Prefetch
  - Compress
  - Block

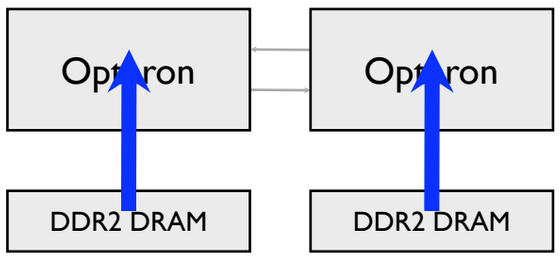
Bind



Single Thread



Multiple Threads,
One memory controller



Multiple Threads,
Both memory controllers

Idea:

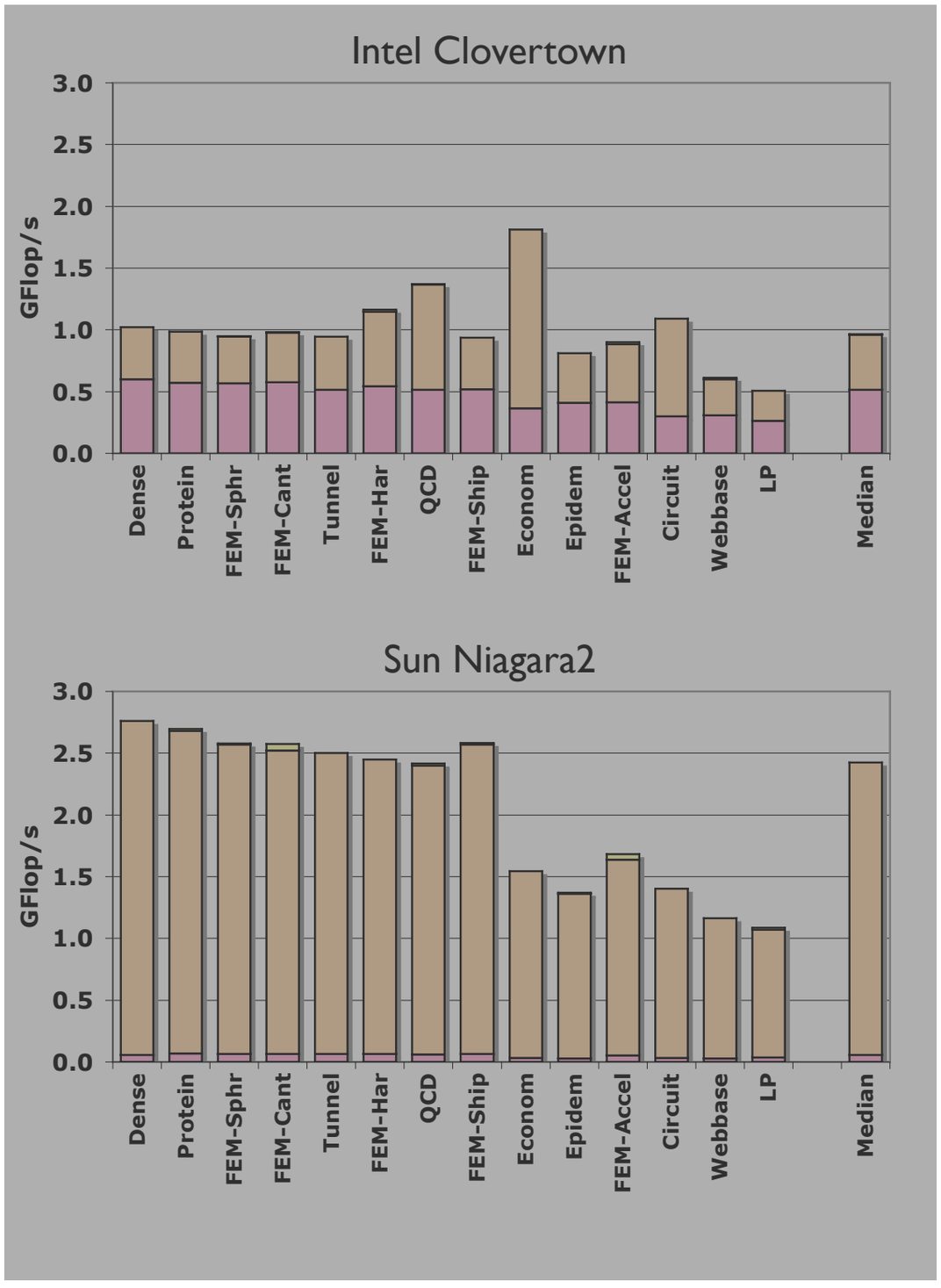
Exploit NUMA, affinity

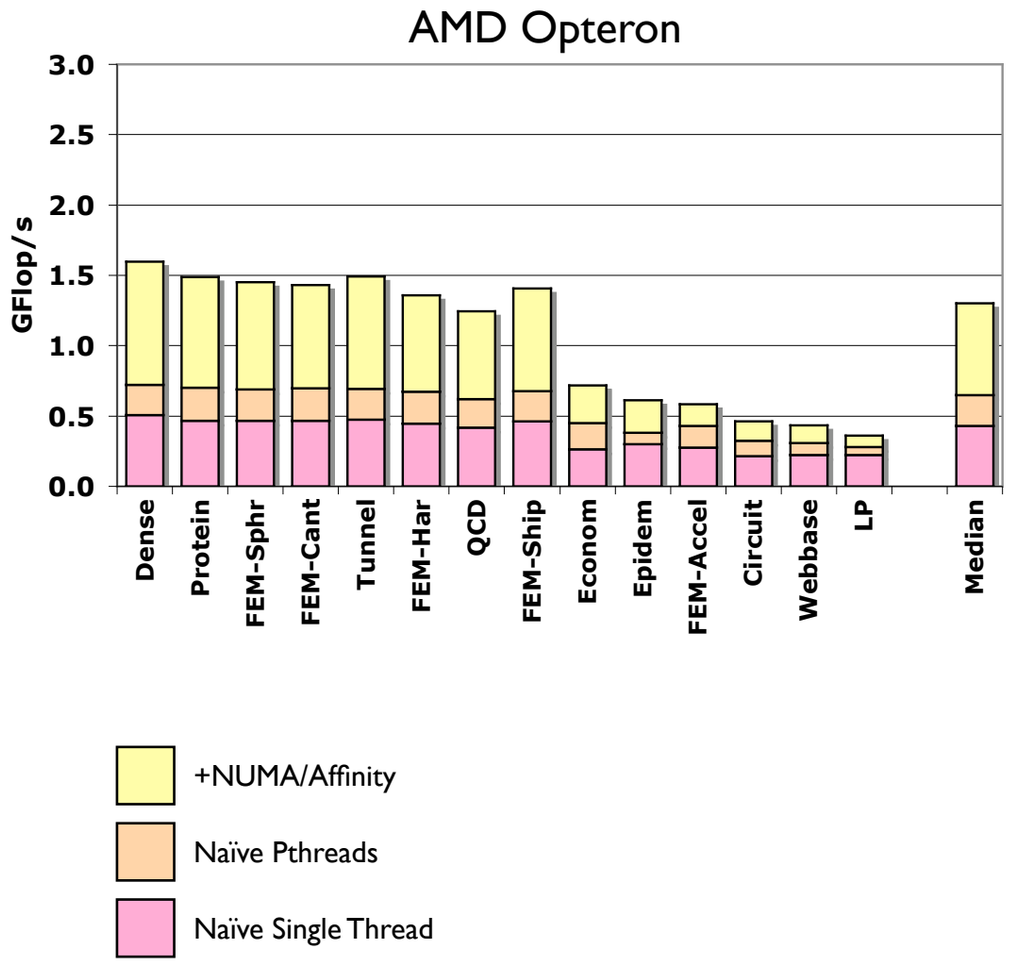
Bind adjacent blocks to adjacent cores using

Cell: libnuma

Opteron: Custom, based on Linux scheduler

Bind


Intel Clovertown
GFlop/s
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median
Sun Niagara2
GFlop/s
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

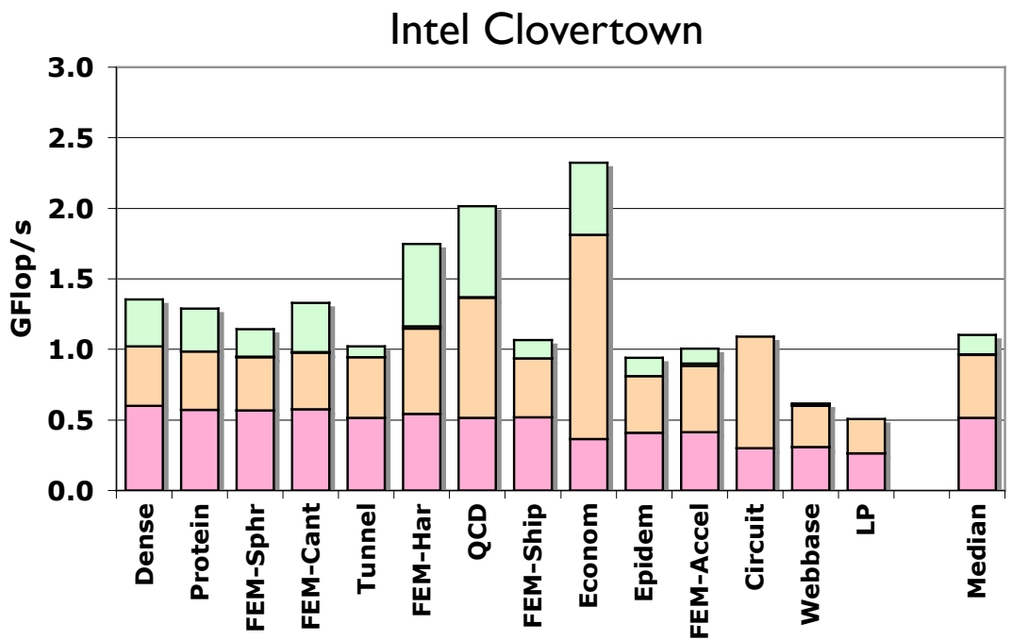

AMD Opteron
GFlop/s
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

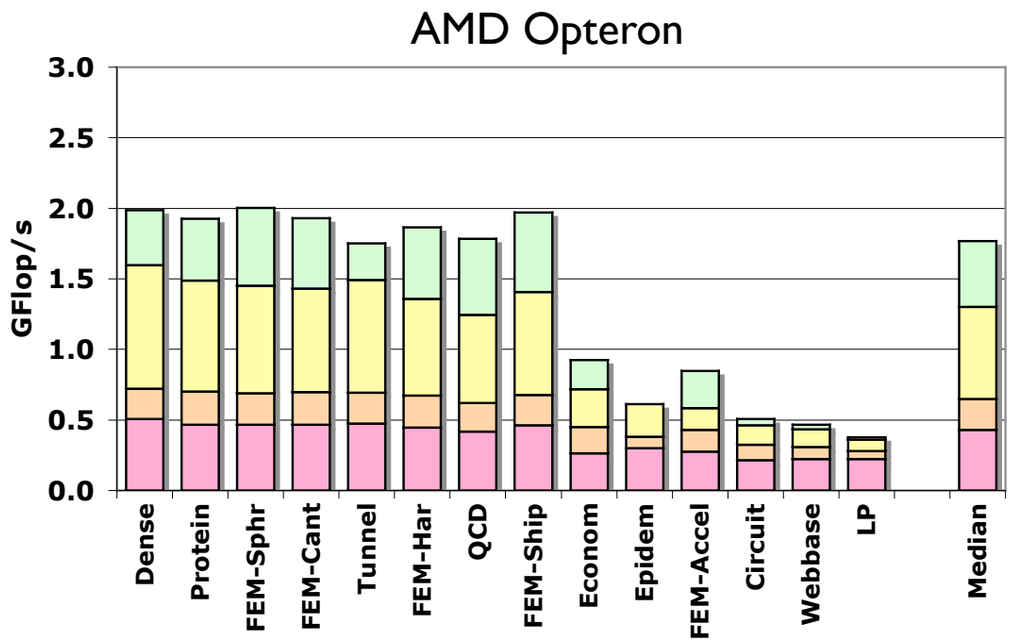
+NUMA/Affinity
Naïve Pthreads
Naïve Single Thread

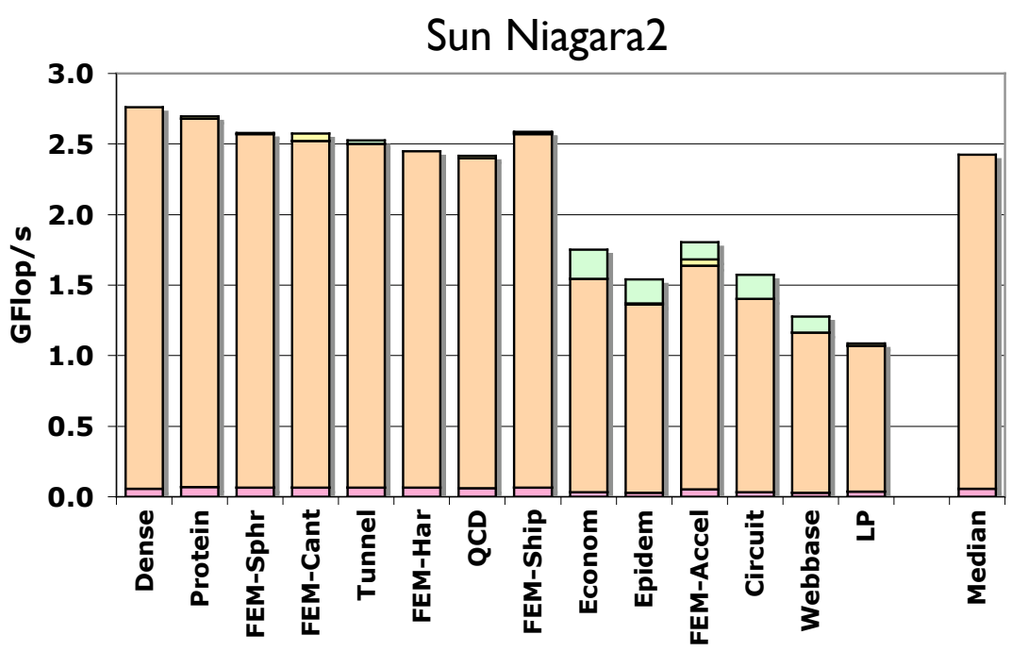
Prefetch

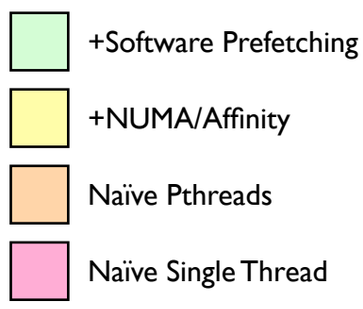
- Search over
  - all distances from 0 (none) to 1 KB
  - all combinations of target
- x86: Prefetch into L1, indicating temporal mode
- Niagara2: L2 only, but still works

Prefetch


Intel Clovertown
GFlop/s
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


AMD Opteron
GFlop/s
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

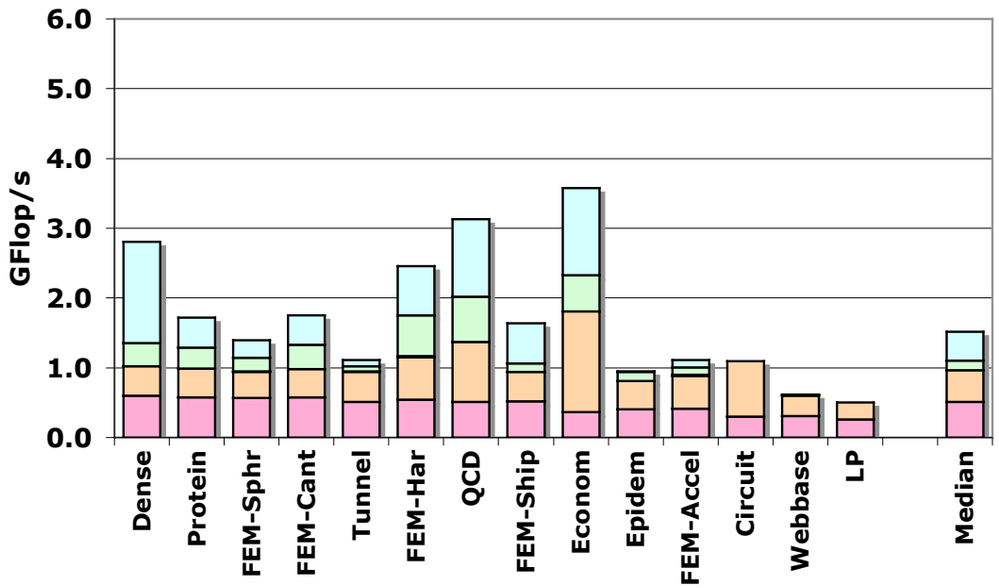

Sun Niagara2
GFlop/s
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

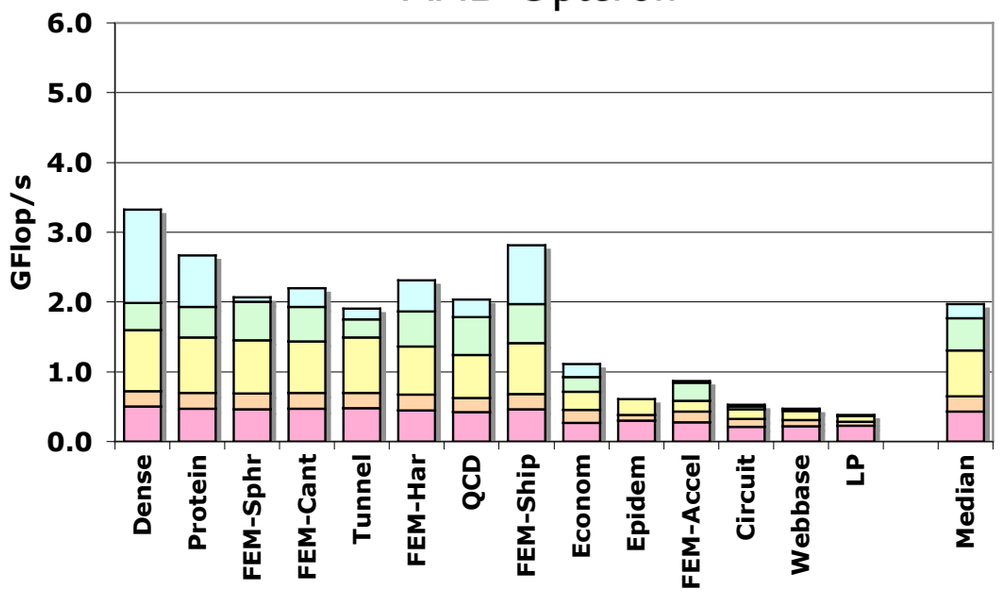

+Software Prefetching
+NUMA/Affinity
Naïve Pthreads
Naïve Single Thread

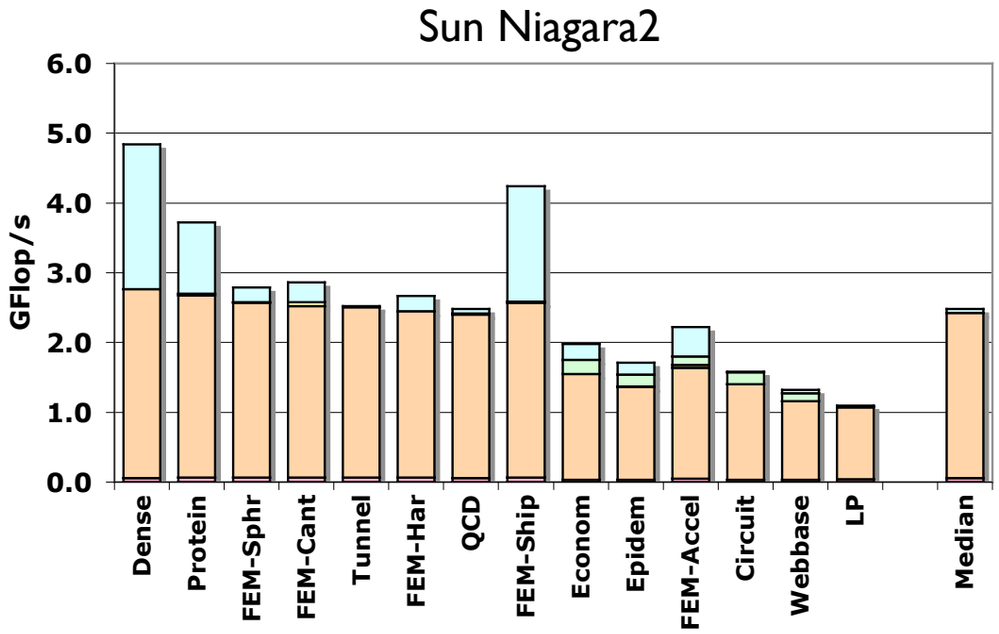
Compress

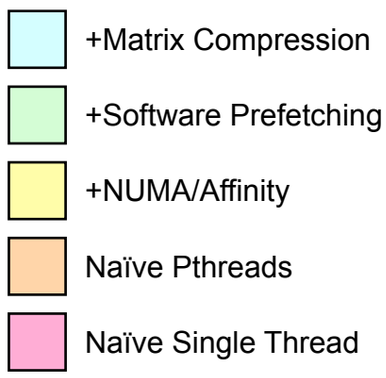
- Intuition: Reading matrix dominates run-time
- Reduce integer index volume and size
  - Power of 2 register blocking
  - BCSR / BCOO formats
  - 16-bit vs. 32-bit indices

Compress

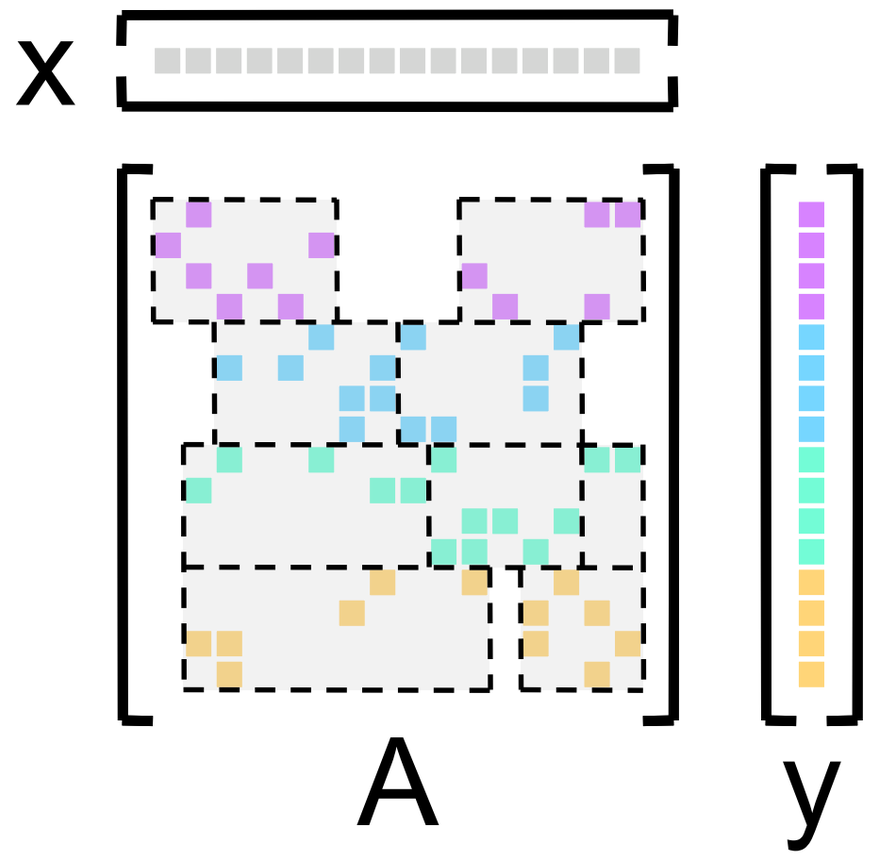

Intel Clovertown
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


AMD Opteron
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


Sun Niagara2
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


+Matrix Compression
+Software Prefetching
+NUMA/Affinity
Naïve Pthreads
Naïve Single Thread

Block (Cache & TLB)



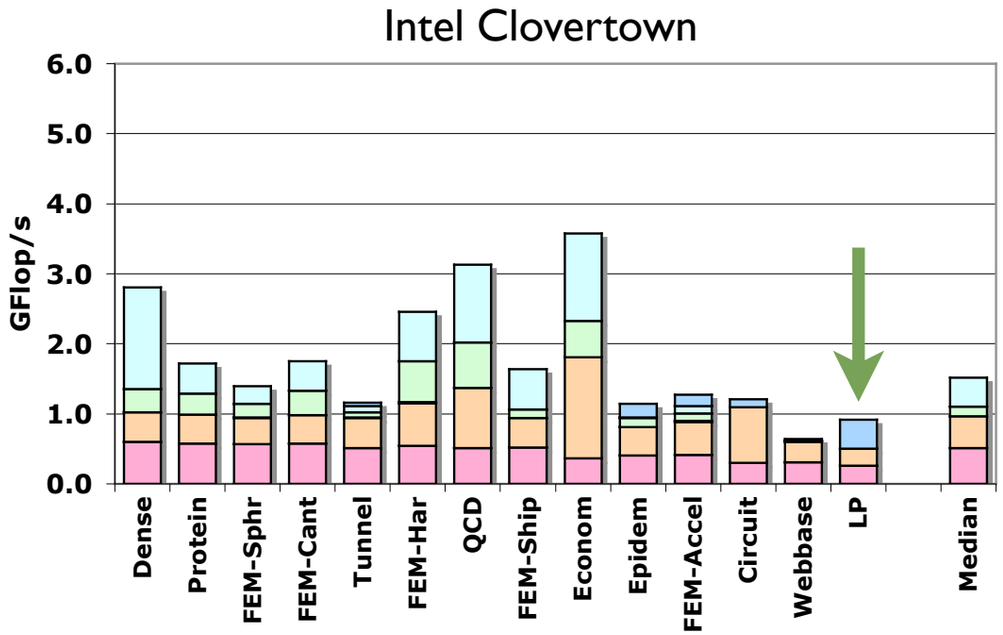
Like dense case,

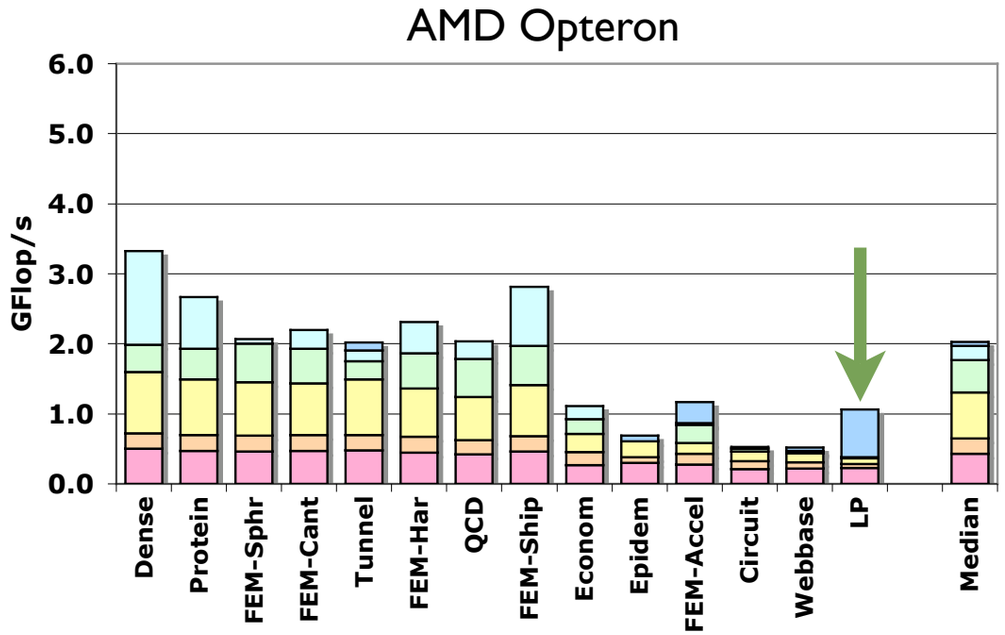
localizes ‘x’ loads for reuse.

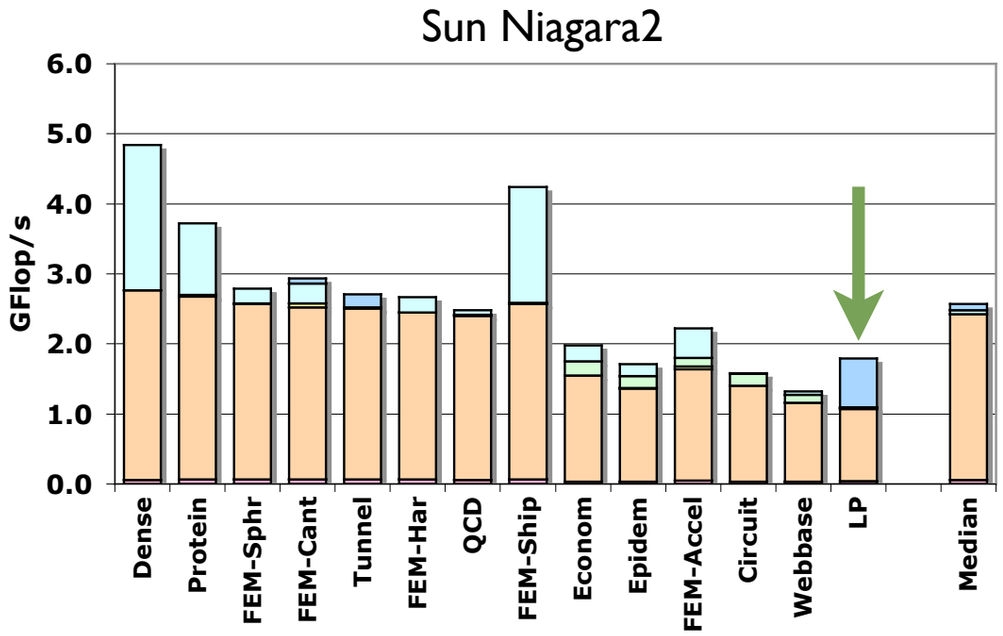
Partition into cache blocks.

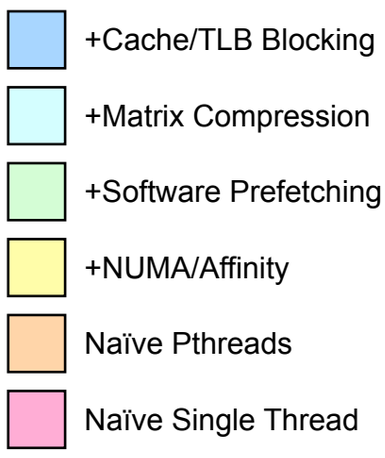
Tune each block.

Block


Intel Clovertown
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


AMD Opteron
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


Sun Niagara2
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

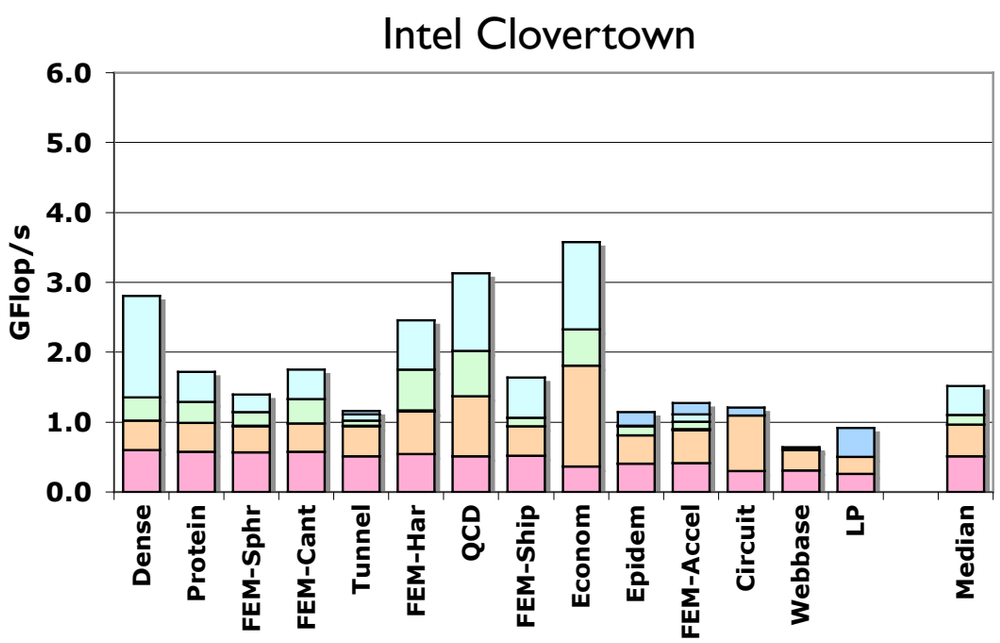

+Cache/TLB Blocking
+Matrix Compression
+Software Prefetching
+NUMA/Affinity
Naïve Pthreads
Naïve Single Thread

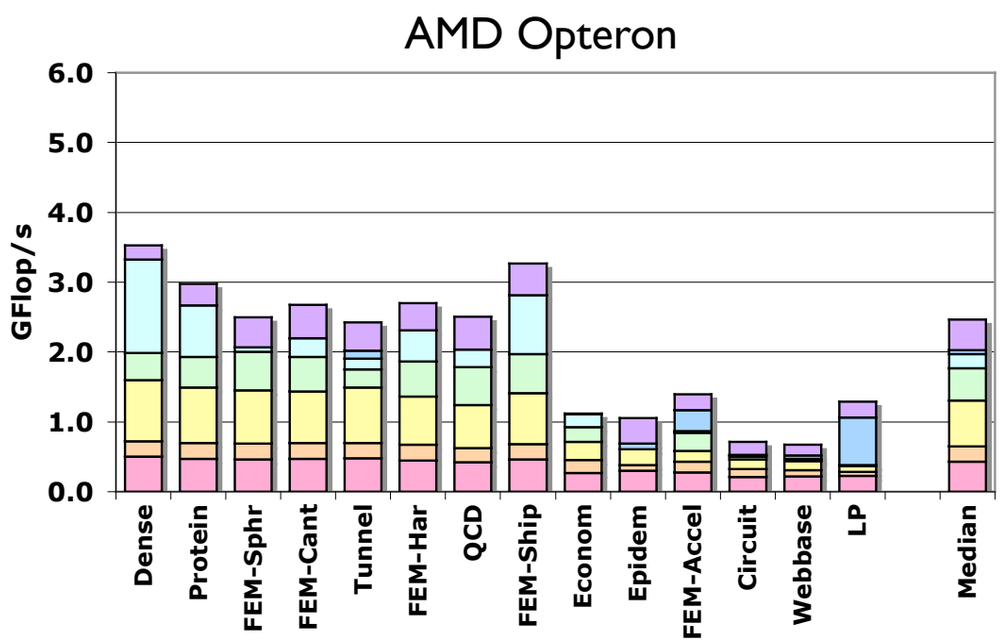
- Multicore optimizations
  - Bind
  - Prefetch
  - Compress
  - Block
  - **Buy (more memory)**

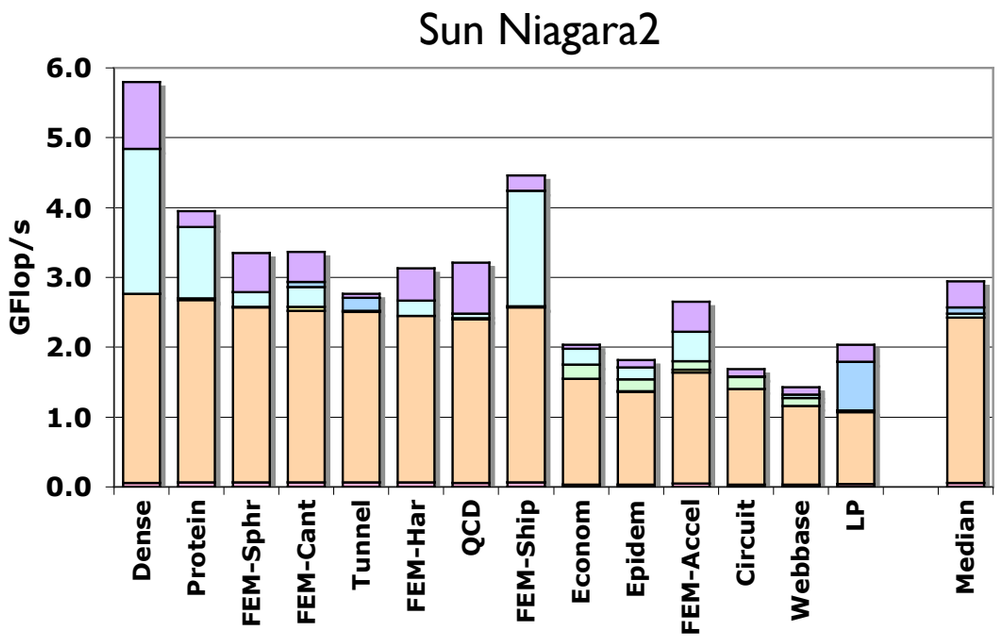
Banks, ranks, and DIMMs

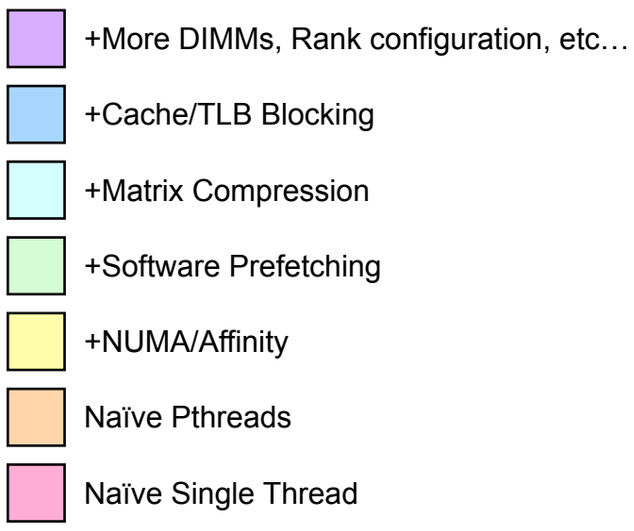
- More threads ⇒ more streams to memory
  - Memory controller reorders some no. of requests
  - Addressing/bank conflicts more likely
- Idea: Add more DIMMs, configure ranks
- (Our Clovertown was fully populated)

Buy (DIMMs)


Intel Clovertown
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

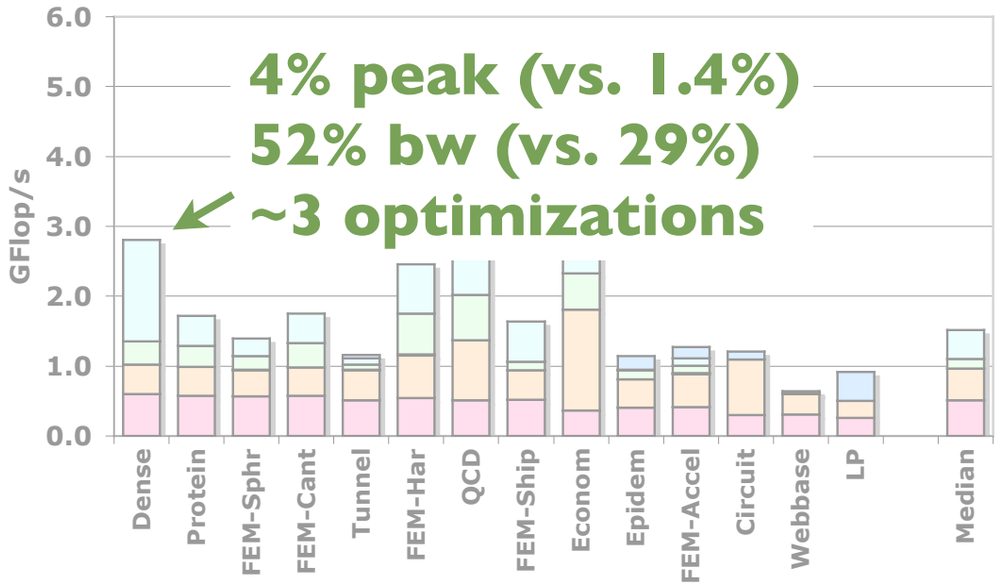

AMD Opteron
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


Sun Niagara2
GFlop/s
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

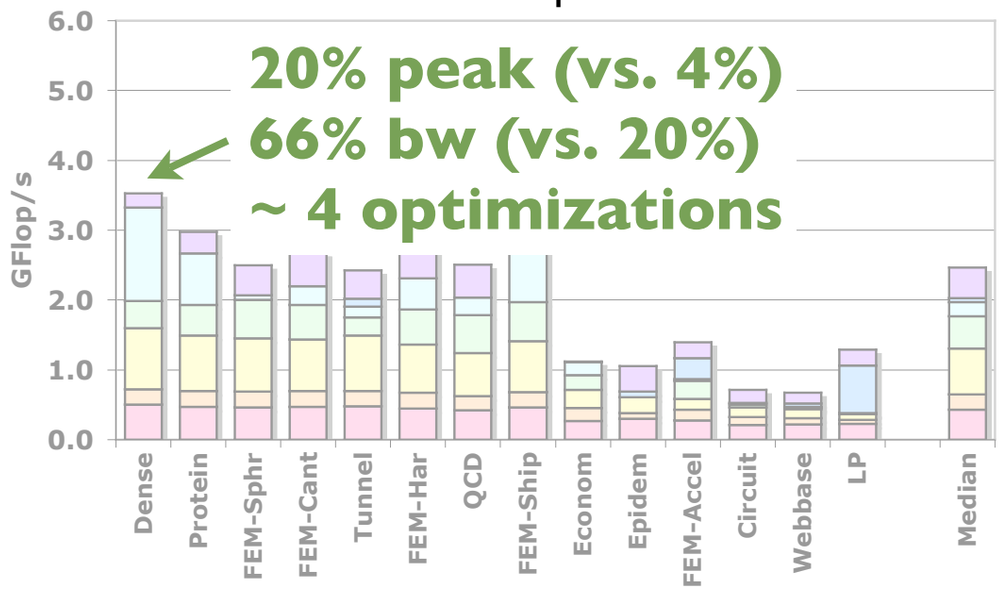

+More DIMMs, Rank configuration, etc…
+Cache/TLB Blocking
+Matrix Compression
+Software Prefetching
+NUMA/Affinity
Naïve Pthreads
Naïve Single Thread

Where do we stand?

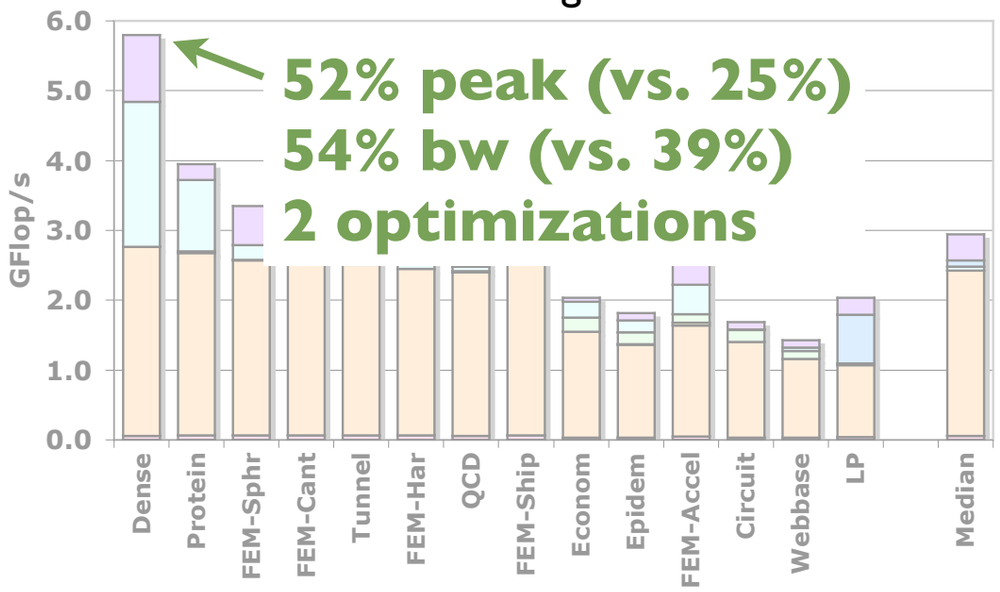
Intel Clovertown

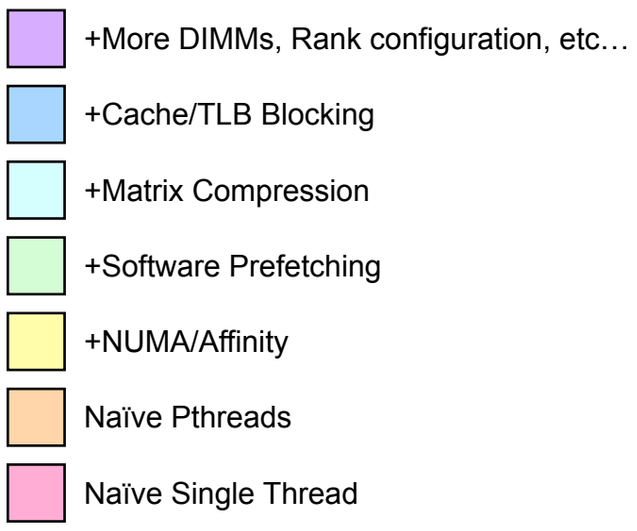


AMD Opteron



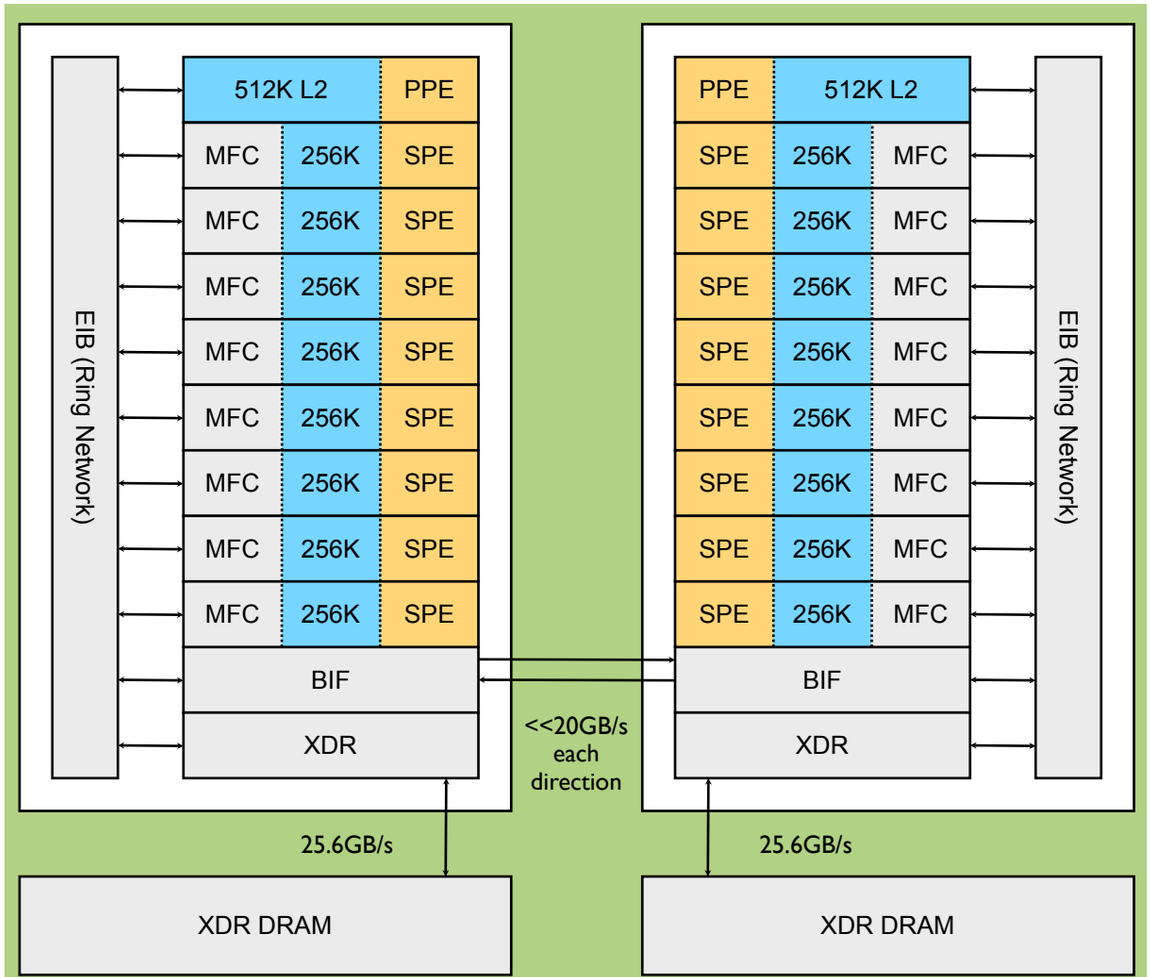
Sun Niagara2





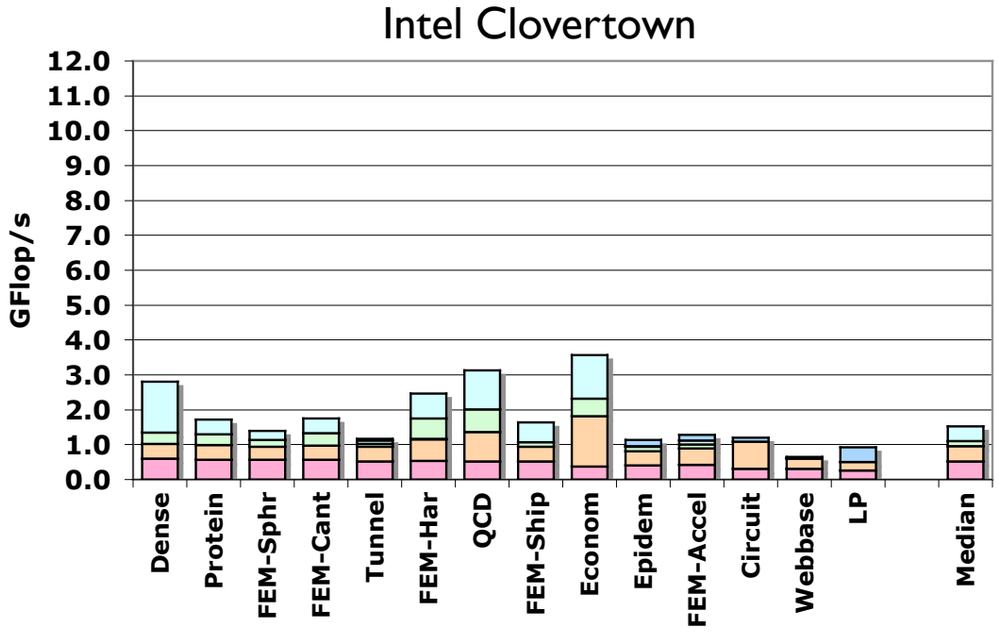
- Cell
  - Consider double-precision, even if slow (14x)
  - Cache blocking ⇒ Local-store blocking
  - Forced 2x1 blocking ⇒ extra traffic

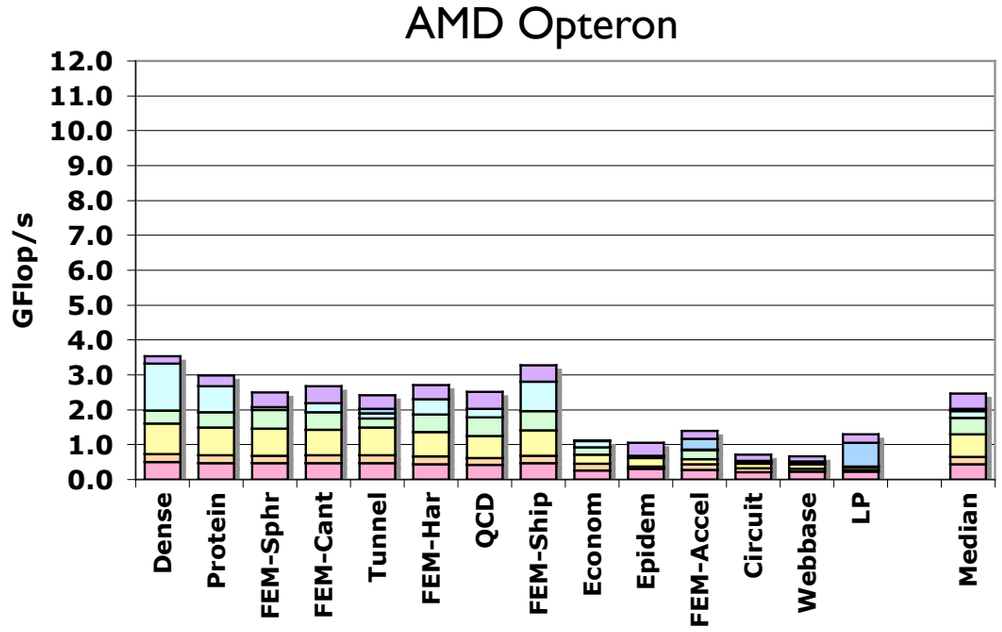
Cell

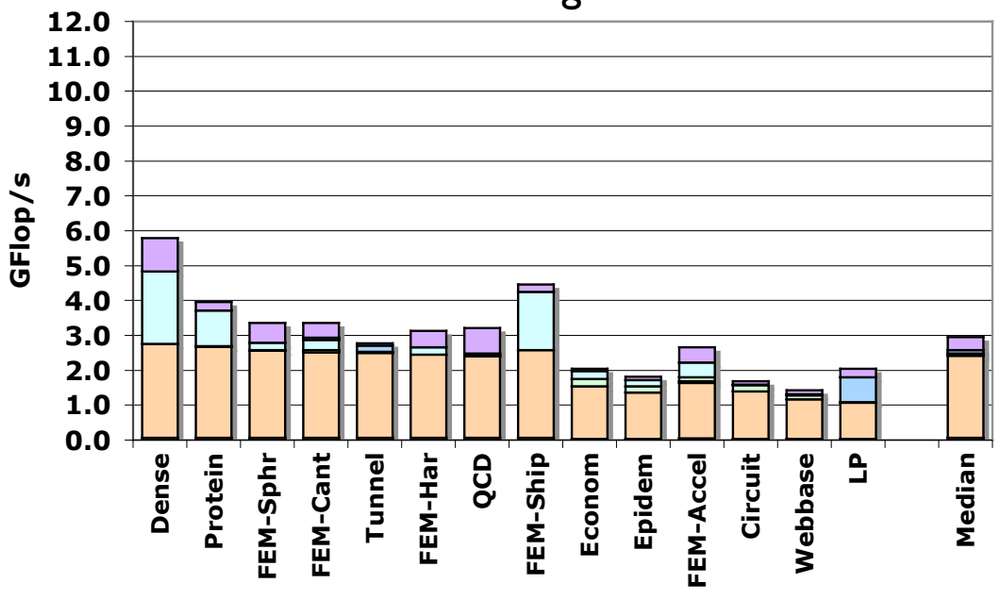


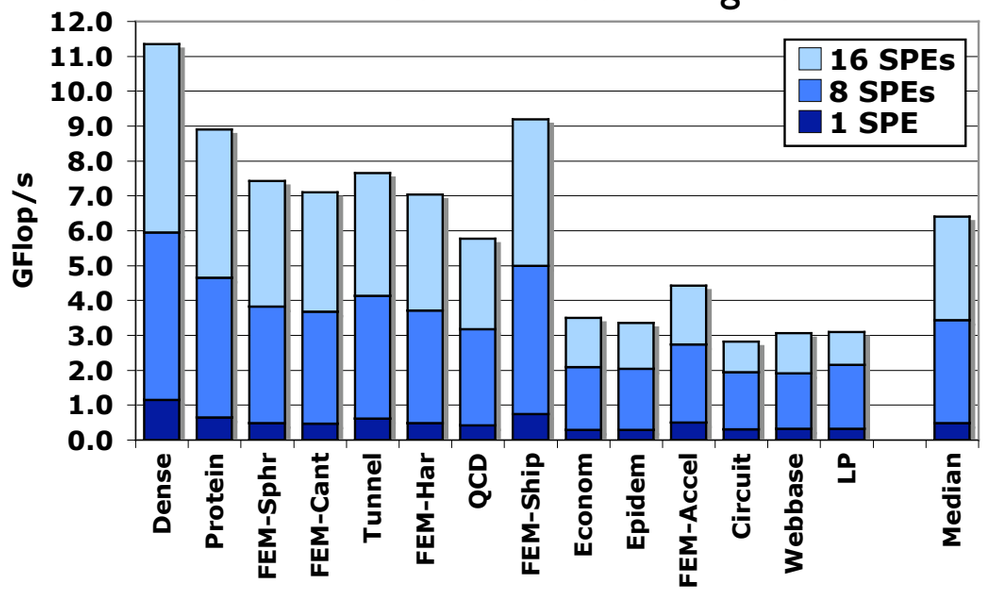
STI Cell Blade (2 x 8)

- Single vs. double: 14x
- Cache-block ⇒ Local-store
- SIMD forces 2x1 blocking

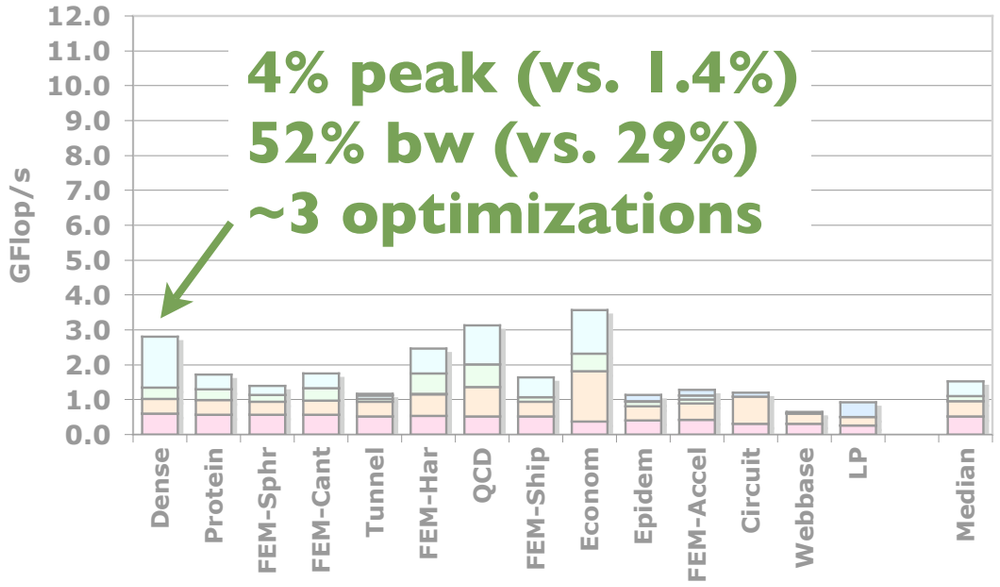

Intel Clovertown
GFlop/s
12.0
11.0
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


AMD Opteron
GFlop/s
12.0
11.0
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

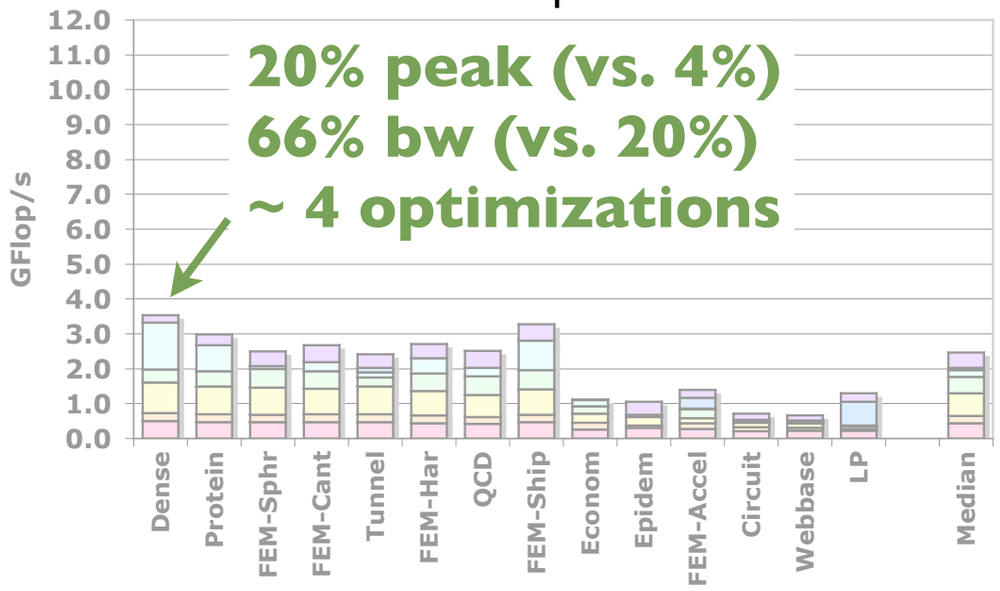

Sun Niagara2
GFlop/s
12.0
11.0
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median


IBM Cell Broadband Engine
GFlop/s
16 SPEs
8 SPEs
1 SPE
12.0
11.0
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
Dense
Protein
FEM-Sphr
FEM-Cant
Tunnel
FEM-Har
QCD
FEM-Ship
Econom
Epidem
FEM-Accel
Circuit
Webbase
LP
Median

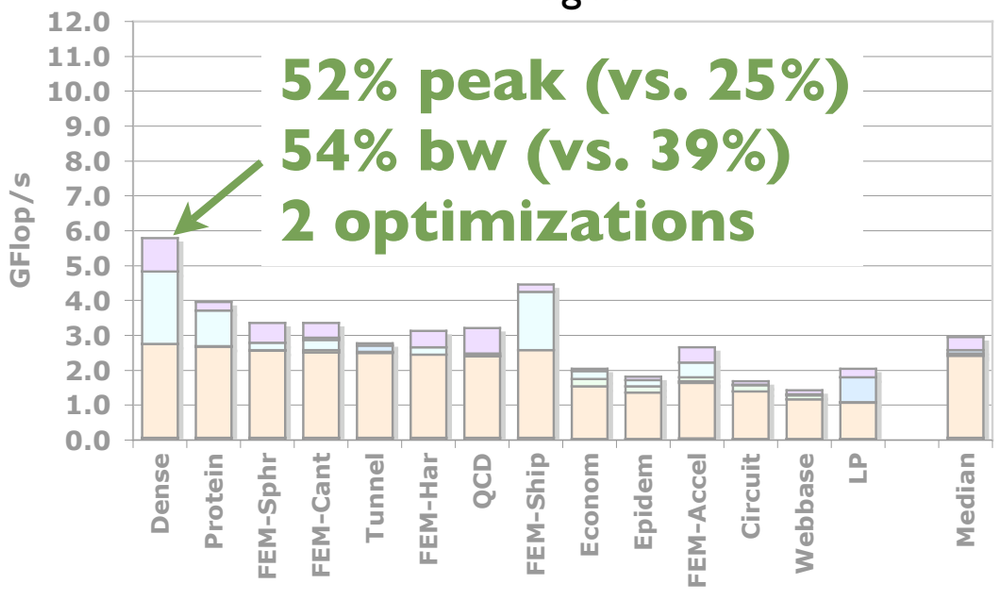
Intel Clovertown



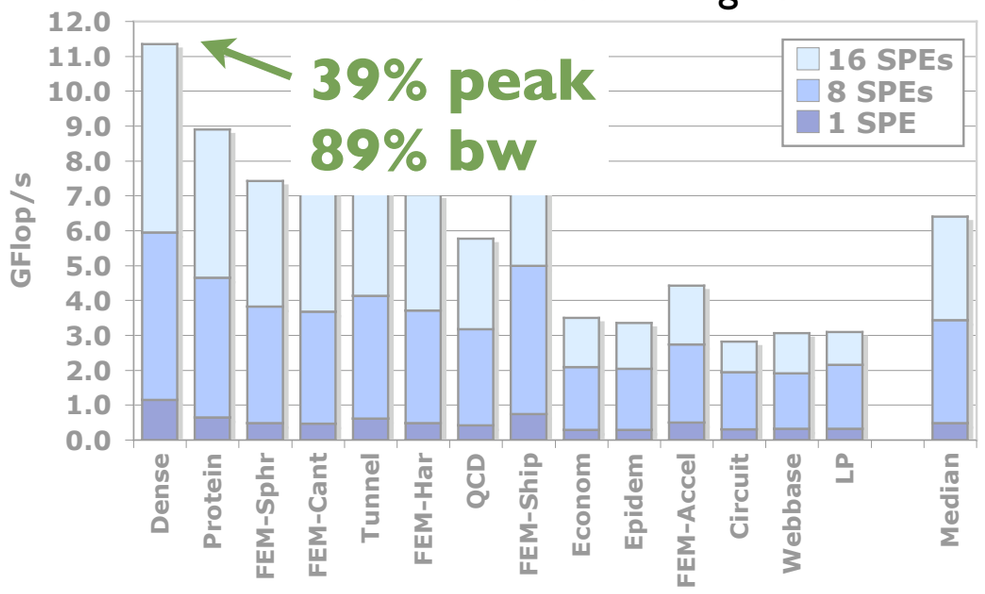
AMD Opteron



Sun Niagara2

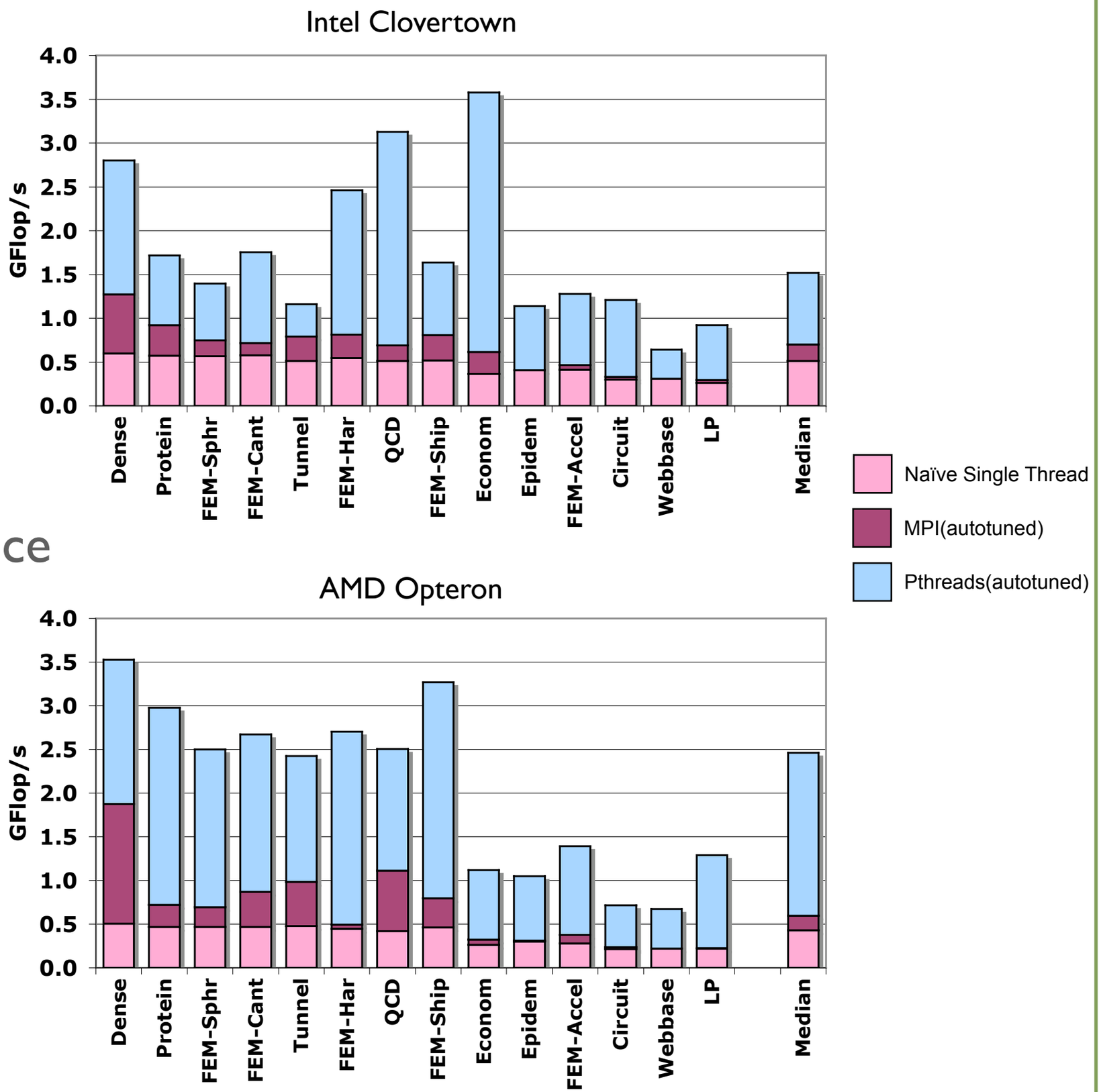


IBM Cell Broadband Engine



- An “off-the-shelf” approach
  - OSKI (tuned serial)
  - + PETSc (parallel MPI)
  - + MPICH (shmem)

Issues:
Copies
Load imbalance



Review

- Key conclusions
  - The most complex architecture needed most tuning yet achieved lowest performance
    - 50–60% DRAM bandwidth vs. 90% on Cell
    - Place, Prefetch, Compress, Block (& Buy, Tune)
- For power/performance issues, see SC'07 paper
- Status and future work
  - Integration into OSKI (heuristics)
  - Improved "off-the-shelf" comparisons (MPI+OMP)