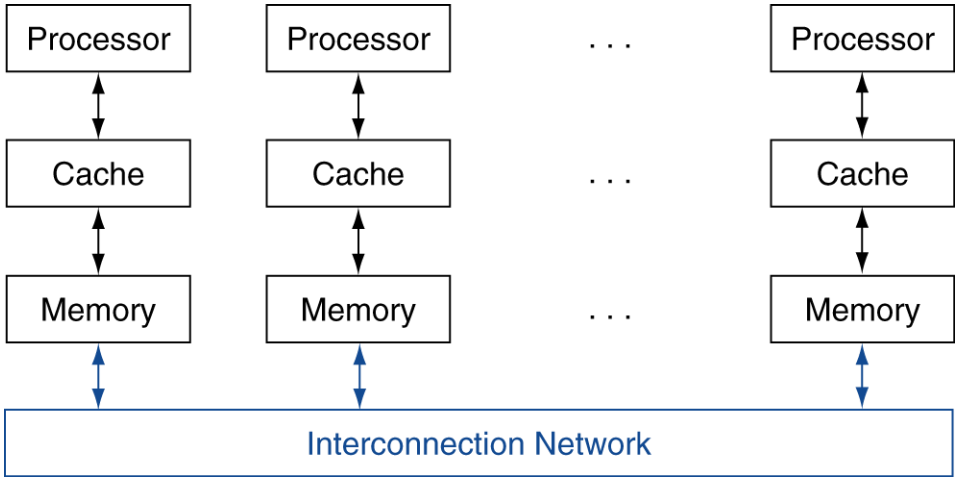
Chapter 6 (continued)

Parallel Processors from Client to Cloud

Message Passing

- Each processor has private physical address space
- Hardware sends/receives messages between processors



Loosely Coupled Clusters

- Network of independent computers
  - Each has private memory and OS
  - Connected using I/O system
    - E.g., Ethernet/switch, Internet
- Suitable for applications with independent tasks
  - Web servers, databases, simulations, …
- High availability, scalable, affordable
- Problems
  - Administration cost (prefer virtual machines)
  - Low interconnect bandwidth
    - c.f. processor/memory bandwidth on an SMP

Sum Reduction (Again)

- Sum 100,000 on 100 processors
- First distribute 100 numbers to each
  - The do partial sums

```
sum = 0;
for (i = 0; i<1000; i = i + 1)
   sum = sum + AN[i];
```

- Reduction
  - Half the processors send, other half receive and add
  - The quarter send, quarter receive and add, …

Sum Reduction (Again)

- Given send() and receive() operations

```
limit = 100; half = 100;/* 100 processors */
repeat
  half = (half+1)/2; /* send vs. receive
                        dividing line */
  if (Pn >= half && Pn < limit)
    send(Pn - half, sum);
  if (Pn < (limit/2))
    sum = sum + receive();
  limit = half; /* upper limit of senders */
until (half == 1); /* exit with final sum */
```

  - Send/receive also provide synchronization
  - Assumes send/receive take similar time to addition

Grid Computing

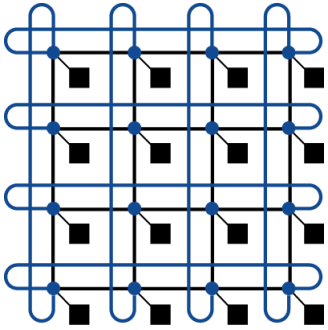
- Separate computers interconnected by long-haul networks
  - E.g., Internet connections
  - Work units farmed out, results sent back
- Can make use of idle time on PCs
  - E.g., SETI@home, World Community Grid

Interconnection Networks

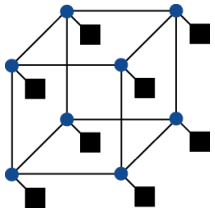
- Network topologies
  - Arrangements of processors, switches, and links

Bus

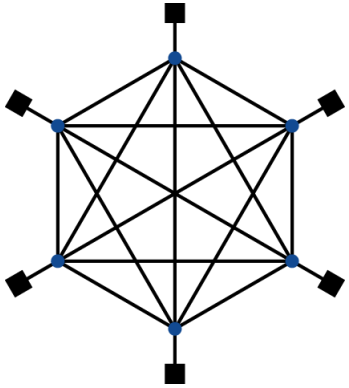
Ring



2D Mesh

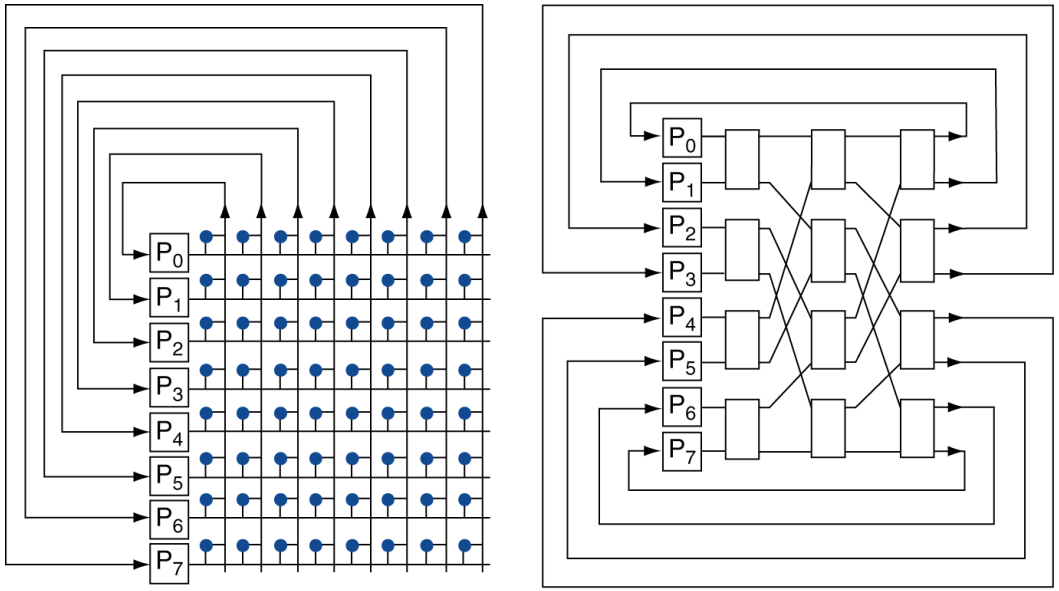


N-cube (N = 3)



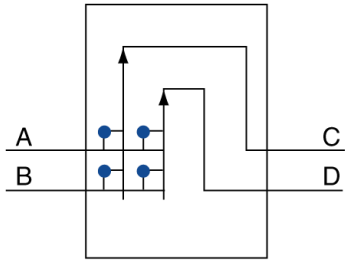
Fully connected

Multistage Networks



a. Crossbar

b. Omega network



c. Omega network switch box

Network Characteristics

- Performance
  - Latency per message (unloaded network)
  - Throughput
    - Link bandwidth
    - Total network bandwidth: link bandwidth multiplied by the number of links
    - Bisection bandwidth:
      - Divide the network into two halves
      - Sum the bandwidth of the links that cross the dividing line
      - If not symmetric, consider the worst case division
  - Congestion delays (depending on traffic)
- Cost
- Power
- Routability in silicon

Parallel Benchmarks

- Linpack: matrix linear algebra
- SPECrate: parallel run of SPEC CPU programs
  - Job-level parallelism
- SPLASH: Stanford Parallel Applications for Shared Memory
  - Mix of kernels and applications, strong scaling
- NAS (NASA Advanced Supercomputing) suite
  - computational fluid dynamics kernels
- PARSEC (Princeton Application Repository for Shared Memory Computers) suite
  - Multithreaded applications using Pthreads and OpenMP

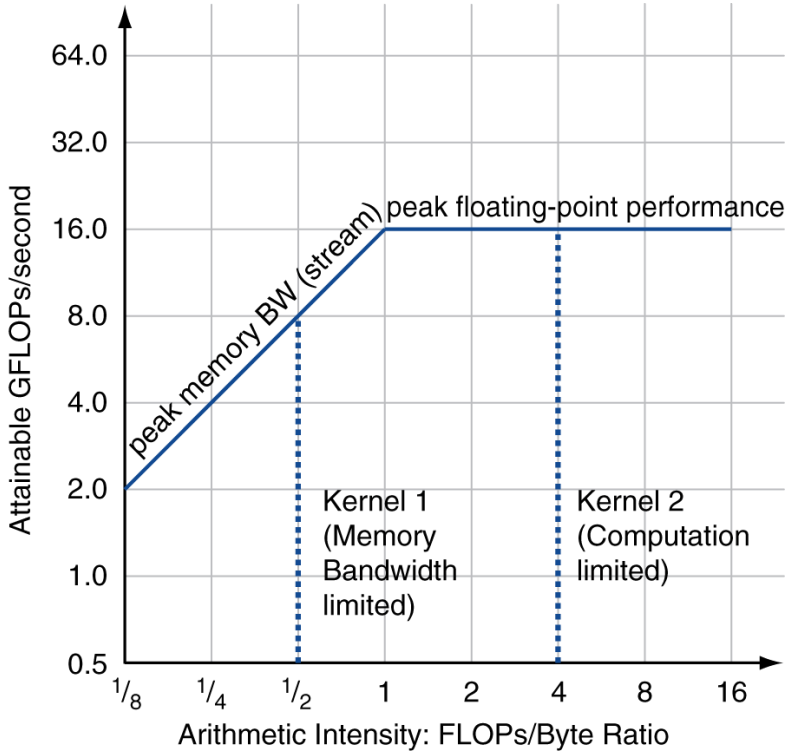
Code or Applications?

- Traditional benchmarks
  - Fixed code and data sets
- Parallel programming is evolving
  - Should algorithms, programming languages, and tools be part of the system?
  - Compare systems, provided they implement a given application
  - E.g., Linpack, Berkeley Design Patterns
- Would foster innovation in approaches to parallelism

Modeling Performance

- Assume performance metric of interest is achievable GFLOPs/sec
  - Measured using computational kernels from Berkeley Design Patterns
- Arithmetic intensity of a kernel
  - FLOPs per byte of memory accessed
- For a given computer, determine
  - Peak GFLOPS (from data sheet)
  - Peak memory bytes/sec (using Stream benchmark)

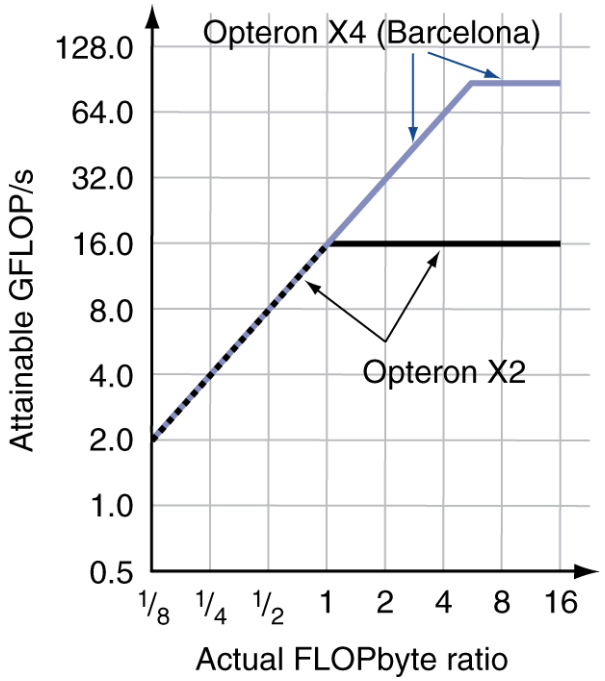
Roofline Diagram



Attainable GPLOPs/sec
= Max (Peak Memory BW × Arithmetic Intensity, Peak FP Performance)

Comparing Systems

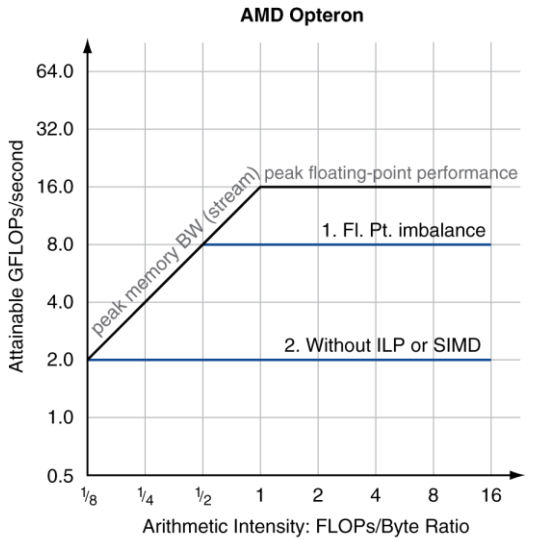
- Example: Opteron X2 vs. Opteron X4
  - 2-core vs. 4-core, 2× FP performance/core, 2.2GHz vs. 2.3GHz
  - Same memory system

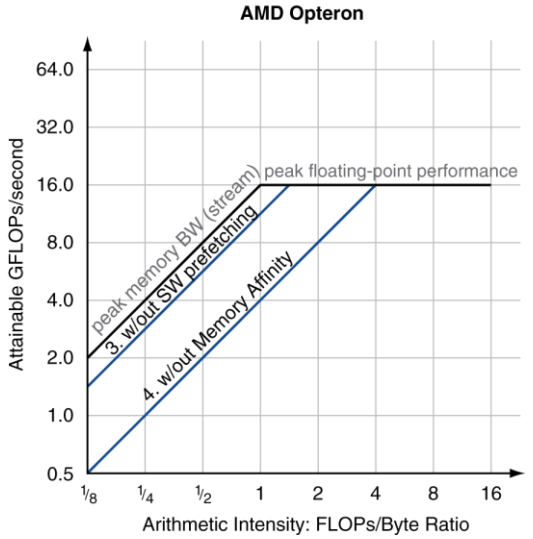


- To get higher performance on X4 than X2
  - Need high arithmetic intensity
  - Or working set must fit in X4's 2MB L-3 cache

Optimizing Performance

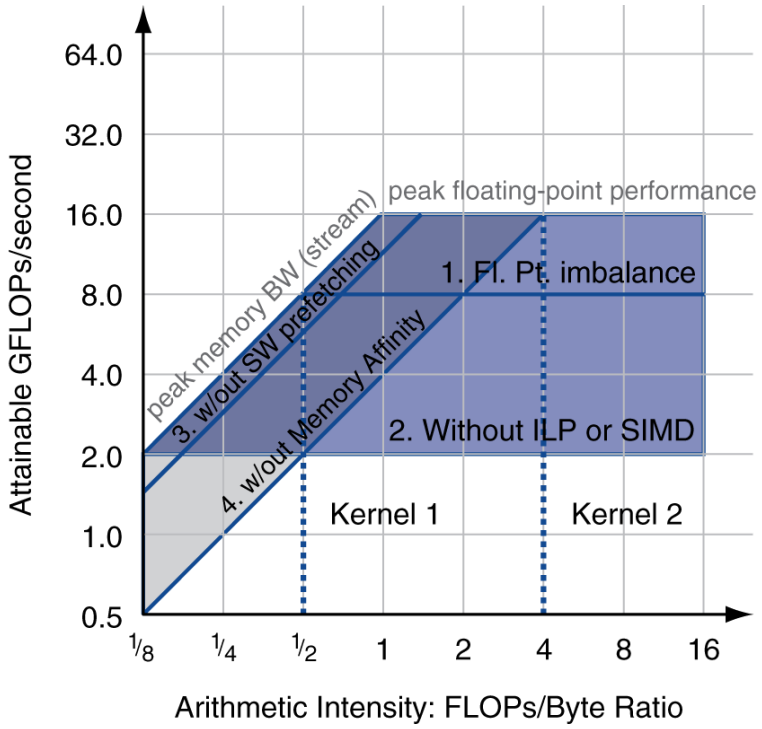
- Optimize FP performance
  - Balance adds & multiplies
  - Improve superscalar ILP and use of SIMD instructions
- Optimize memory usage
  - Software prefetch
    - Avoid load stalls
  - Memory affinity
    - Avoid non-local data accesses





Optimizing Performance

- Choice of optimization depends on arithmetic intensity of code



- Arithmetic intensity is not always fixed
  - May scale with problem size
  - Caching reduces memory accesses
    - Increases arithmetic intensity

Pop Quiz

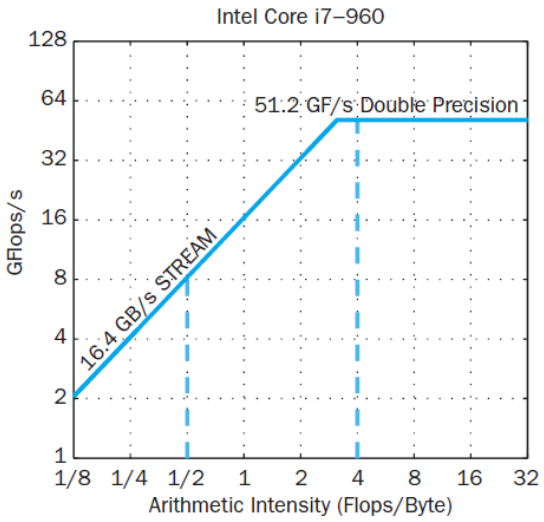
- There is always a "best" way to improve performance, which can be found by only looking at a roofline diagram

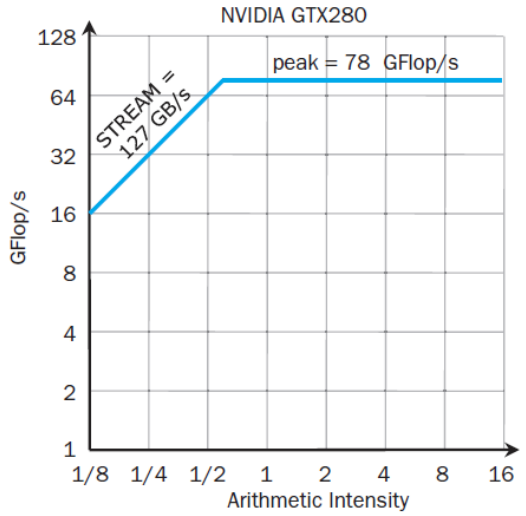
- A: True
- B: False

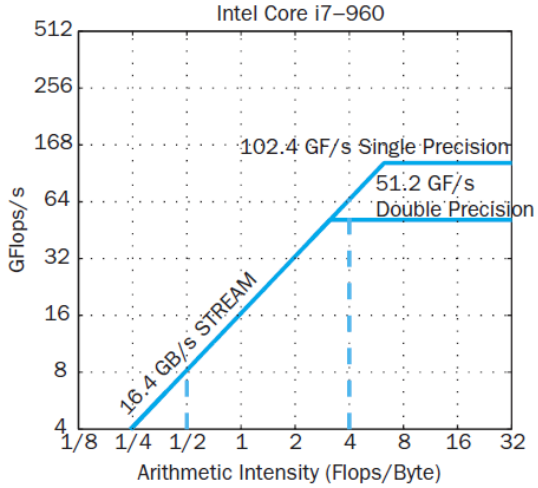
i7-960 vs. NVIDIA Tesla 280/480

	Core i7-960	GTX 280	GTX 480	Ratio 280/i7	Ratio 480/i7
Number of processing elements (cores or SMs)	4	30	15	7.5	3.8
Clock frequency (GHz)	3.2	1.3	1.4	0.41	0.44
Die size	263	576	520	2.2	2.0
Technology	Intel 45 nm	TCMS 65 nm	TCMS 40 nm	1.6	1.0
Power (chip, not module)	130	130	167	1.0	1.3
Transistors	700 M	1400 M	3100 M	2.0	4.4
Memory brandwith (GBytes/sec)	32	141	177	4.4	5.5
Single frecision SIMD width	4	8	32	2.0	8.0
Dobule precision SIMD with	2	1	16	0.5	8.0
Peak Single frecision scalar FLOPS (GFLOP/sec)	26	117	63	4.6	2.5
Peak Single frecision s SIMD FLOPS (GFLOP/Sec)	102	311 to 933	515 to 1344	3.0-9.1	6.6-13.1
(SP 1 add or multiply)	N.A.	(311)	(515)	(3.0)	(6.6)
(SP 1 instruction fused)	N.A	(622)	(1344)	(6.1)	(13.1)
(face SP dual issue fused)	N.A	(933)	N.A	(9.1)	–
Peal double frecision SIMD FLOPS (GFLOP/sec)	51	78	515	1.5	10.1

Rooflines


Intel Core i7–960
128
64
32
16
8
4
2
1
GFlops/s
51.2 GF/s Double Precision
16.4 GB/s STREAM
1/8 1/4 1/2 1 2 4 8 16 32
Arithmetic Intensity (Flops/Byte)


NVIDIA GTX280
128
64
32
16
8
4
2
1
GFlop/s
peak = 78 GFlop/s
STREAM = 127 GB/s
1/8 1/4 1/2 1 2 4 8 16
Arithmetic Intensity


Intel Core i7–960
512
256
168
64
32
16
8
4
GFlops/s
102.4 GF/s Single Precision
51.2 GF/s
Double Precision
16.4 GB/s STREAM
1/8 1/4 1/2 1 2 4 8 16 32
Arithmetic Intensity (Flops/Byte)

Benchmarks

Kernel	Units	Core i7-960	GTX 280	GTX 280/ i7-960
SGEMM	GFLOP/sec	94	364	3.9
MC	Billion paths/sec	0.8	1.4	1.8
Conv	Million pixels/sec	1250	3500	2.8
FFT	GFLOP/sec	71.4	213	3.0
SAXPY	GBytes/sec	16.8	88.8	5.3
LBM	Million lookups/sec	85	426	5.0
Solv	Frames/sec	103	52	0.5
SpMV	GFLOP/sec	4.9	9.1	1.9
GJK	Frames/sec	67	1020	15.2
Sort	Million elements/sec	250	198	0.8
RC	Frames/sec	5	8.1	1.6
Search	Million queries/sec	50	90	1.8
Hist	Million pixels/sec	1517	2583	1.7
Bilat	Million pixels/sec	83	475	5.7

Performance Summary

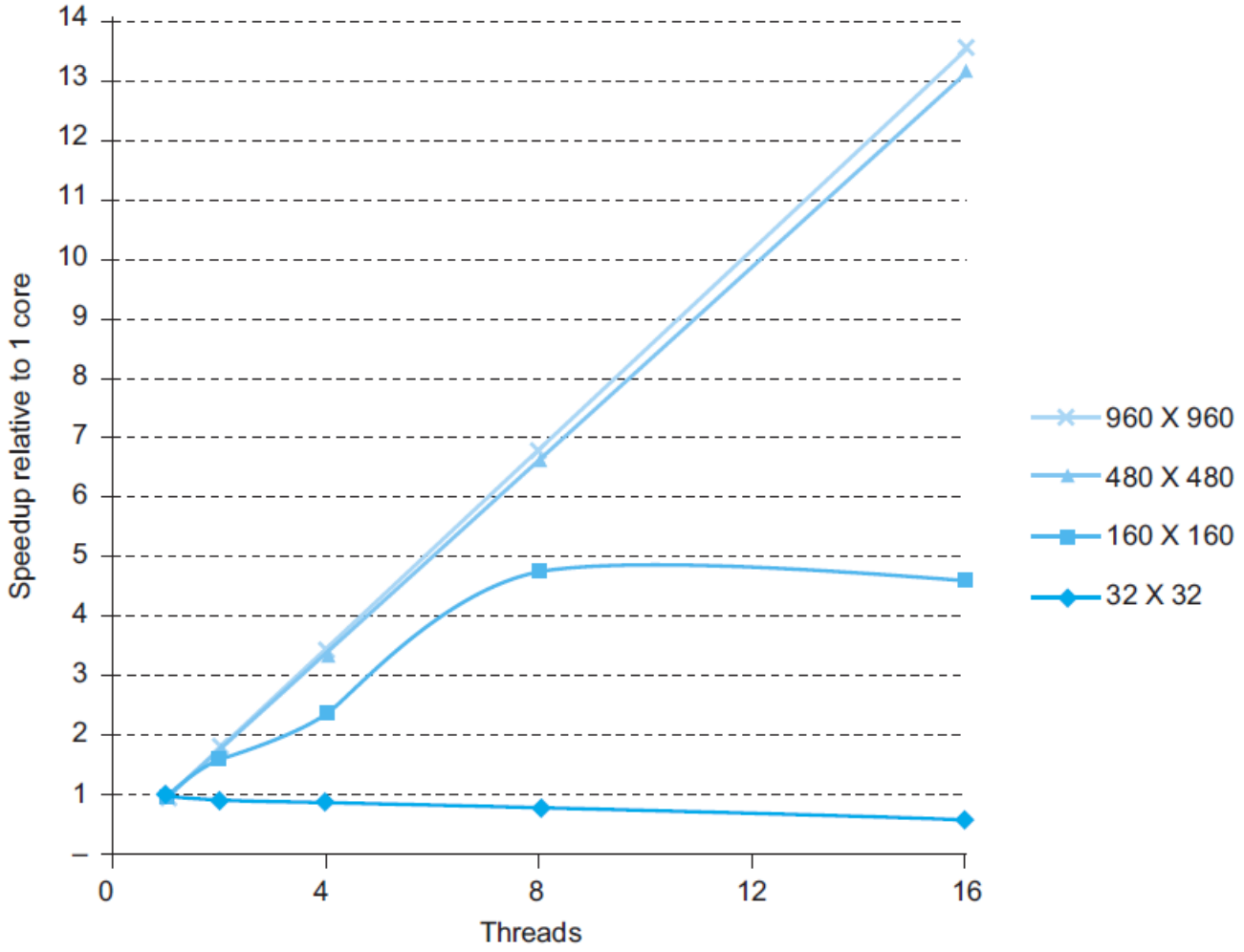
- GPU (480) has 4.4 X the memory bandwidth
  - Benefits memory bound kernels
- GPU has 13.1 X the single precision throughout, 2.5 X the double precision throughput
  - Benefits FP compute bound kernels
- CPU cache prevents some kernels from becoming memory bound when they otherwise would on GPU
- GPUs offer scatter-gather, which assists with kernels with strided data
- Lack of synchronization and memory consistency support on GPU limits performance for some kernels

Multi-threading DGEMM

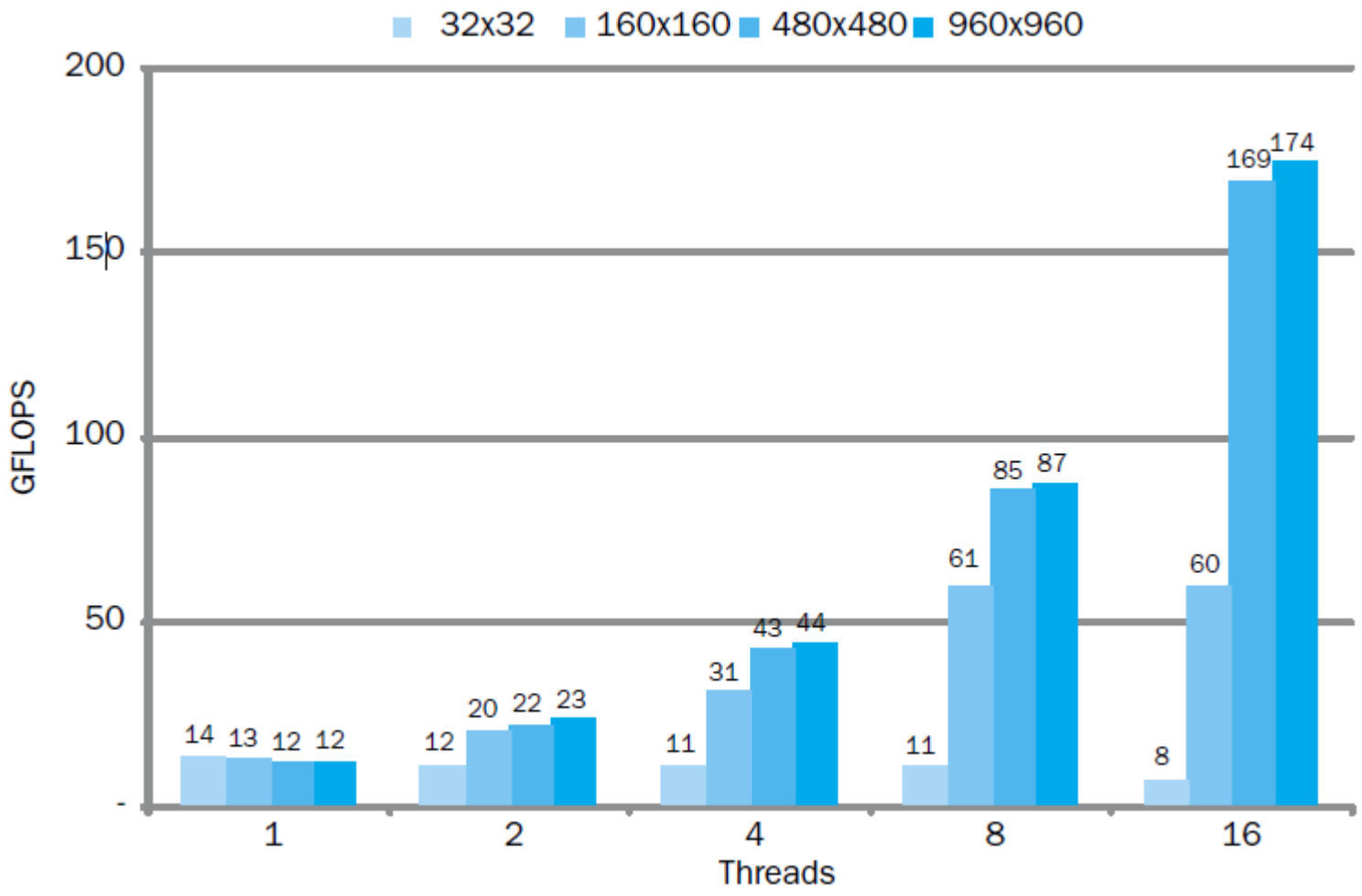
- Use OpenMP:

```
void dgemm (int n, double* A, double* B, double* C)
{
#pragma omp parallel for
 for ( int sj = 0; sj < n; sj += BLOCKSIZE )
  for ( int si = 0; si < n; si += BLOCKSIZE )
   for ( int sk = 0; sk < n; sk += BLOCKSIZE )
    do_block(n, si, sj, sk, A, B, C);
}
```

Multithreaded DGEMM



Multithreaded DGEMM



Fallacies

- Amdahl's Law doesn't apply to parallel computers
  - Since we can achieve linear speedup
  - But only on applications with weak scaling
- Peak performance tracks observed performance
  - Marketers like this approach!
  - But compare Xeon with others in example
  - Need to be aware of bottlenecks

Pitfalls

- Not developing the software to take account of a multiprocessor architecture
  - Example: using a single lock for a shared composite resource
    - Serializes accesses, even if they could be done in parallel
    - Use finer-granularity locking

Concluding Remarks

- Goal: higher performance by using multiple processors
- Difficulties
  - Developing parallel software
  - Devising appropriate architectures
- SaaS importance is growing and clusters are a good match
- Performance per dollar and performance per Joule drive both mobile and WSC

Concluding Remarks (con't)

- SIMD and vector operations match multimedia applications and are easy to program