Perspectives on the "Memory Wall"

John D. McCalpin, Ph.D

IBM Global Microprocessor Development

Austin, TX

The "Memory Wall"

- **In December, 1994, Bill Wulf and Sally McKee published a short paper:**
  Hitting the Memory Wall: Implications of the Obvious
  - **Wulf and McKee claim that diverging exponential increase in performance of processors and memory will "soon" drive performance to being completely memory-dominated.**

- **Example Projected Average Memory Access Times:**
  - **1.53 cycles in 2000**
  - **8.25 cycles in 2005**
  - **98.8 cycles in 2010**

- **It has been almost 10 years – what has really happened?**

Today's talk:

- Wulf and McKee did not distinguish latency & bandwidth
  - This is crucial to understanding "the wall" for real applications
- Wulf and McKee did not consider application size or run-time scaling or cache size scaling
  - Does it make sense to run the same problem 100x faster?
  - Caches are often 30x larger than 10 years ago
- Four parts:
  - 1975 – 1994: Trends leading to the hypothesis of "the wall"
  - 1995 – 2004: What has happened since this paper?
  - 2005 – 2010: What are we going to do near term?
  - 2010 – 20xx: What about the long term?

1975 – 1994: Looking Back

- **Historical Precedent**
  - **Processor speed improvements outpaced memory speed improvements several times already**
    - **1960's mainframes**
    - **1970's-1990's vector supercomputers**
  - **Mainframe solution**
    - **Caches**
  - **Supercomputer solution**
    - **SRAM memory**
    - **Heavy interleaving across many banks**

Memory on Cray Vector Supercomputers

	CPUs	Cpu Cycle Time (ns)	banks	banks per cpu	Latency (cpu clocks)	Busy Time (cpu clocks)	Banks Needed per cpu	Out-standing state per cpu
Cray 1	1	12.5	16	16	12	4	4 (*)	96 B (*)
Cray 2	4	4.1	128	32	59	57	57 (*)	472 B (*)
Cray X/MP	4	8.5	64	16	17	8	24	408 B
Cray Y/MP	8	6.0	256	32	22	8	24	528 B
Cray C90	16	4.2	512	32	30	8	24	720 B
Cray T90	32	2.2	1024	32	51	7 (CM02) 4 (CM03)	21 (CM02) 12 (CM03)	1224 B

(*) These systems only supported one 64-bit memory reference per cpu per cycle – all other systems supported three 64-bit memory references per cycle.

Please Note: I am still working on verifying the specific values in this chart – Many of these are likely to be correct, but they should be seen as illustrative, rather than authoritative

So what happened to the vector machines?

- Many bad things happened, but it all comes down to $$$
  - SRAM too expensive
  - Big Crossbars too expensive
  - Attack of the Killer Micros took away revenue associated with “cache-friendly” applications.
- So let’s look at “The Attack of the Killer Micros”

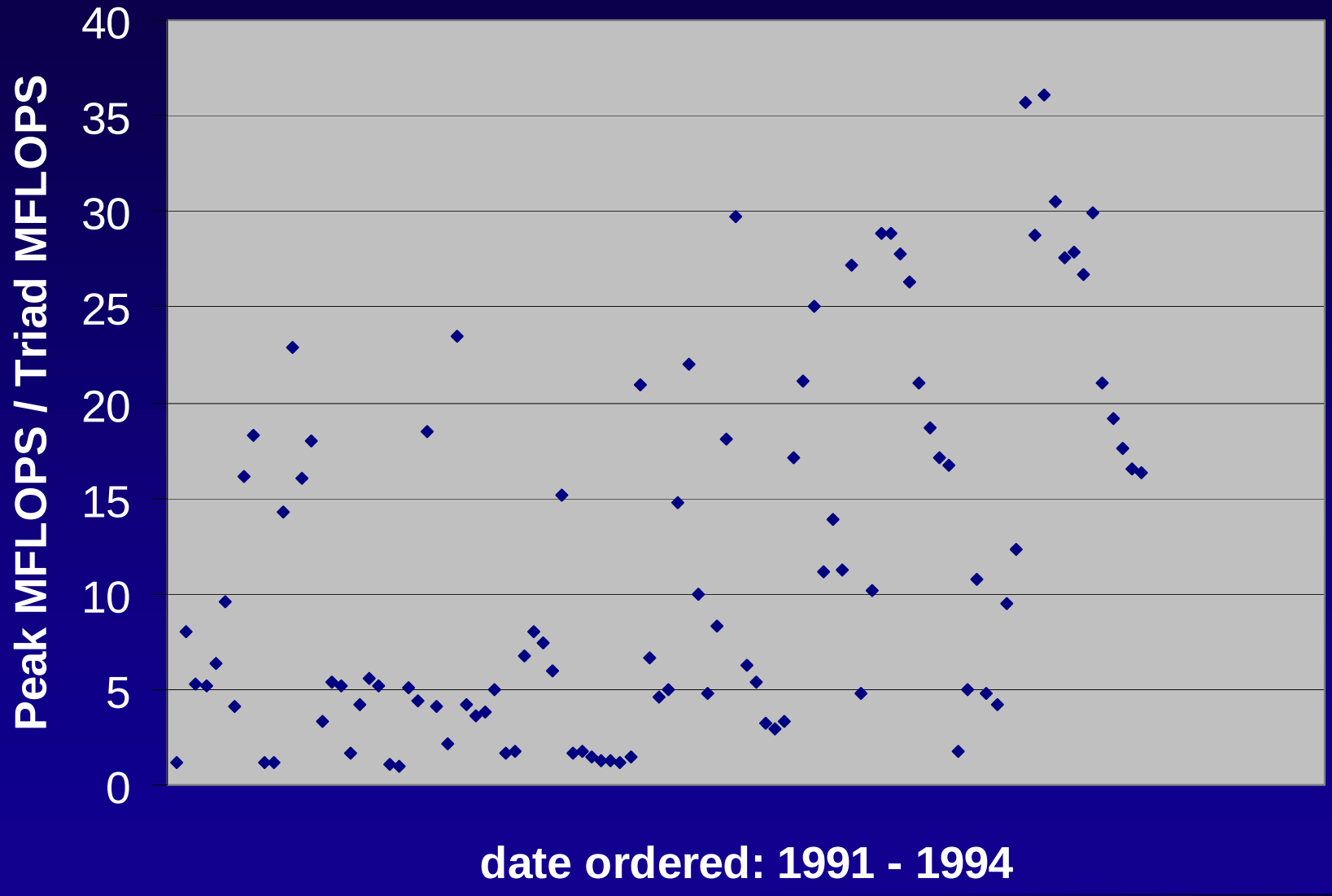
The Attack of the Killer Micros

- In 1990, HP launched the PA-RISC processors and IBM launched the POWER processors
  - Inexpensive
  - Fast floating-point arithmetic
  - Used small caches
  - Bandwidth per FLOP was not too bad, but….
- Rapid increases in CPU frequency were not matched by improvements in memory performance
  - Most caches were “miss and wait”
  - Memory system throughput was typically 1 cache line per latency
  - Latencies were dominated by ~80 ns DRAM access times
  - This access time has changed VERY slowly over last 20 years

Responses: Early 1990's

- Sustained Memory bandwidth improved by
  - Non-blocking caches
    - Stall on use, rather than stall on reference
    - Hit under miss
  - Multiple outstanding cache misses
    - IBM POWER2 supported 2
    - Later processors typically supported 4 misses in mid 1990's

Machine Balance: 1991-1994

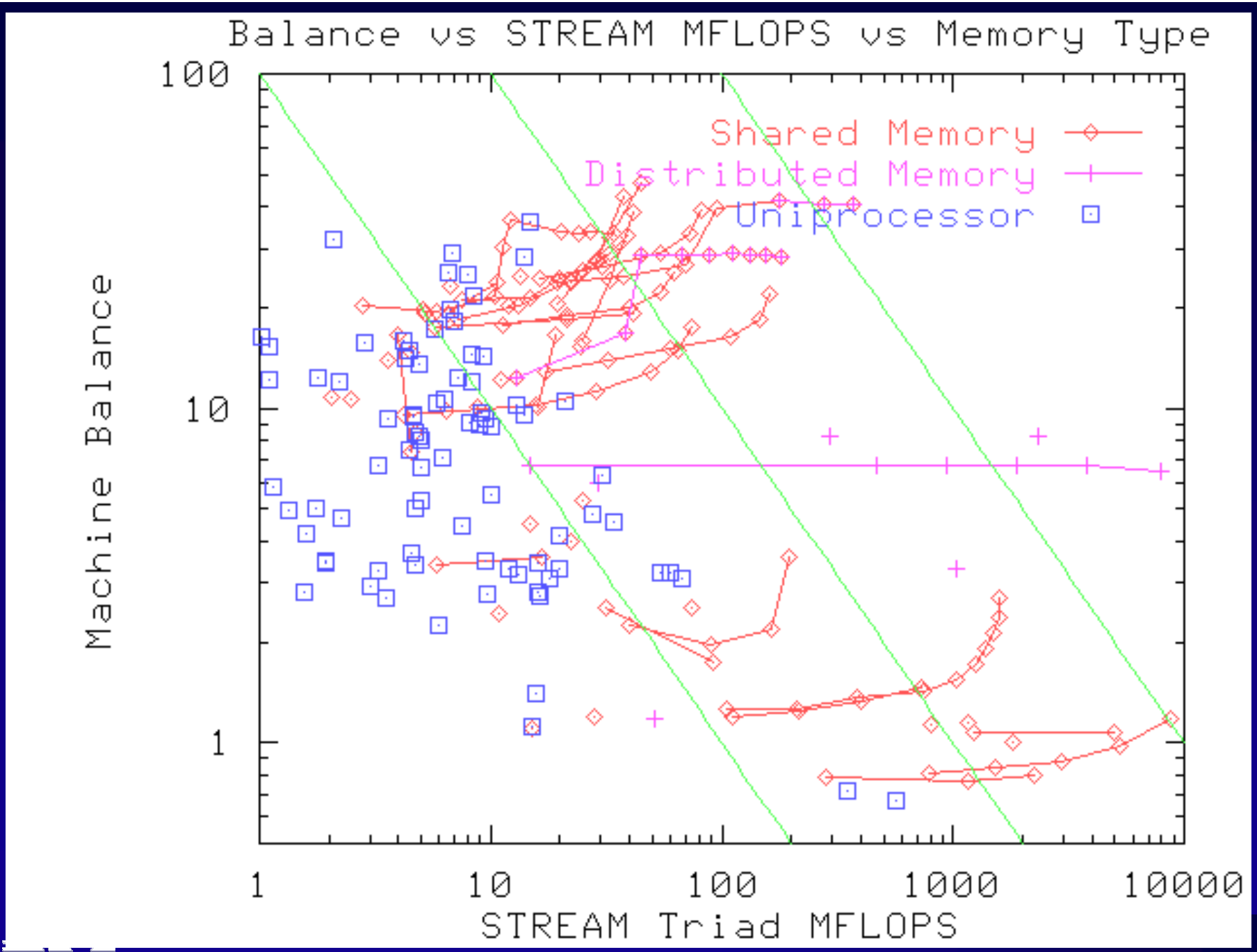


IBM

POWER

New Obstacles: mid 1990's

- RISC Processors moved into SMPs
  - Shared busses have serious electrical constraints on frequency
    - Started out ok, then rapidly fell behind increasing cpu speed
    - Problem still plagues IA32 processors in 2004:
      - Uniprocessor front-side bus = 800 MHz
      - Dual-processor front-side bus = 533 MHz → 1/3 BW/cpu
      - Quad-processor front-side bus = 400 MHz → 1/8 BW/cpu !!!
  - Putting data on point-to-point crossbars does not help
    - Address bus used for cache coherency becomes the bottleneck
      - Sun E10000
  - Latency increased significantly
    - IBM POWER2 uniprocessor: <150 ns latency
    - SGI Power Challenge SMP: >1000 ns latency


Balance vs STREAM MFLOPS vs Memory Type
Shared Memory
Distributed Memory
Uniprocessor
Machine Balance
100
10
1
1
10
100
1000
10000
STREAM Triad MFLOPS

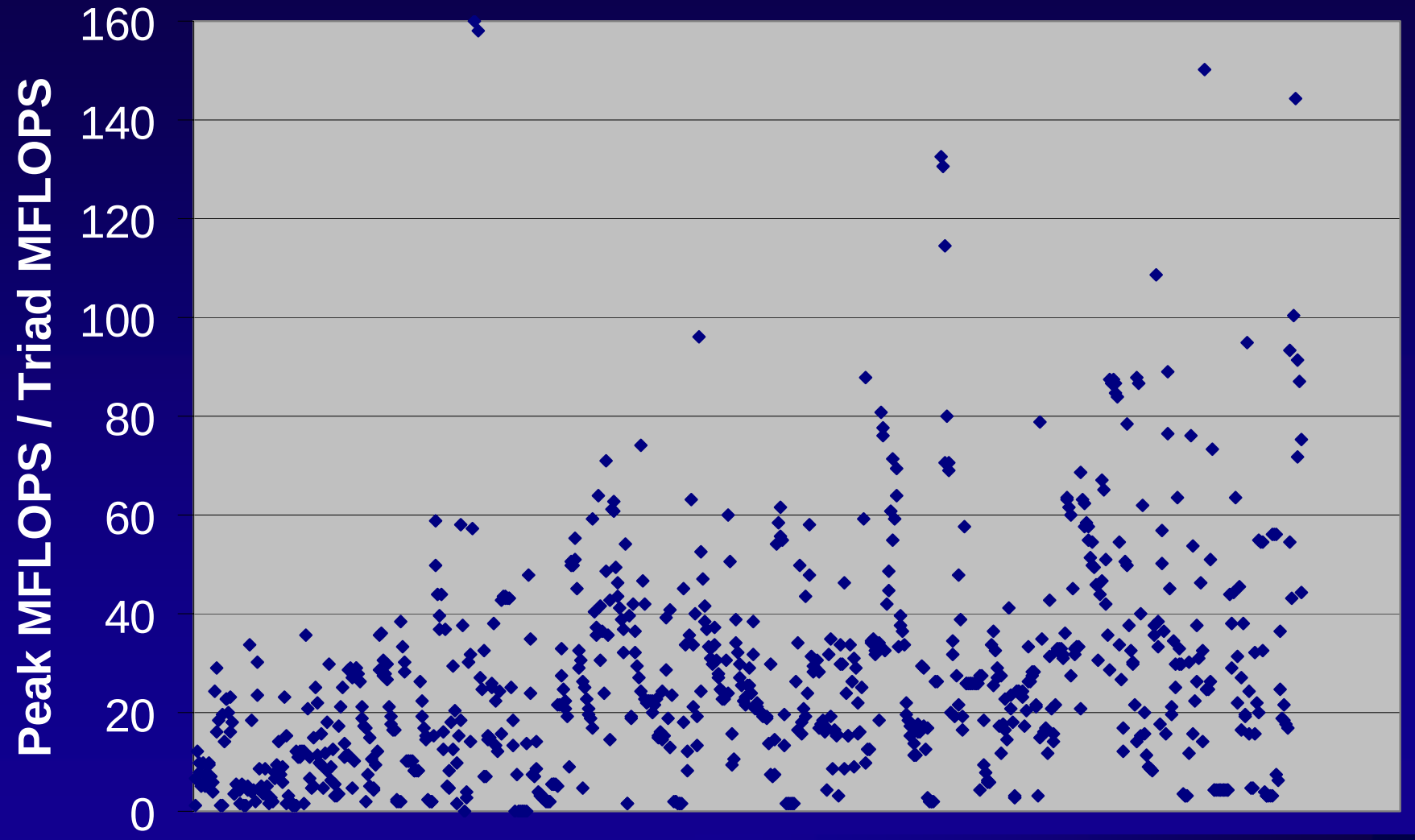
Responses: mid 1990's to present

- Focus on uniprocessors or small SMPs or MPPs for high-bandwidth workloads
  - IBM POWER2, P2SC
  - DEC DS20
  - Cray T3E
- For large SMPs: Distributed-directory-based, cache-coherent, distributed shared memory with non-uniform memory access latency
  - E.g., SGI Origin2000, Origin3000, Altix3000
  - HP Superdome
  - HP GS320, GS1280

New Responses, continued

- On-board memory controllers
  - Reduces latency for some transactions
  - Requires SMP interconnect fabric
    - Better not be a bus!
  - Does not solve cache coherency issues
    - Still need directory-based NUMA for large systems
    - Broadcast over point-to-point links works for small systems
      - E.g., Opteron 2p, 4p, and maybe 8p
- Off-board memory “concentrators”
  - Multiplex multiple DIMM channels onto a narrower, higher-speed channel to minimize use of expensive pins on CPU package

Machine Balances from the STREAM Database: 1991 - 2004



IBM

POWER

How have we done?

- Overall balances have gotten slightly worse,
  - 1991: IBM RS/6000-540, 60 MFLOPS, 144 MB/s, balance = 5
  - 2003: IBM p655+, 6800 MFLOPS/cpu, 5000 MB/s/cpu, balance = 16
- But caches are much larger
  - 1991: IBM RS/6000-540: 64 kB L1 Data
  - 2003: IBM p655+: 32 kB L1 Data, 1536 kB L2, 32768 kB L3
- It only takes a modest re-use rate in these caches to tolerate the 3x reduction in bandwidth per peak FLOP
  - Caches actually work better for many HPC applications than anyone anticipated
- A 3x bandwidth reduction in 12 years is not a very scary exponential trend….
  - Wulf & McKee speculated numbers like 64x in 10 years

New Bandwidth Challenges: 2004+

- DRAM bank cycle time still large
  - T_rc = 70 ns on typical DDR devices
  - 32 Byte transfer time is
    - DDR200 20 ns
    - DDR266
    - DDR333
    - DDR400 10 ns
    - DDR2/533
    - DDR2/667
    - DDR2/800 5 ns
  - Requires 14 banks with DDR2/800 to fill pipeline even with perfect address stream behaviour
    - Assumes closed page mode

New BW challenges: 2004+, continued

- DRAM electrical interfaces are becoming very difficult
  - DDR200: relatively easy
  - DDR400: multiple DIMMs per channel is very hard
  - DDR2/800:
    - 1 DIMM/channel runs at 800 MHz
    - 2 DIMMs/channel runs at 400 MHz
- Power
  - Driving these bandwidth levels is getting to be a large fraction of the chip power budget
  - Activating all these DRAMs is a large fraction of the rack power/cooling budget

The Bad News

- This is not the whole story
  - Latency in 1991 was about 14 cpu clocks
  - Latency in 2003 was about 400 cpu clocks
  - We need 10x more concurrent memory references in order to maintain 1/3 of the bandwidth/Peak FLOP
  - Our ISAs are not designed for this
  - Our implementations are not designed for this
    - Multiple outstanding cache misses are more of an afterthought than a core part of the design
    - The Tera MTA is a clear exception

How Bad is this Bad News?

- I don't think enough information is available to be sure
  - i.e., lots of opportunity for academic research
- We need to understand the temporal locality, predictability and prefetchability of address streams in real applications much better than we do now

Extrapolating again….

- It is easy to imagine a new paradigm, in which
  - Computation is free
  - Memory accesses with spatial locality are free (prefetch)
  - Memory accesses with temporal locality are free (cache reuse)
  - All the cost is in the memory fetches that could not be prefetched – how many of these are there in “real” apps?
- But we are not there yet – how far down that path will we go?

How far will we go?

- Extrapolation requires trends to continue
  - Processors keep getting faster at rapid rate
  - Memory bandwidth keeping up with processor speed
  - Memory latency staying almost constant

Valid Assumptions?

- Processor's getting faster?
  - Moore's Law is facing its biggest challenge yet
    - Leakage current >> active current as we shrink devices
    - CMP might keep chip performance on current trajectory?
    - The rebirth of architecture?
- Memory Bandwidth keeping up?
  - Point-to-point electrical signalling will keep up
  - But massive quantities of DRAM are required to provide adequate parallelism – maybe too much $$$?
- Memory latency should stay almost constant for commodity DRAM
  - But $(GB/s) might favor non-standard memory technology

Short Term vs Long Term Reactions

- Machines for delivery in 2008 are already being designed
  - Mostly a continuation of current trends
    - SMT for latency tolerance will become ubiquitous
  - Innovative technology applied to backward-compatible ISAs
- We need better latency tolerance to convert latency-limited code to BW-limited code
- Latency tolerance ideas
  - SMT
  - Assist threads
  - Address stream autocorrelation
  - Continuous program optimization
  - Region-based prefetching
- All of these are hacks – are they good enough hacks?

Change?

- How long will it take to change?
- Look at how long it took to switch the majority of Industrial ISV codes to Message-Passing parallelism
  - Clusters of inexpensive systems initially became available in ~1990
  - Maybe 5% of important ISV codes were available in MPI in 1997
  - Maybe 75% of important ISV codes available in MPI today
- I don't see an obviously disruptive paradigm to port to – once one appears, add 10-15 years for mainstream adoption.
  - A social and economic issue more than a technical one

Disruptive technologies

- Read “The Innovator’s Dilemna”
- Lots of people are looking at game technology to replace PCs
- My money is on embedded technology for replacing PC’s in the mass market:
  - My PDA vs the machine I used for my computational M.S. thesis:
    - Much higher clock frequency
    - Much better internal cpu parallelism
    - 256x as much DRAM
    - DRAM is 3x larger than my original hard disk!
    - Operates for hours without wall power, vs <1 second
- “Good Enough” performance + unique value = ???

Will HPC follow the mass market?

- Since ~1990, HPC workloads have shifted to systems developed for increasingly large and broad customer bases
  - Mainframes
  - Minis
  - Workstations
  - RISC SMPs
  - PC's
  - PC-based servers
- There is no guarantee that the next evolution in the mass market will be useful for HPC, and HPC is not big enough to drive the mass market
  - Game technology could probably be morphed to support HPC
  - Embedded technology might ignore HPC requirements

If we do not follow, then how to lead?

- A fundamental issue remains the enablement of parallelism
  - Disruptive technology transitions often bring drastic reductions in price and price/performance, but typically bring no performance advantage
  - Now that the cost of a “tier 1” single-processor compute platform is well under $2k, further reductions in price will not enable a broader audience of scientists and engineers to have access to “tier 1” uniprocessor systems
- Hardware must support dramatically superior programming models to enable effective use of parallel systems
  - Parallel semantics must be intrinsic
  - Latency tolerance must be intrinsic
  - Load balancing must be intrinsic
  - Failure-tolerant client/server semantics must be intrinsic

Summary

- We have not hit the memory wall, but we are pushing up against it pretty hard
- No disasters are imminent (~4 years), in part because of the slowing of the rate of increase of processor performance
- If we want performance rates to continue to increase, whether processor performance resumes its prior growth, or stalls near current levels, we need a paradigm shift in our programming models to enable the application of significant parallelism to real applications without heroic programming efforts