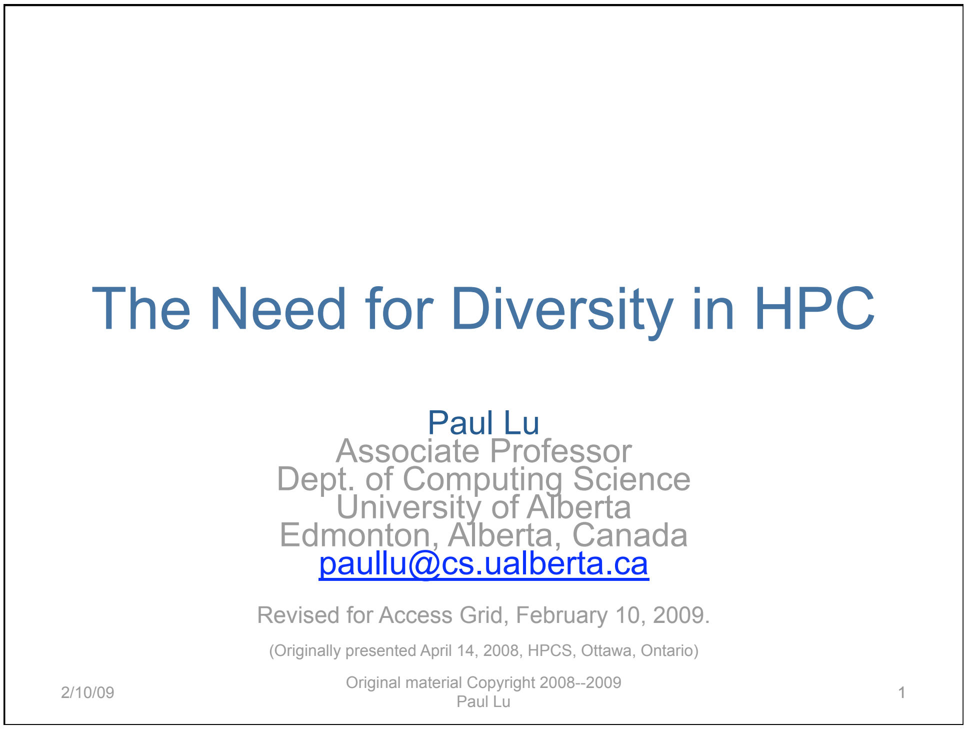
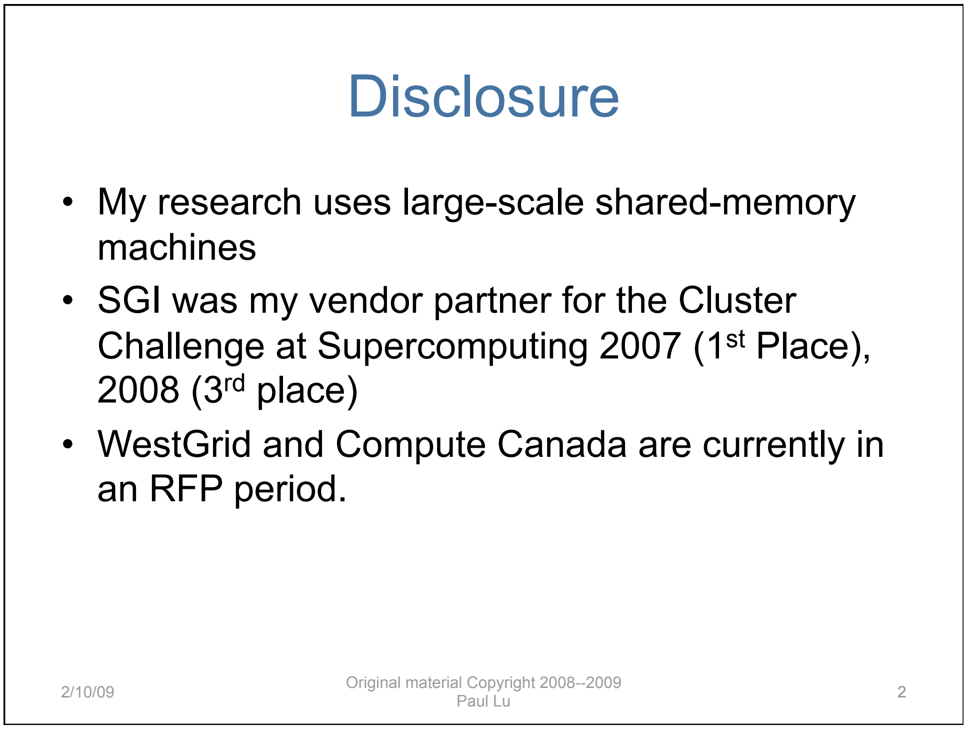


The Need for Diversity in HPC

Paul Lu
Associate Professor
Dept. of Computing Science
University of Alberta
Edmonton, Alberta, Canada
paullu@cs.ualberta.ca

Revised for Access Grid, February 10, 2009.

(Originally presented April 14, 2008, HPCS, Ottawa, Ontario)



Disclosure

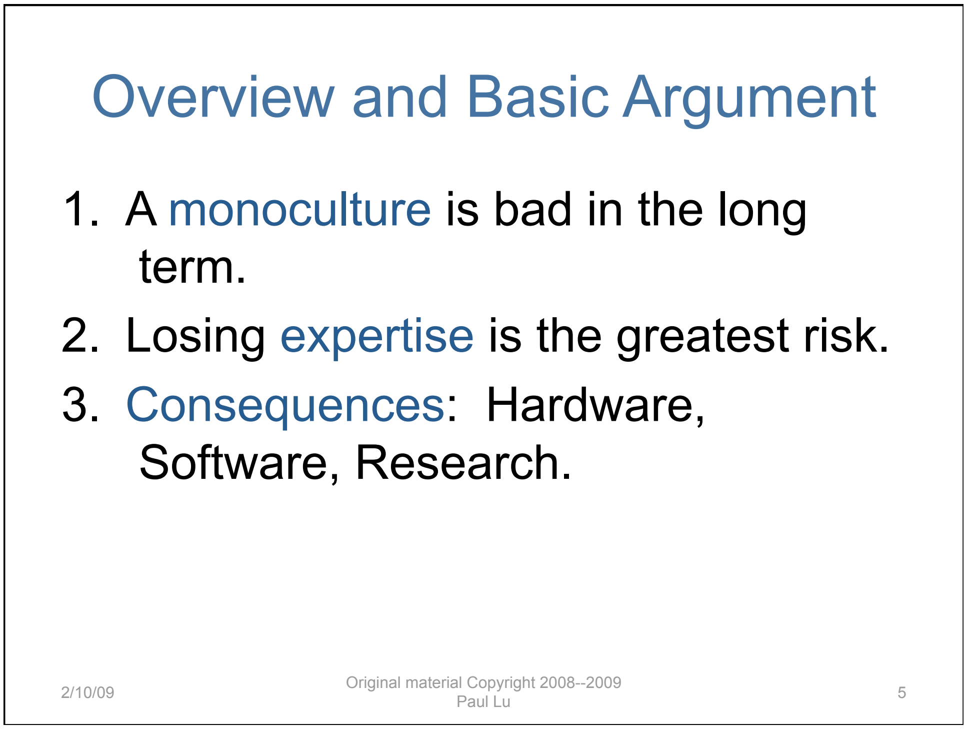
- My research uses large-scale shared-memory machines
- SGI was my vendor partner for the Cluster Challenge at Supercomputing 2007 (1st Place), 2008 (3rd place)
- WestGrid and Compute Canada are currently in an RFP period.

Acknowledgements

- Jonathan Schaeffer, Duane Szafron, Nelson Amaral, Cam Macdonell, Maria Cutumisu, Pawel Gburzynski, Brent Gorda, Greg Wilson, my CMPUT 399 class
- Graphs from Top500.org
- Slides from David Patterson
- Information from Ken Koch
  - LANL Roadrunner Summary 3/12/2007

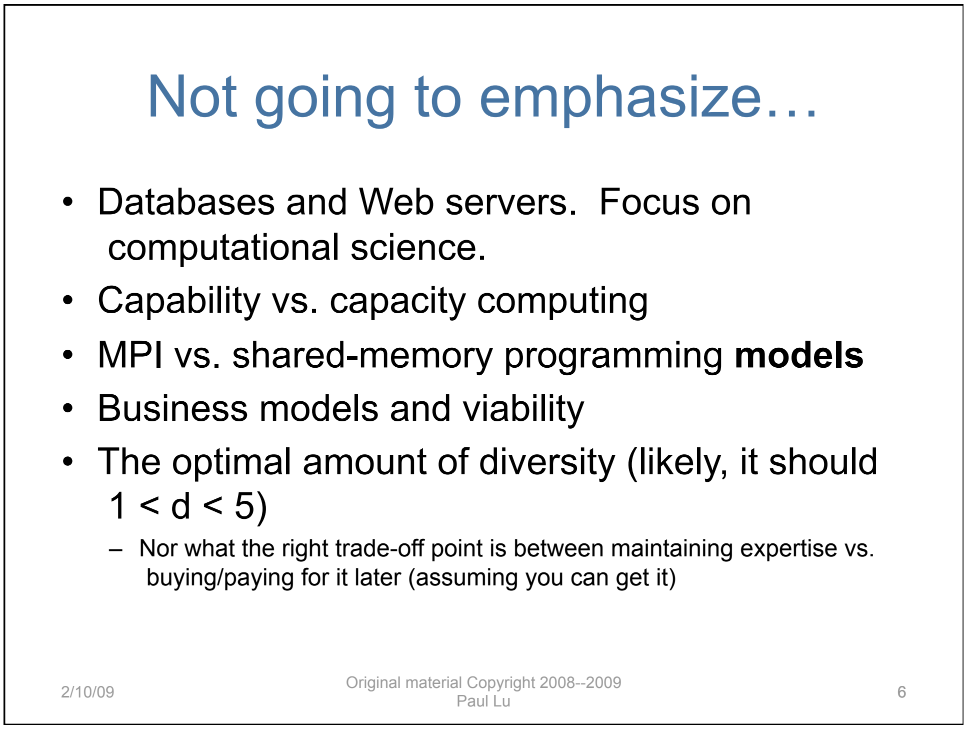
The Audience

- High-performance computing and simulation *users*
- Not necessarily hardware or OS researchers



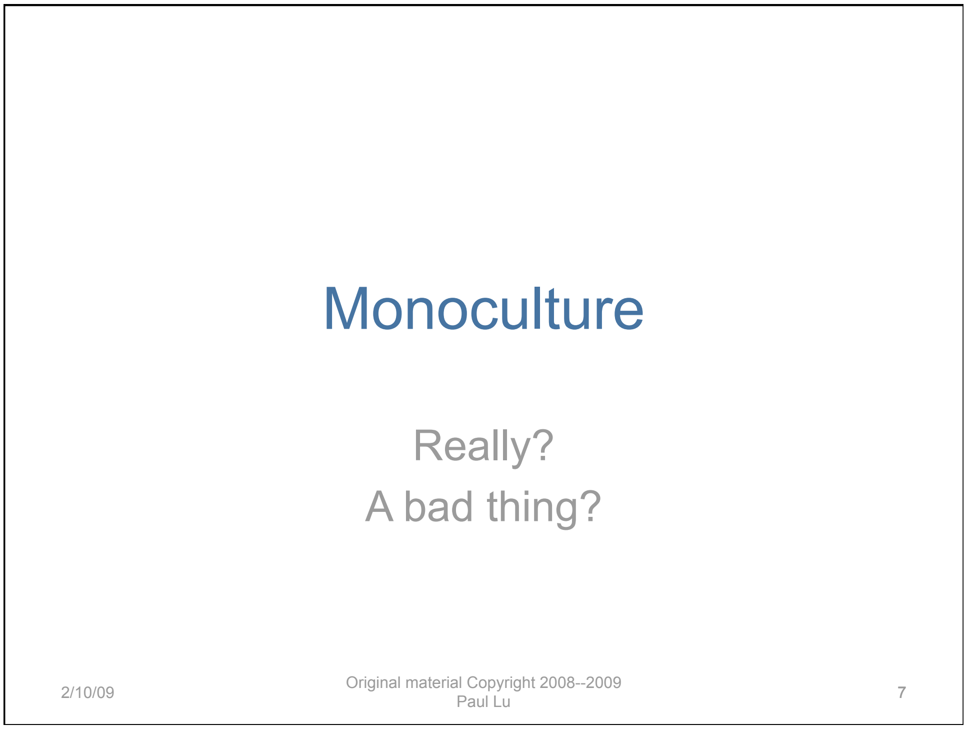
Overview and Basic Argument

1. A monoculture is bad in the long term.
2. Losing expertise is the greatest risk.
3. Consequences: Hardware, Software, Research.



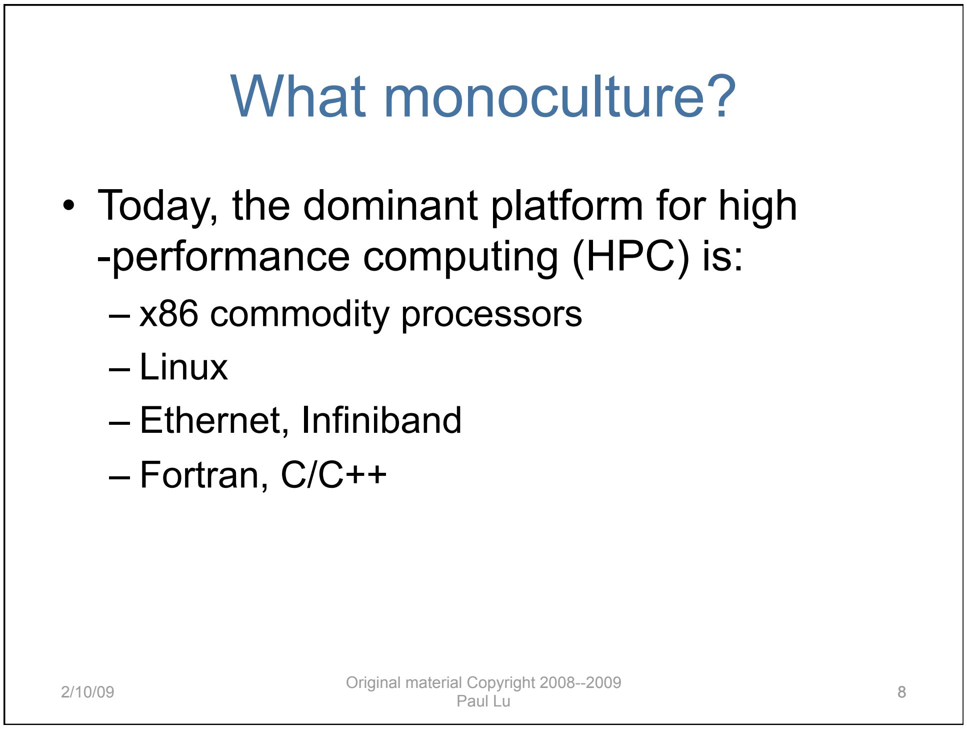
Not going to emphasize…

- Databases and Web servers. Focus on computational science.
- Capability vs. capacity computing
- MPI vs. shared-memory programming **models**
- Business models and viability
- The optimal amount of diversity (likely, it should 1 < d < 5)
  - Nor what the right trade-off point is between maintaining expertise vs. buying/paying for it later (assuming you can get it)



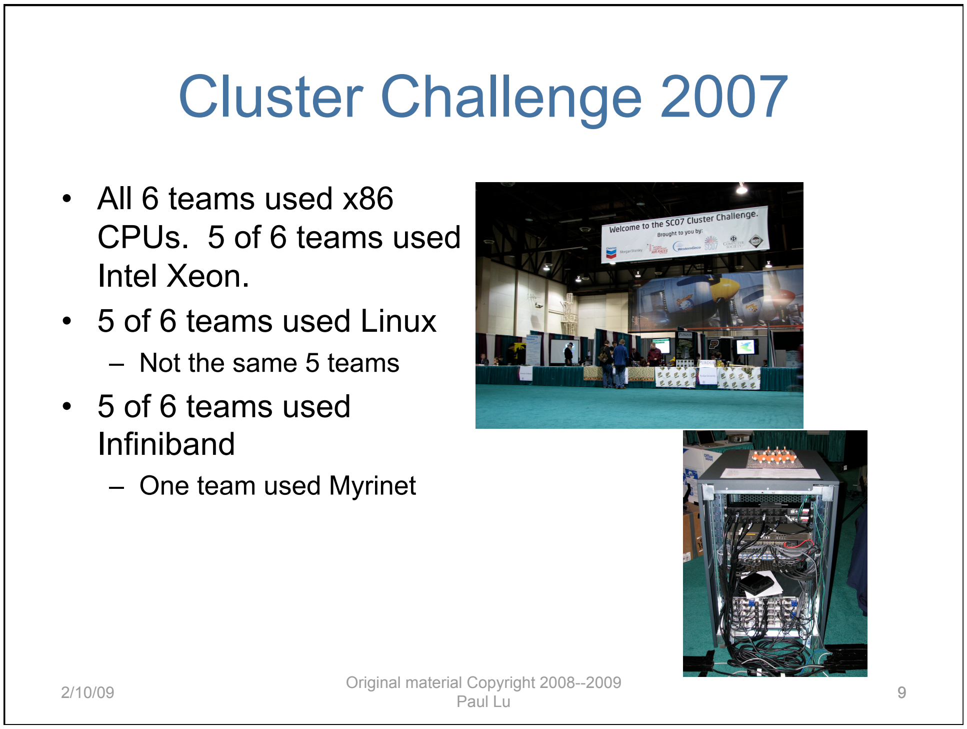
Monoculture

Really?
A bad thing?



What monoculture?

- Today, the dominant platform for high -performance computing (HPC) is:
  - x86 commodity processors
  - Linux
  - Ethernet, Infiniband
  - Fortran, C/C++

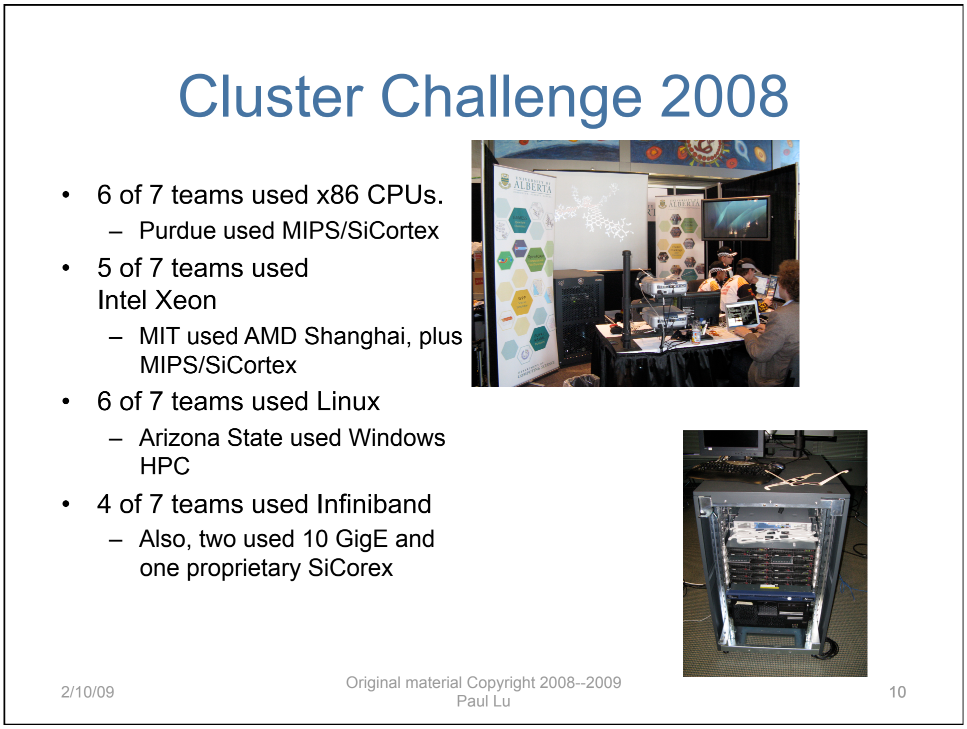


Cluster Challenge 2007

- All 6 teams used x86 CPUs. 5 of 6 teams used Intel Xeon.
- 5 of 6 teams used Linux
  - Not the same 5 teams
- 5 of 6 teams used Infiniband
  - One team used Myrinet

2/10/09

9

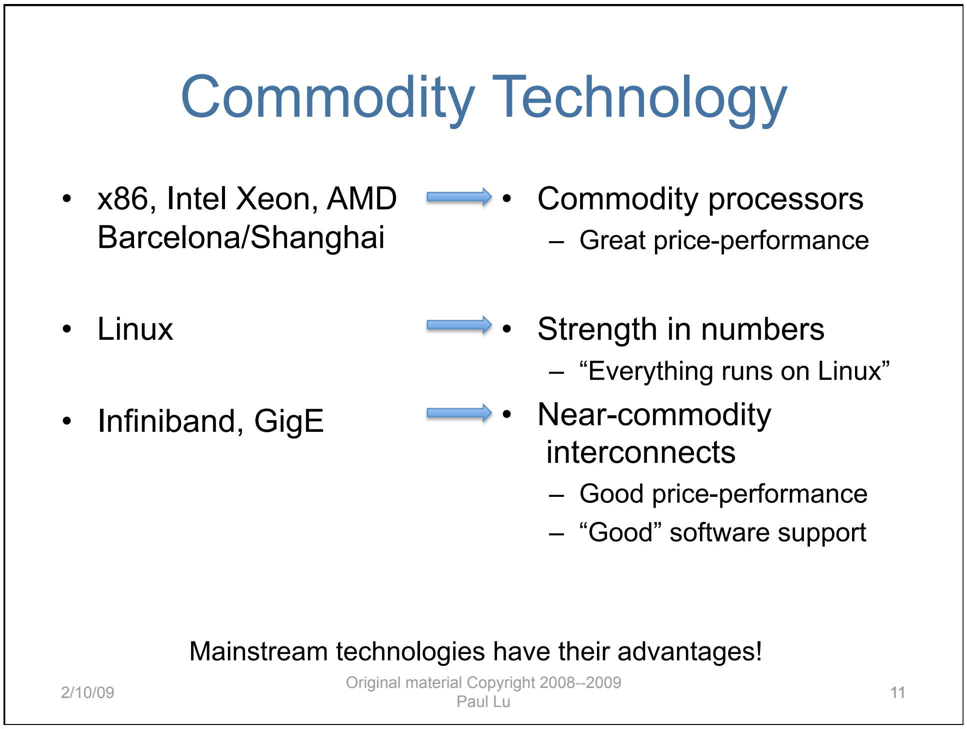


Cluster Challenge 2008

- 6 of 7 teams used x86 CPUs.
  - Purdue used MIPS/SiCortex
- 5 of 7 teams used Intel Xeon
  - MIT used AMD Shanghai, plus MIPS/SiCortex
- 6 of 7 teams used Linux
  - Arizona State used Windows HPC
- 4 of 7 teams used Infiniband
  - Also, two used 10 GigE and one proprietary SiCorex

2/10/09

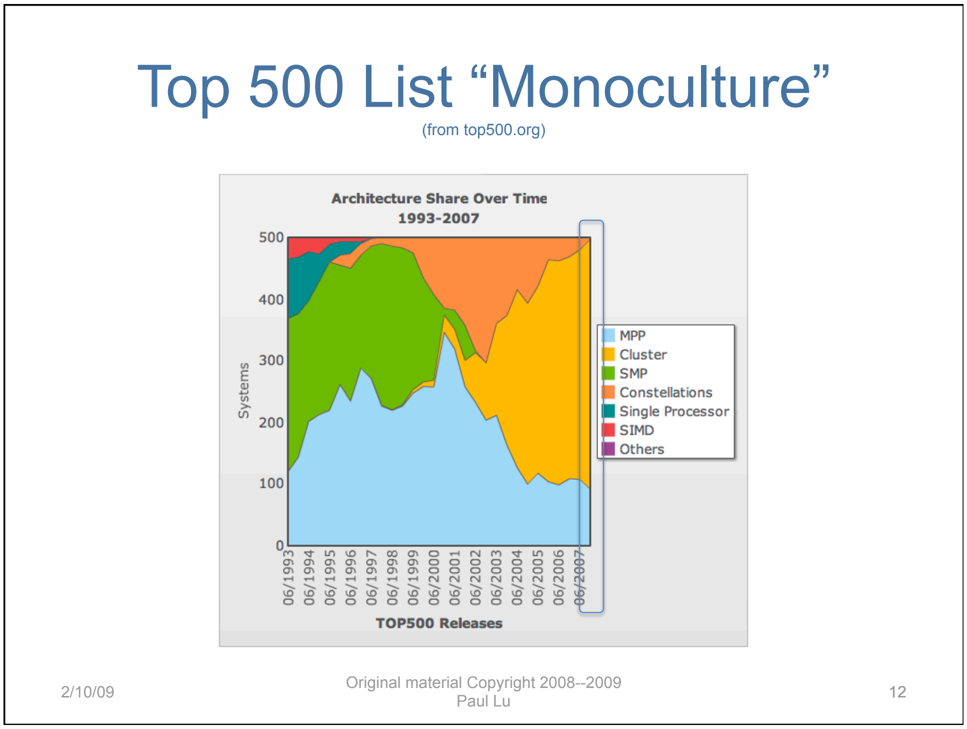
10



Commodity Technology

- x86, Intel Xeon, AMD Barcelona/Shanghai → Commodity processors
  - Great price-performance
- Linux → Strength in numbers
  - "Everything runs on Linux"
- Infiniband, GigE → Near-commodity interconnects
  - Good price-performance
  - "Good" software support

Mainstream technologies have their advantages!

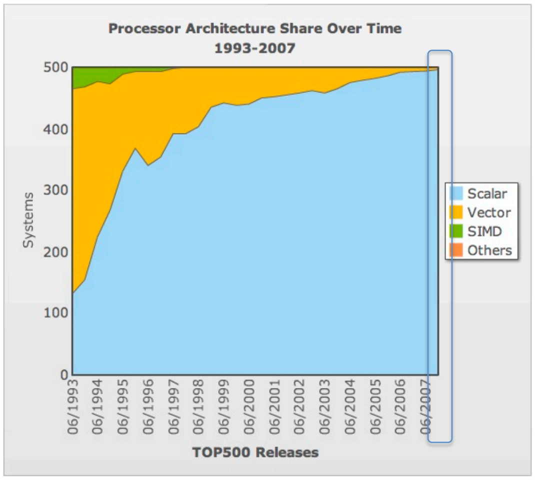


Top 500 List "Monoculture"

(from top500.org)

Top 500 List “Monoculture”

(from top500.org)



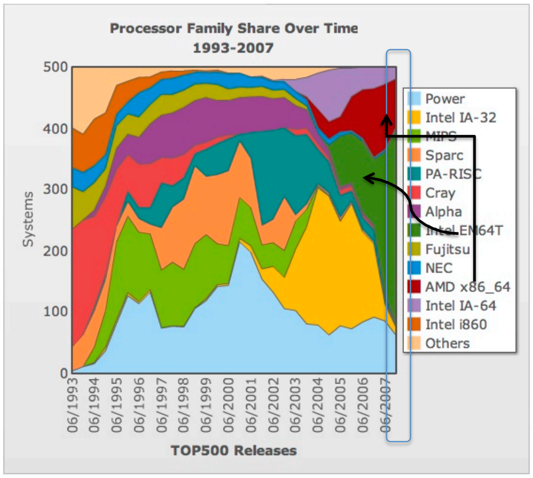
2/10/09

Original material Copyright 2008--2009
Paul Lu

13

Top 500 List “Monoculture”

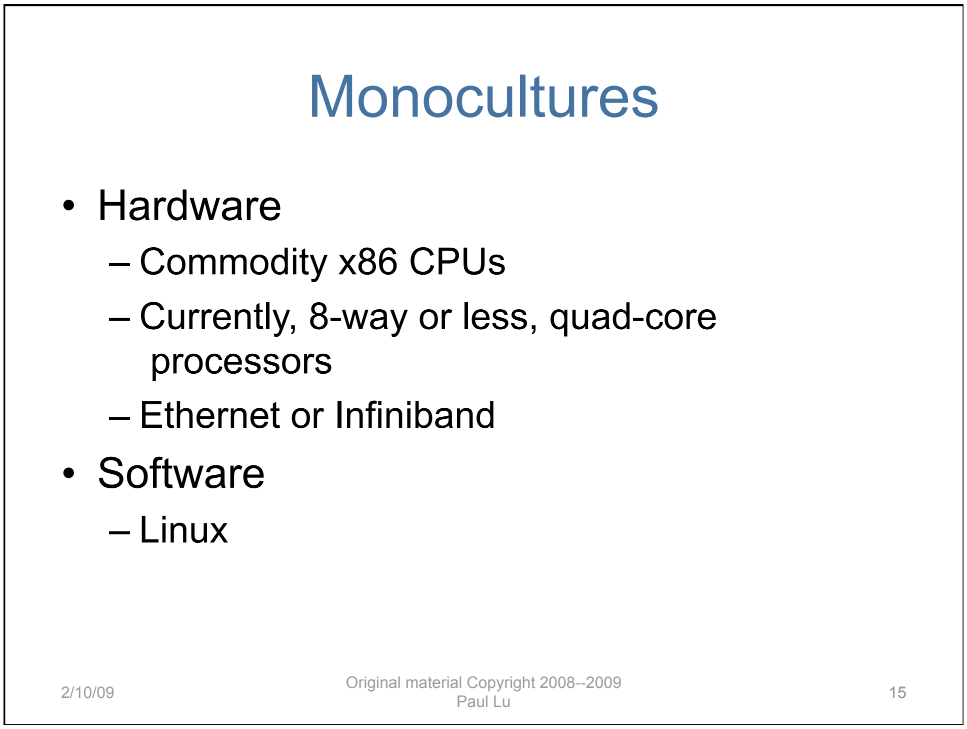
(from top500.org)



2/10/09

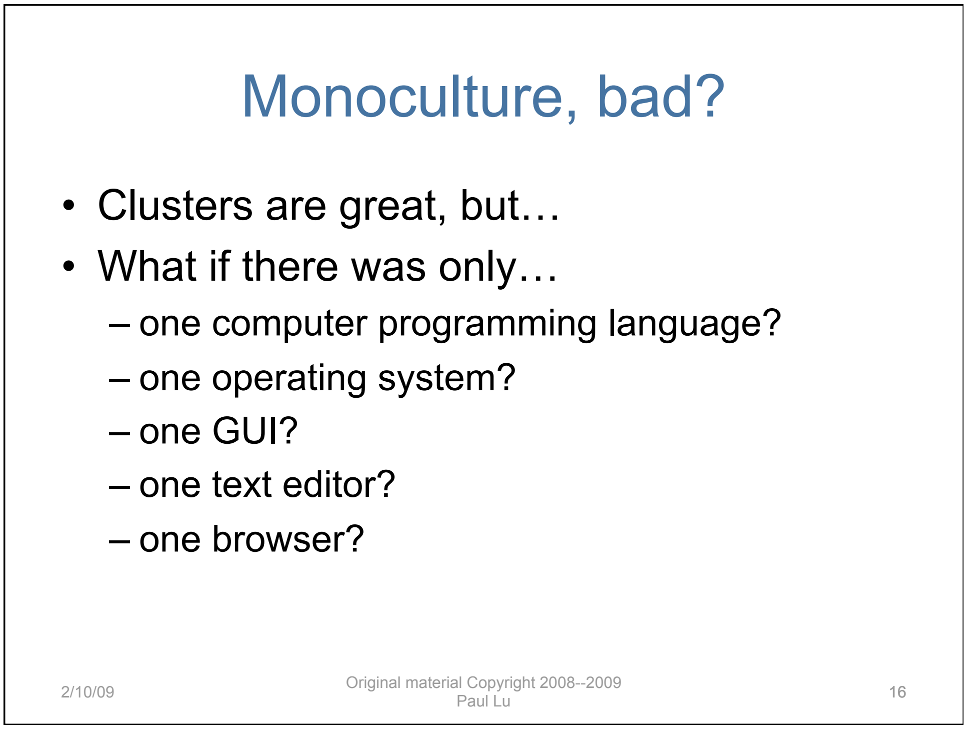
Original material Copyright 2008--2009
Paul Lu

14



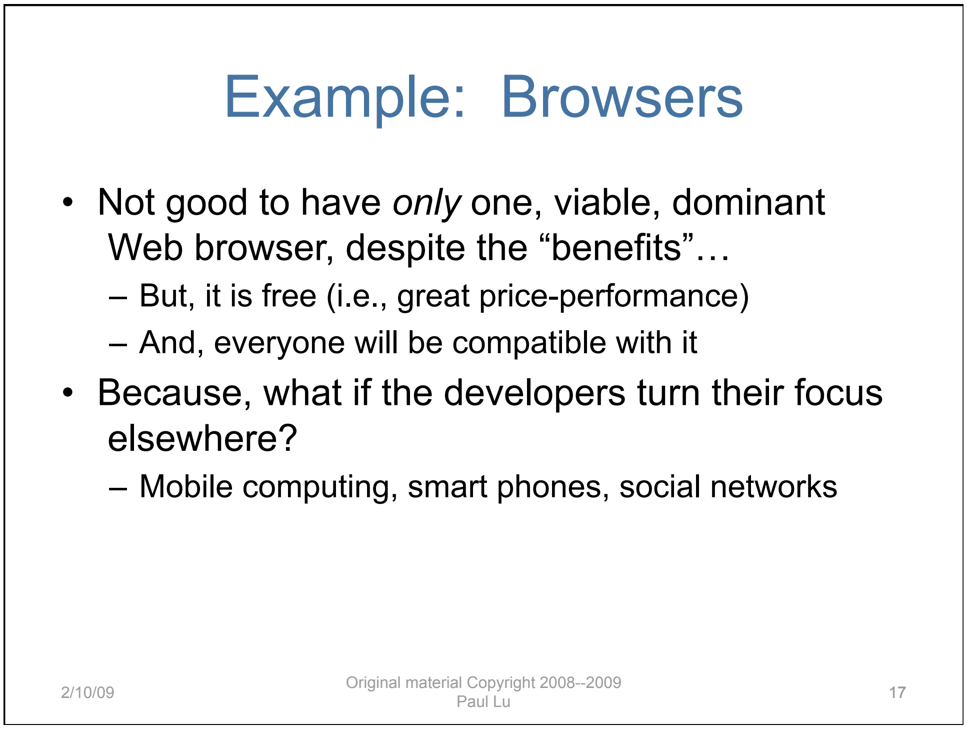
Monocultures

- Hardware
  - Commodity x86 CPUs
  - Currently, 8-way or less, quad-core processors
  - Ethernet or Infiniband
- Software
  - Linux



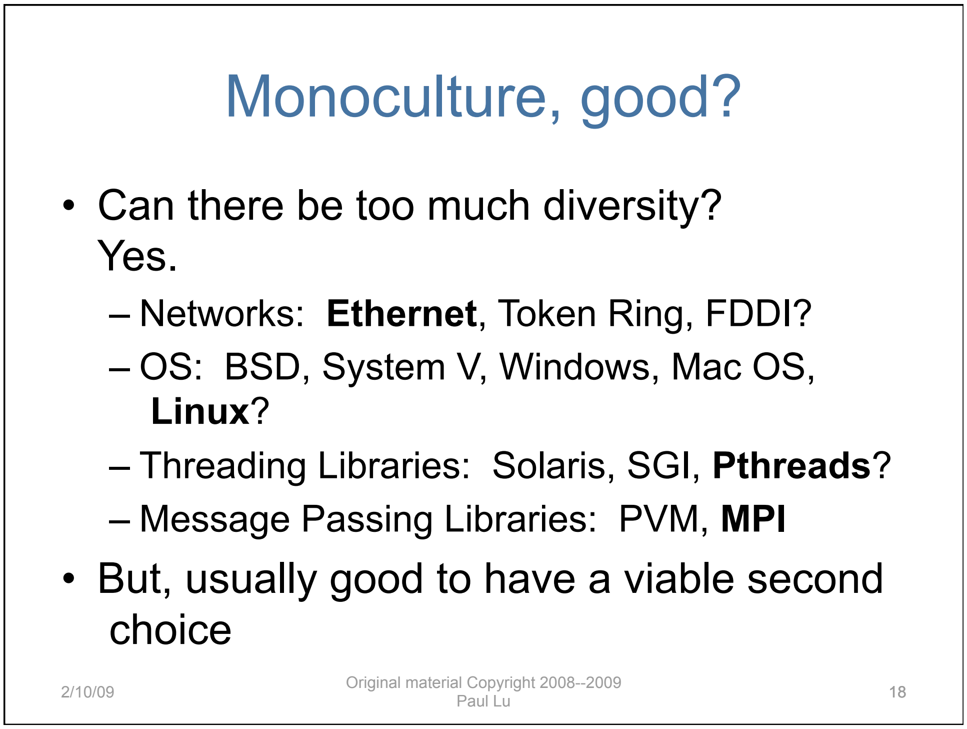
Monoculture, bad?

- Clusters are great, but…
- What if there was only…
  - one computer programming language?
  - one operating system?
  - one GUI?
  - one text editor?
  - one browser?



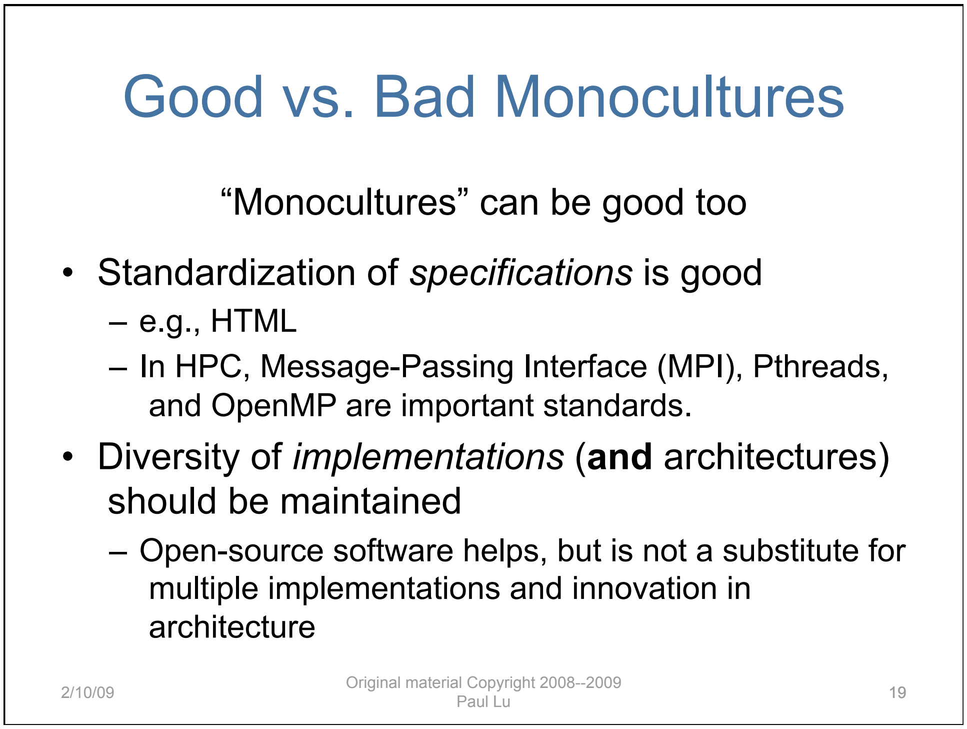
Example: Browsers

- Not good to have *only* one, viable, dominant Web browser, despite the “benefits”…
  - But, it is free (i.e., great price-performance)
  - And, everyone will be compatible with it
- Because, what if the developers turn their focus elsewhere?
  - Mobile computing, smart phones, social networks



Monoculture, good?

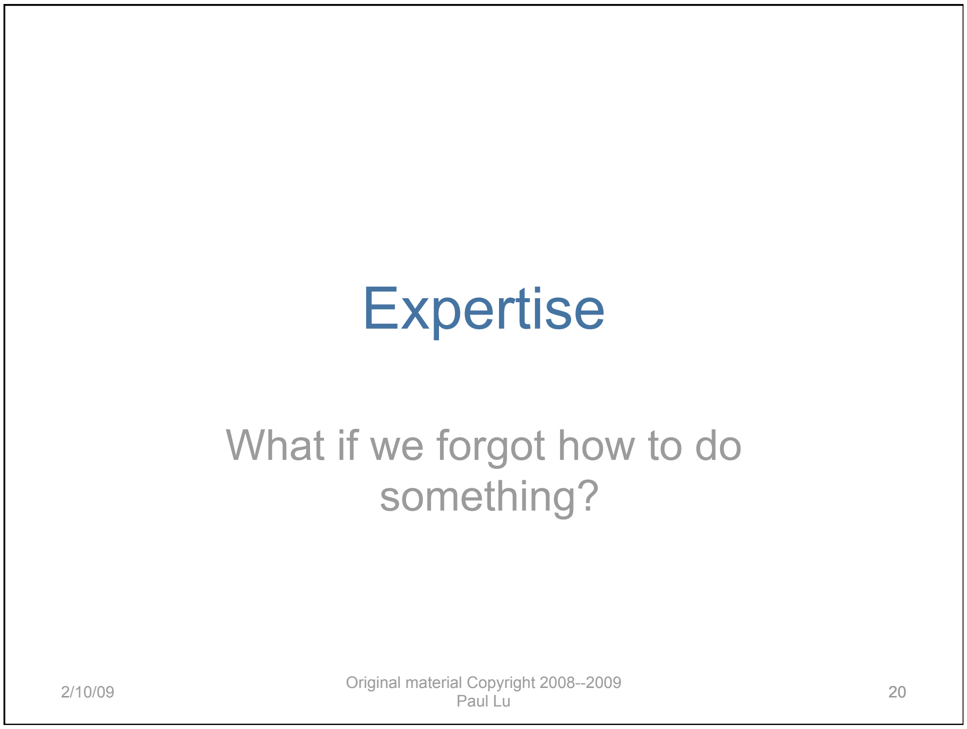
- Can there be too much diversity?
  Yes.
  - Networks: **Ethernet**, Token Ring, FDDI?
  - OS: BSD, System V, Windows, Mac OS, **Linux**?
  - Threading Libraries: Solaris, SGI, **Pthreads**?
  - Message Passing Libraries: PVM, **MPI**
- But, usually good to have a viable second choice



Good vs. Bad Monocultures

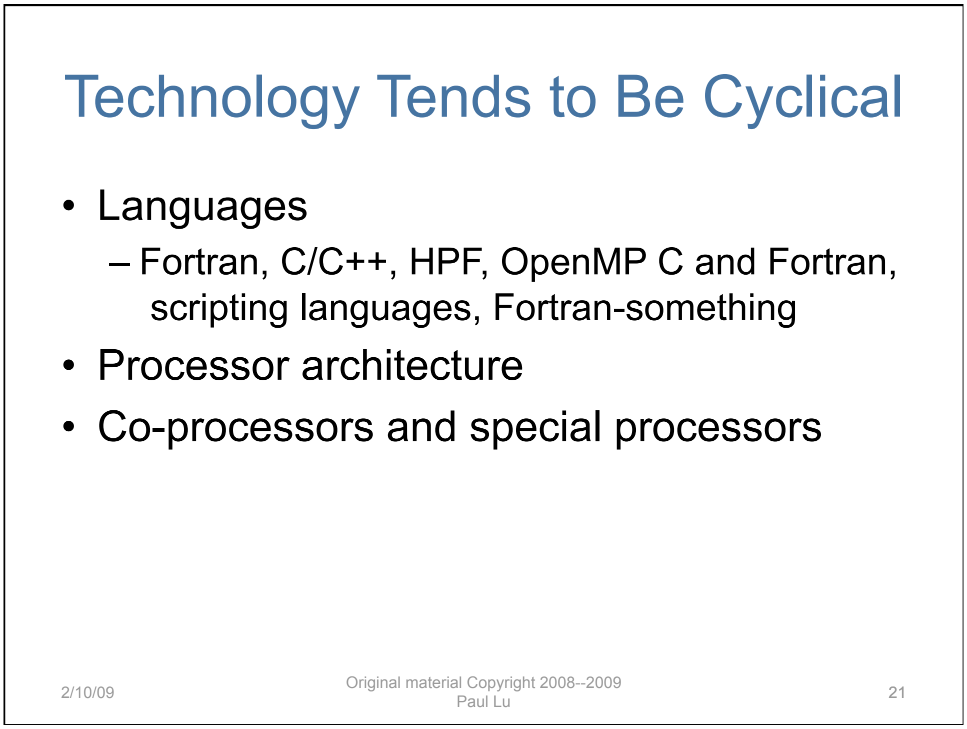
"Monocultures" can be good too

- Standardization of *specifications* is good
  - e.g., HTML
  - In HPC, Message-Passing Interface (MPI), Pthreads, and OpenMP are important standards.
- Diversity of *implementations* (**and** architectures) should be maintained
  - Open-source software helps, but is not a substitute for multiple implementations and innovation in architecture



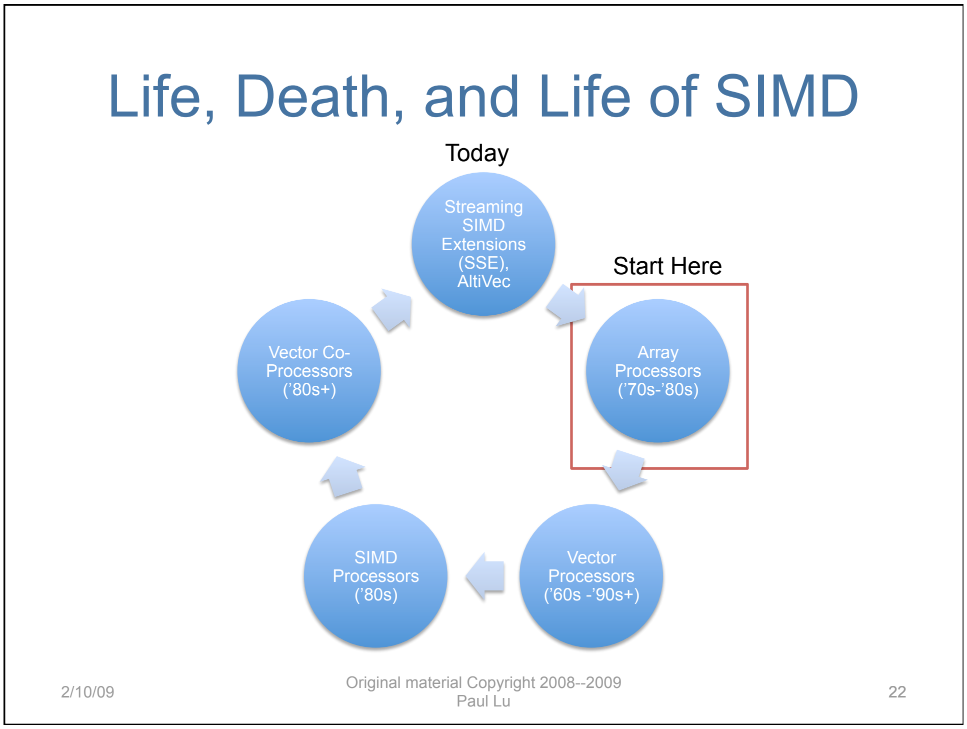
Expertise

What if we forgot how to do something?



Technology Tends to Be Cyclical

- Languages
  - Fortran, C/C++, HPF, OpenMP C and Fortran, scripting languages, Fortran-something
- Processor architecture
- Co-processors and special processors



Life, Death, and Life of SIMD

Today

Streaming SIMD Extensions (SSE), AltiVec

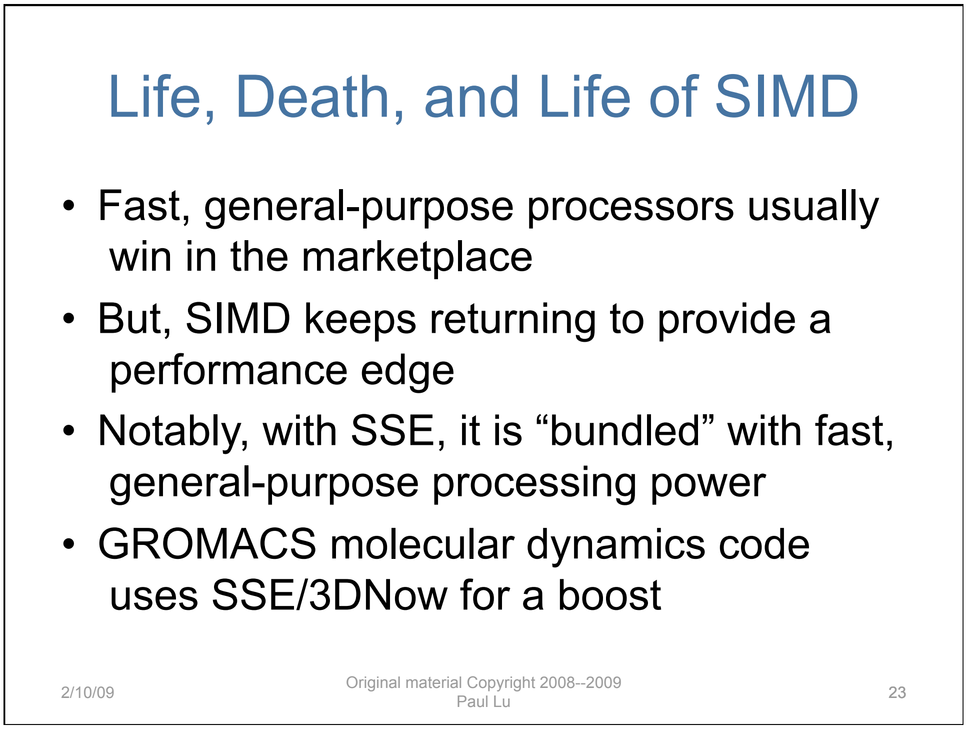
Start Here

Array Processors ('70s-'80s)

Vector Processors ('60s -'90s+)

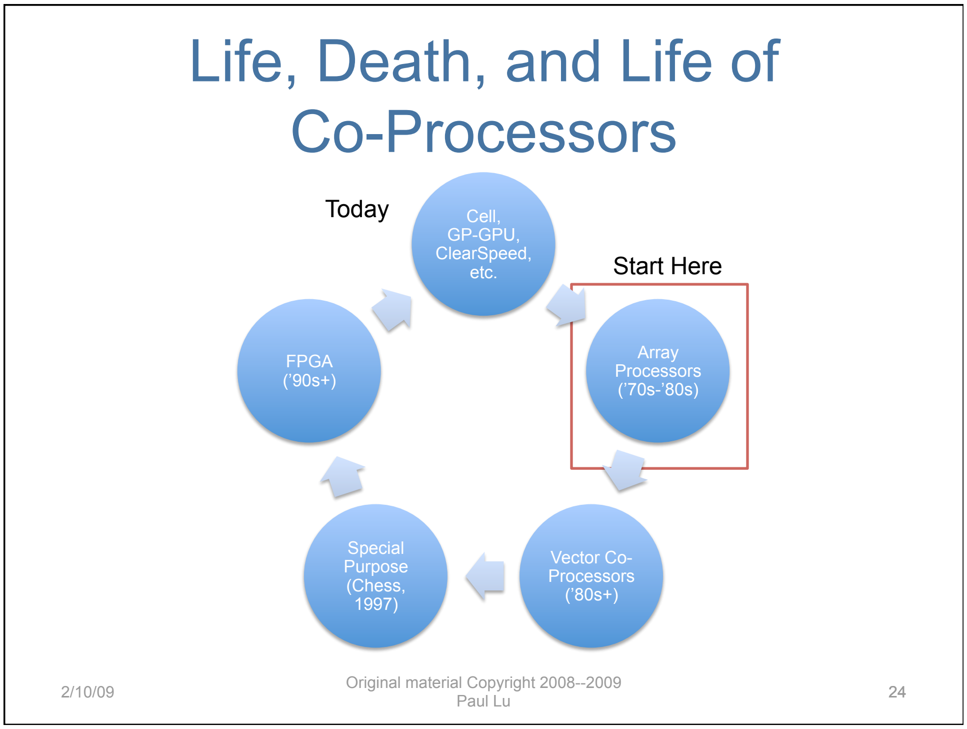
SIMD Processors ('80s)

Vector Co-Processors ('80s+)



Life, Death, and Life of SIMD

- Fast, general-purpose processors usually win in the marketplace
- But, SIMD keeps returning to provide a performance edge
- Notably, with SSE, it is “bundled” with fast, general-purpose processing power
- GROMACS molecular dynamics code uses SSE/3DNow for a boost



Life, Death, and Life of Co-Processors

Today

Cell, GP-GPU, ClearSpeed, etc.

Start Here

Array Processors ('70s-'80s)

Vector Co-Processors ('80s+)

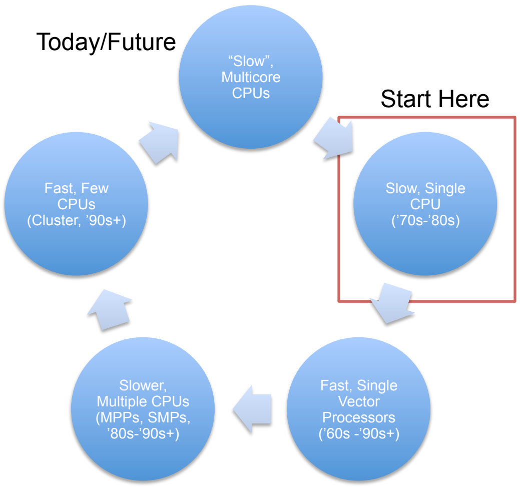
Special Purpose (Chess, 1997)

FPGA ('90s+)

Life, Death, and Life of Co-Processors

- As with SSE, GP-GPU may become a "bundled" feature of fast processors
- Upcoming Mac OS "Snow Leopard" supports using GPUs
- Loss of low-level architectural and coding skills is quite common
- Of course, do NOT jump on every bandwagon. But, remember how to jump.

Life, Death, and Life of "Massive Parallelism"


Today/Future
"Slow", Multicore CPUs
Start Here
Slow, Single CPU ('70s-'80s)
Fast, Single Vector Processors ('60s -'90s+)
Slower, Multiple CPUs (MPPs, SMPs, '80s-'90s+)
Fast, Few CPUs (Cluster, '90s+)

Life, Death, and Life of "Massive Parallelism"

- Next era will NOT provide "automatic" performance improvements each year via clock speed.
- Multicore may not be as well-suited for capacity computing
  - Multiple jobs sharing a processor/socket contend for (depending on architecture) memory bandwidth, I/O bandwidth, etc.
  - Performance mainly through better use of cores per-job (i.e., parallelism)

What if we lost the skills?

- When was the last time your group wrote a *new* Fortran program?
- Is the SSE/AltiVec capability of your CPU used at all?
  - Intel is moving towards Advanced Vector Extensions, 256-bits
- Who in your group can scale your application if you had 64 cores sharing a common memory?

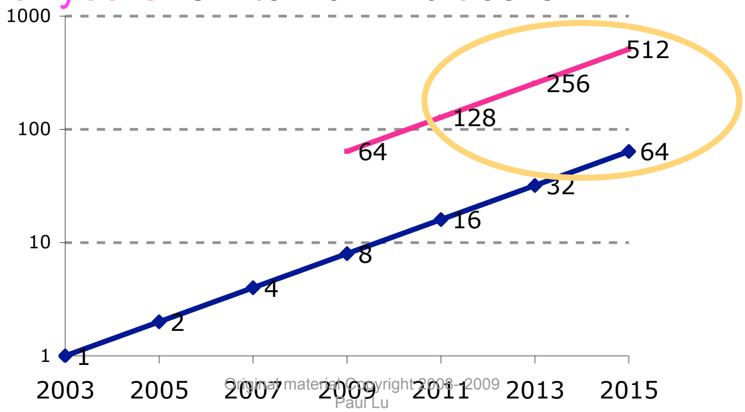
What if we missed the boat?

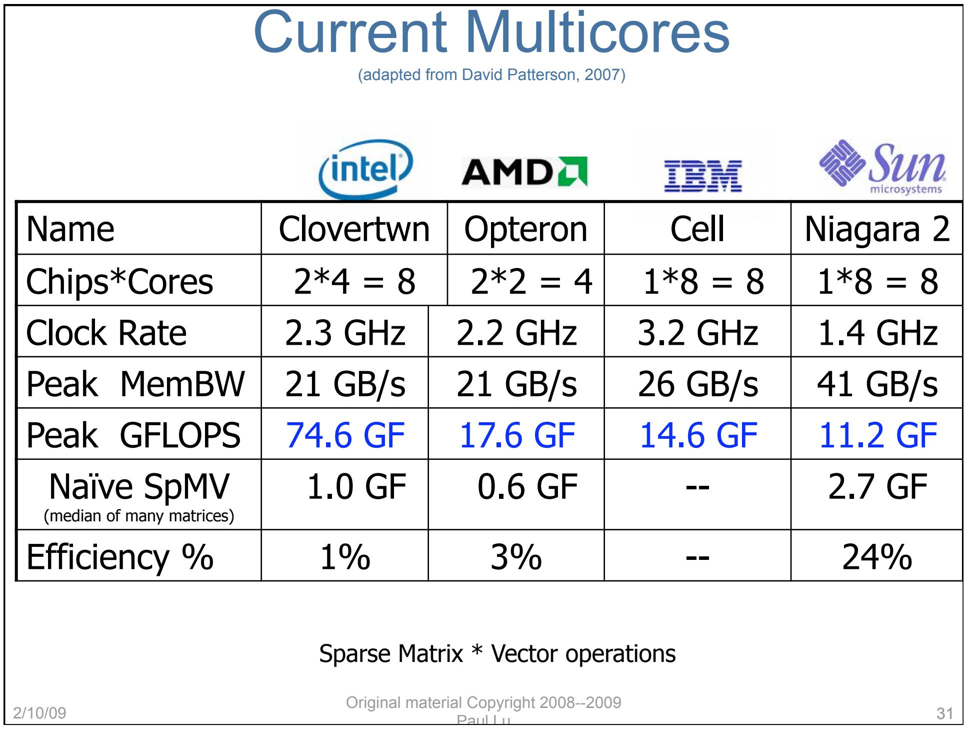
- We can foresee the days of 64 cores and more on one socket
- These look a lot like SMPs
- How will we deal with multicore and manycore systems if our platforms are currently 8- or 16-way?

Why Target 64+ Cores?

(adapted from David Patterson, 2007)

- Multicore: 2X / 2 yrs ⇒ ≈ 64 cores in 8 years
- Manycore: 8X to 16X multicore



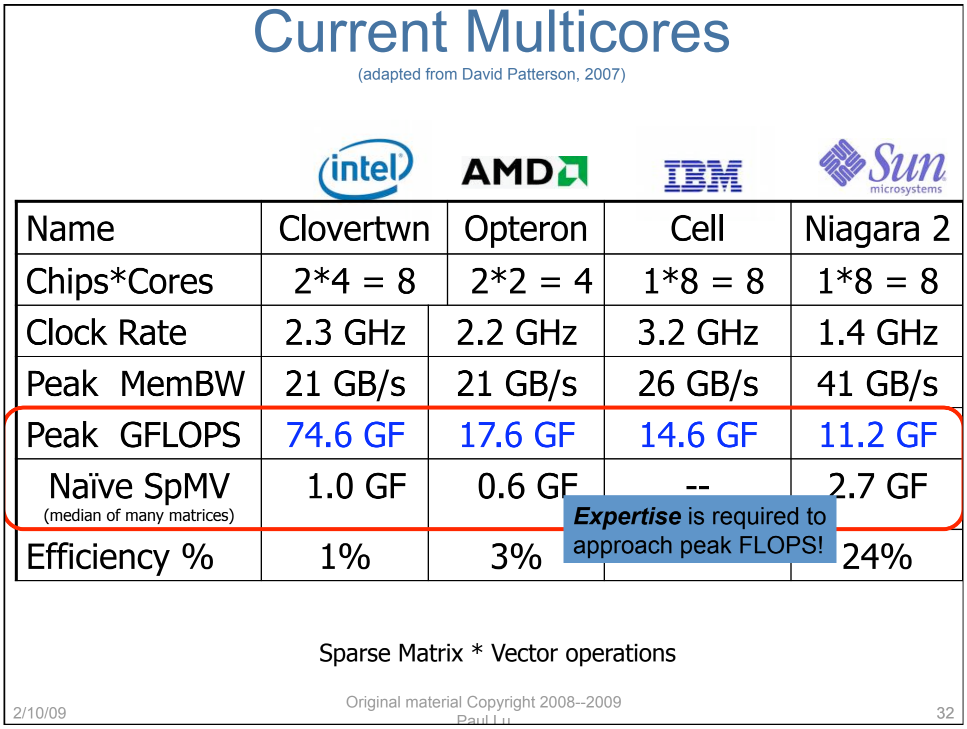


Current Multicores

(adapted from David Patterson, 2007)

	intel	AMD	IBM	Sun microsystems
Name	Clovertwn	Opteron	Cell	Niagara 2
Chips*Cores	2*4 = 8	2*2 = 4	1*8 = 8	1*8 = 8
Clock Rate	2.3 GHz	2.2 GHz	3.2 GHz	1.4 GHz
Peak MemBW	21 GB/s	21 GB/s	26 GB/s	41 GB/s
Peak GFLOPS	74.6 GF	17.6 GF	14.6 GF	11.2 GF
Naïve SpMV (median of many matrices)	1.0 GF	0.6 GF	--	2.7 GF
Efficiency %	1%	3%	--	24%

Sparse Matrix * Vector operations



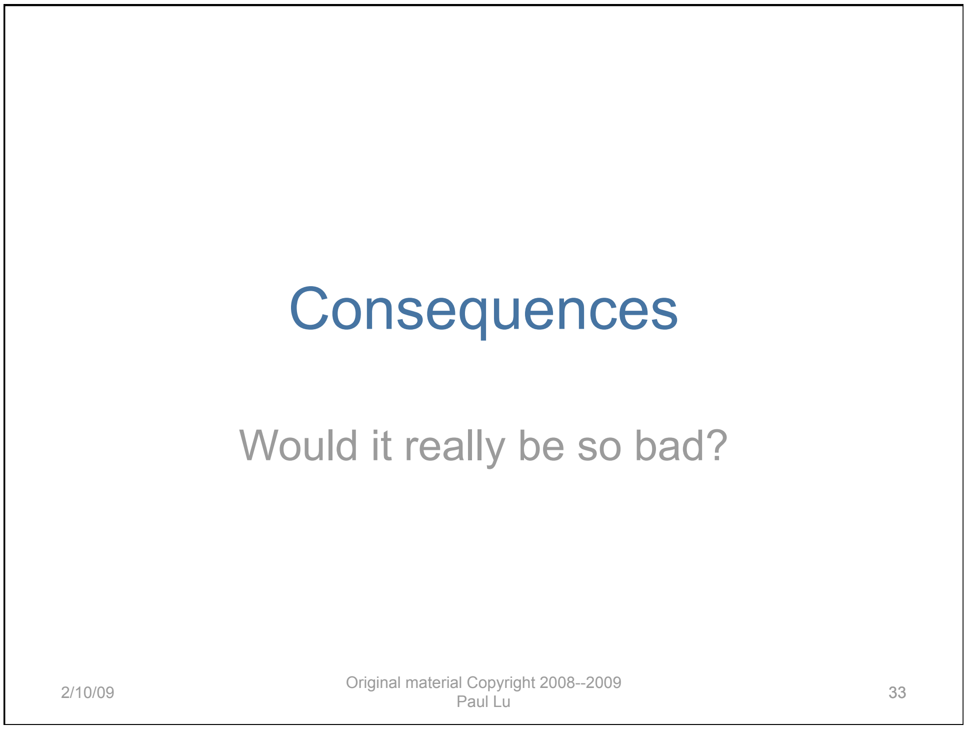
Current Multicores

(adapted from David Patterson, 2007)

	intel	AMD	IBM	Sun microsystems
Name	Clovertwn	Opteron	Cell	Niagara 2
Chips*Cores	2*4 = 8	2*2 = 4	1*8 = 8	1*8 = 8
Clock Rate	2.3 GHz	2.2 GHz	3.2 GHz	1.4 GHz
Peak MemBW	21 GB/s	21 GB/s	26 GB/s	41 GB/s
Peak GFLOPS	74.6 GF	17.6 GF	14.6 GF	11.2 GF
Naïve SpMV (median of many matrices)	1.0 GF	0.6 GF	--	2.7 GF
Efficiency %	1%	3%	--	24%

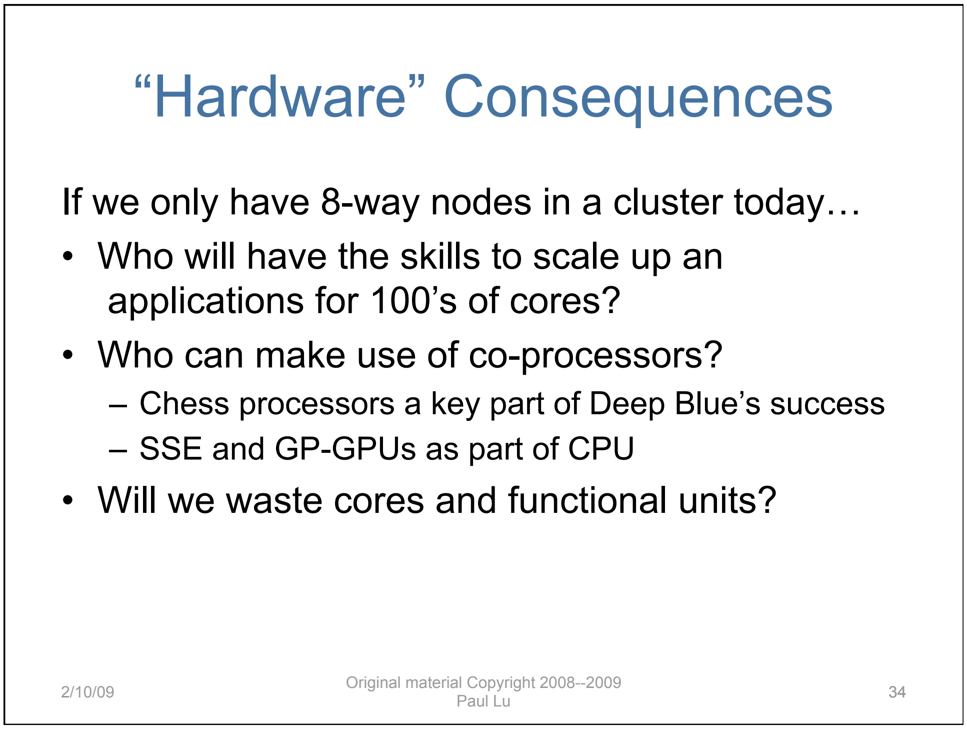
Expertise is required to approach peak FLOPS!

Sparse Matrix * Vector operations



Consequences

Would it really be so bad?



“Hardware” Consequences

If we only have 8-way nodes in a cluster today…

- Who will have the skills to scale up an applications for 100’s of cores?
- Who can make use of co-processors?
  - Chess processors a key part of Deep Blue’s success
  - SSE and GP-GPUs as part of CPU
- Will we waste cores and functional units?

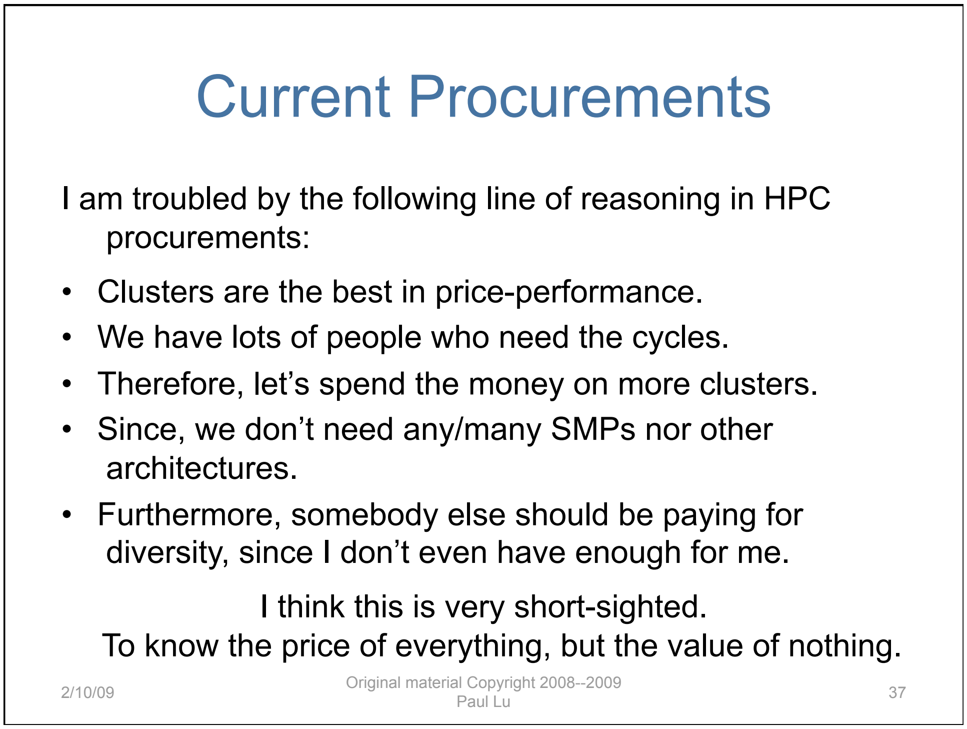
“Software” Consequences

If Linux is the “only” OS

- Who will work on scalability and NUMA issues?
  - Linux may become focused on mobile devices, or Web servers, or the next trend
- What if that desirable feature is not available on Linux (e.g., ZFS and license conflict)?
- Will we have “golden handcuffs”?

Research Consequences

- Using co-processors, your research competitor beats you to the answer
  - Deep Blue (1997) co-processors in chess
- Stagnant innovation
  - The history of browsers, compilers, etc.
- No platforms for capability applications and algorithm development
  - Latency-sensitive, irregular applications

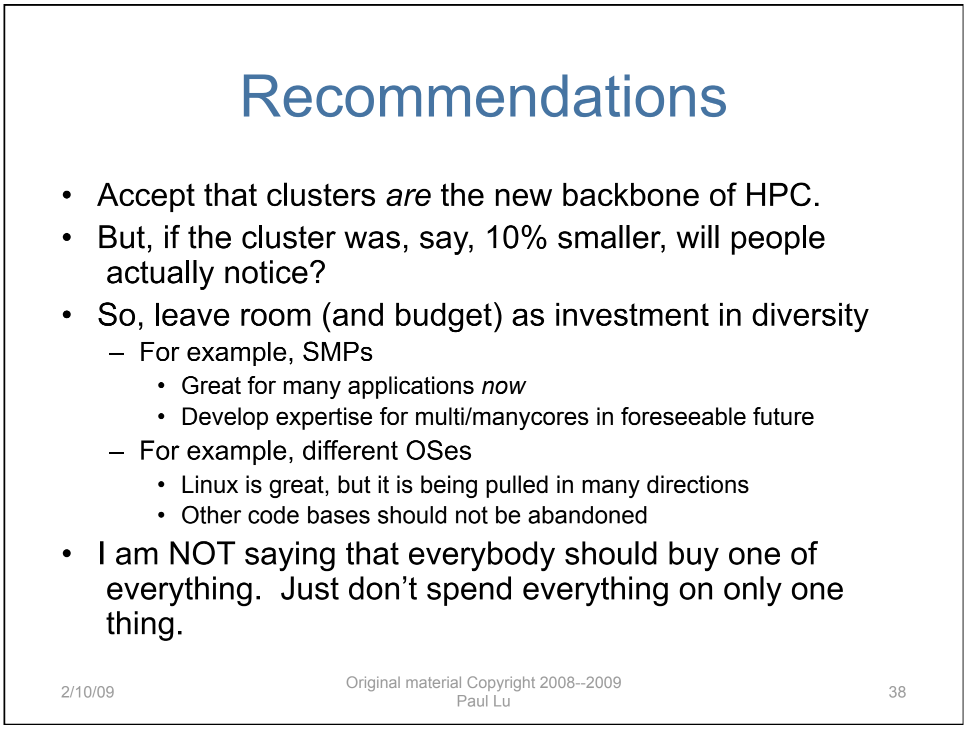


Current Procurements

I am troubled by the following line of reasoning in HPC procurements:

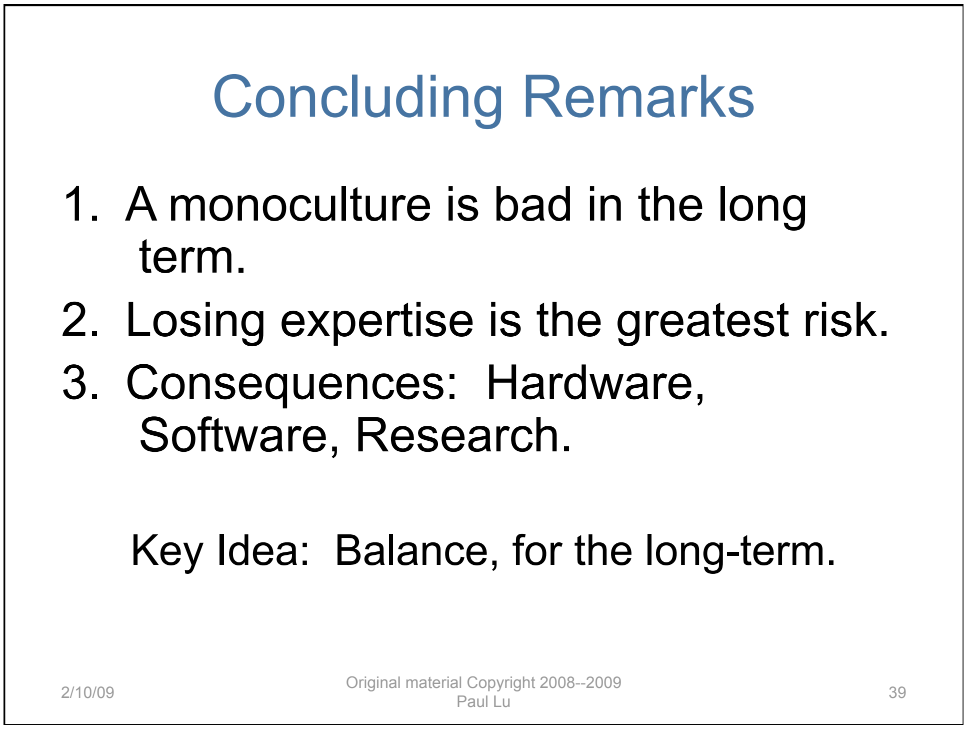
- Clusters are the best in price-performance.
- We have lots of people who need the cycles.
- Therefore, let’s spend the money on more clusters.
- Since, we don’t need any/many SMPs nor other architectures.
- Furthermore, somebody else should be paying for diversity, since I don’t even have enough for me.

I think this is very short-sighted.
To know the price of everything, but the value of nothing.



Recommendations

- Accept that clusters *are* the new backbone of HPC.
- But, if the cluster was, say, 10% smaller, will people actually notice?
- So, leave room (and budget) as investment in diversity
  - For example, SMPs
    - Great for many applications *now*
    - Develop expertise for multi/manycores in foreseeable future
  - For example, different OSes
    - Linux is great, but it is being pulled in many directions
    - Other code bases should not be abandoned
- I am NOT saying that everybody should buy one of everything. Just don’t spend everything on only one thing.



Concluding Remarks

1. A monoculture is bad in the long term.
2. Losing expertise is the greatest risk.
3. Consequences: Hardware, Software, Research.

Key Idea: Balance, for the long-term.