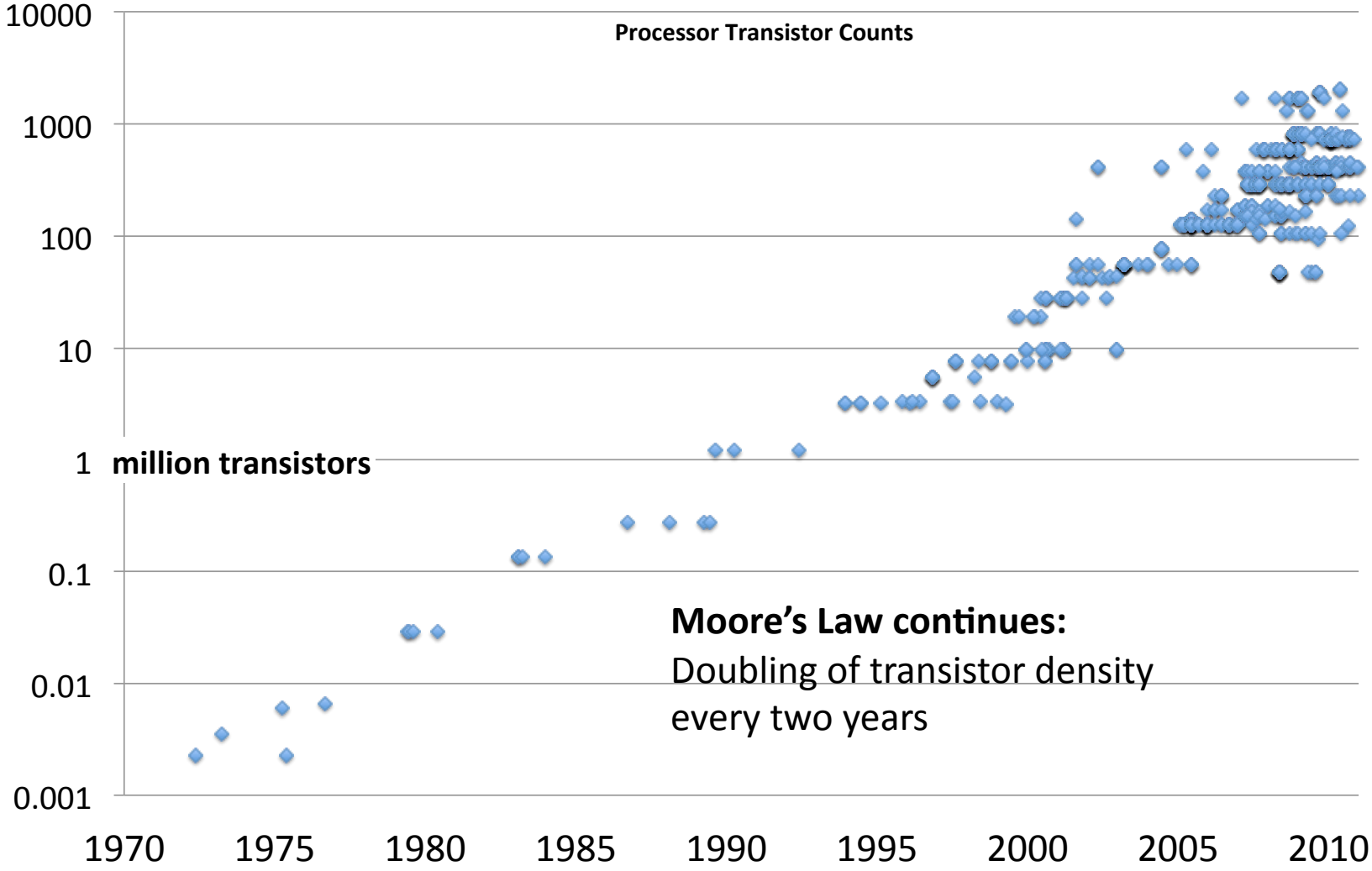
Hardware Trends in HPC

A Report from SC2011

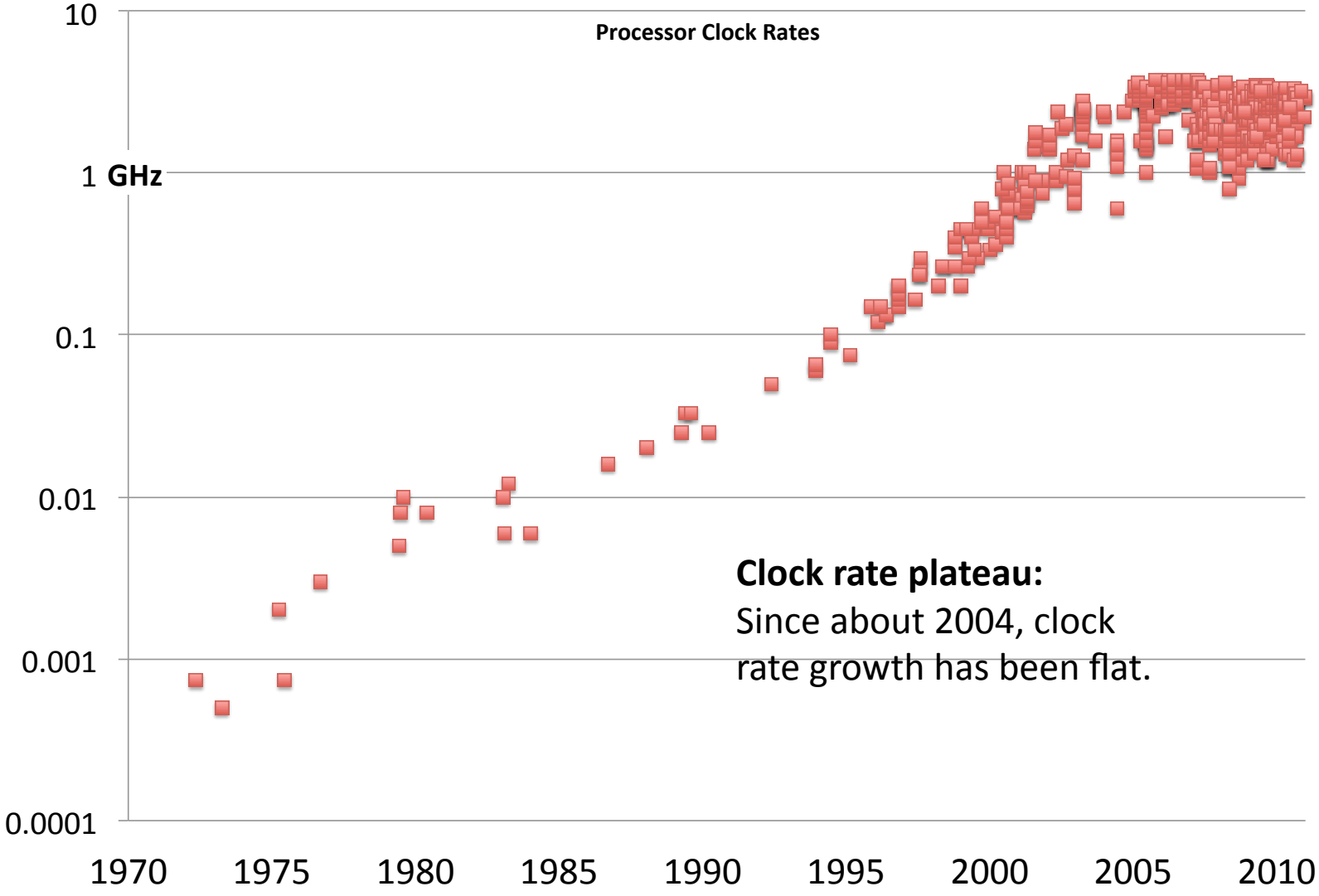
Credits

- These slides were extracted from tutorials and technical presentations at SC2011. Copyrights retained by the original authors.
- Michael McCool, Arch Robinson and James Reinders from Intel Software to appear in the book "*Structured Parallel Programming Using Intel® Parallel Building Blocks*".
- Michael Wolfe, Portland Group
- Jeffrey Vetter, Georgia Tech and ORNL

Hardware Evolution: Transistors on a Chip



Hardware Evolution: Clock Rate



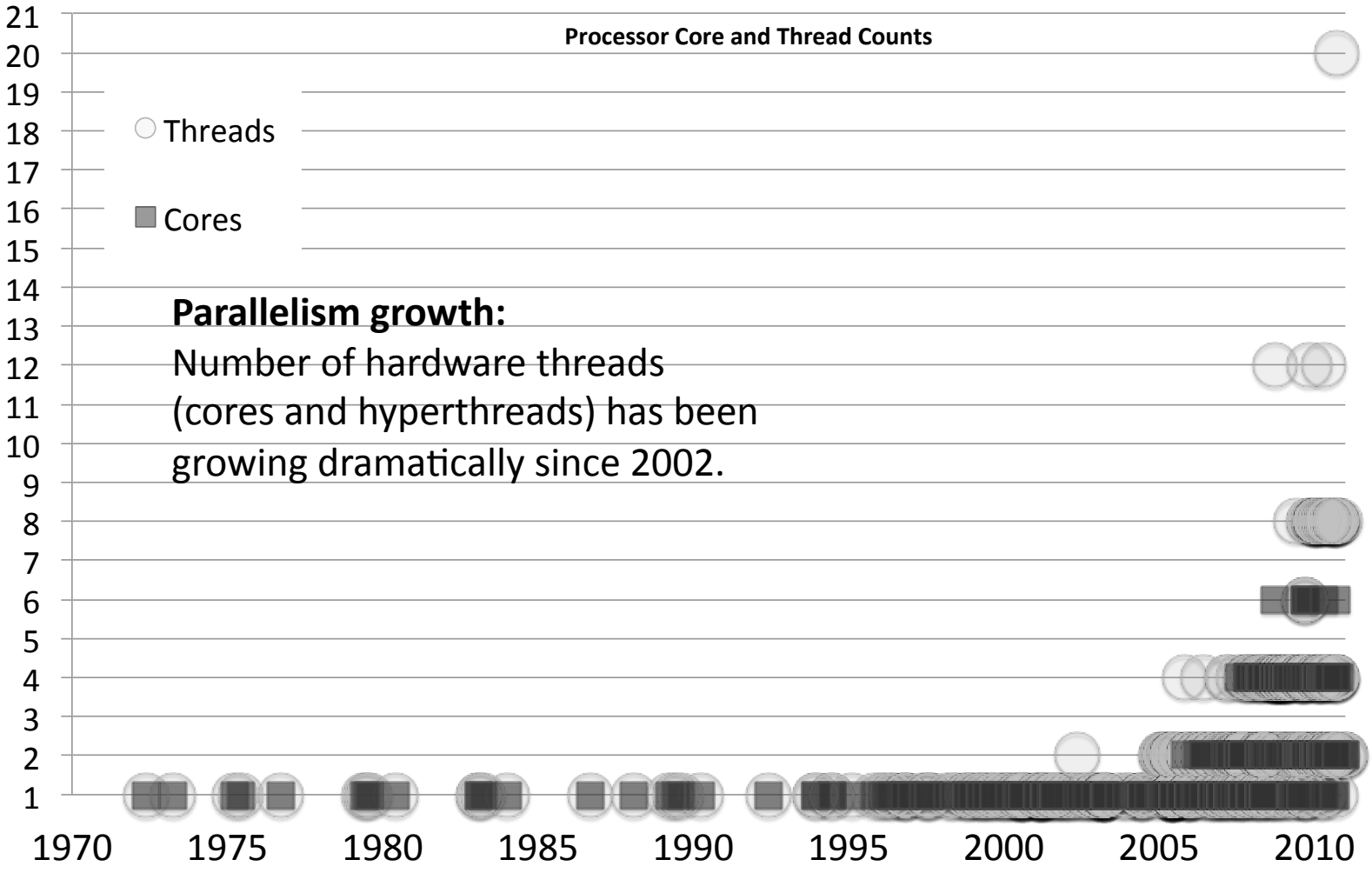
Hardware Evolution

There are limits to "automatic" improvement of scalar performance:

1. **The Power Wall:** Clock frequency cannot be increased without exceeding air cooling.
2. **The Memory Wall:** Access to data is a limiting factor.
3. **The ILP Wall:** All the existing instruction-level parallelism (ILP) is already being used.

→ **Conclusion:** Explicit parallel mechanisms and explicit parallel programming are *required* for performance scaling.

Hardware Parallelism: Cores and Threads



Parallelism growth:
Number of hardware threads (cores and hyperthreads) has been growing dramatically since 2002.

Intel Sandy Bridge

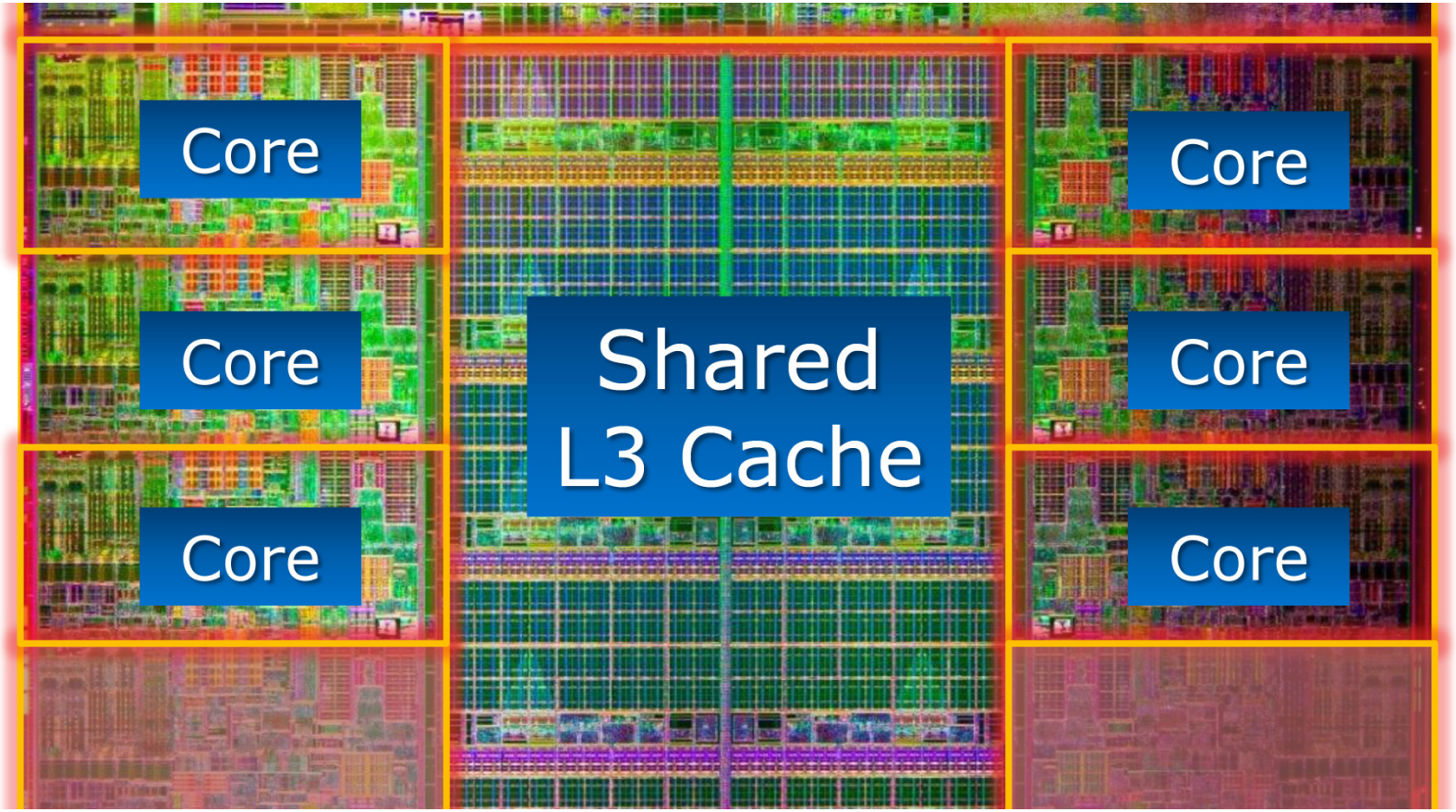
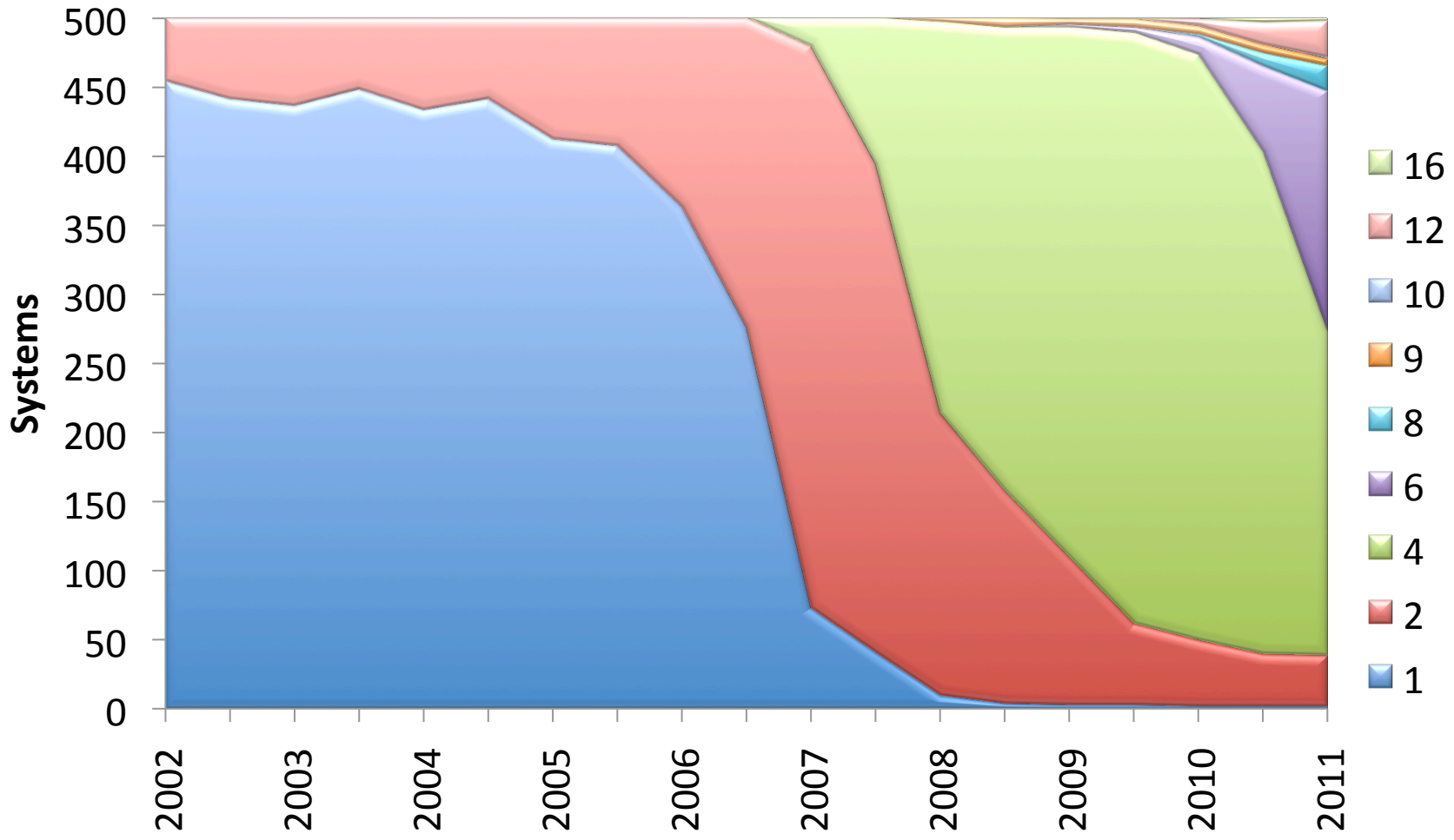


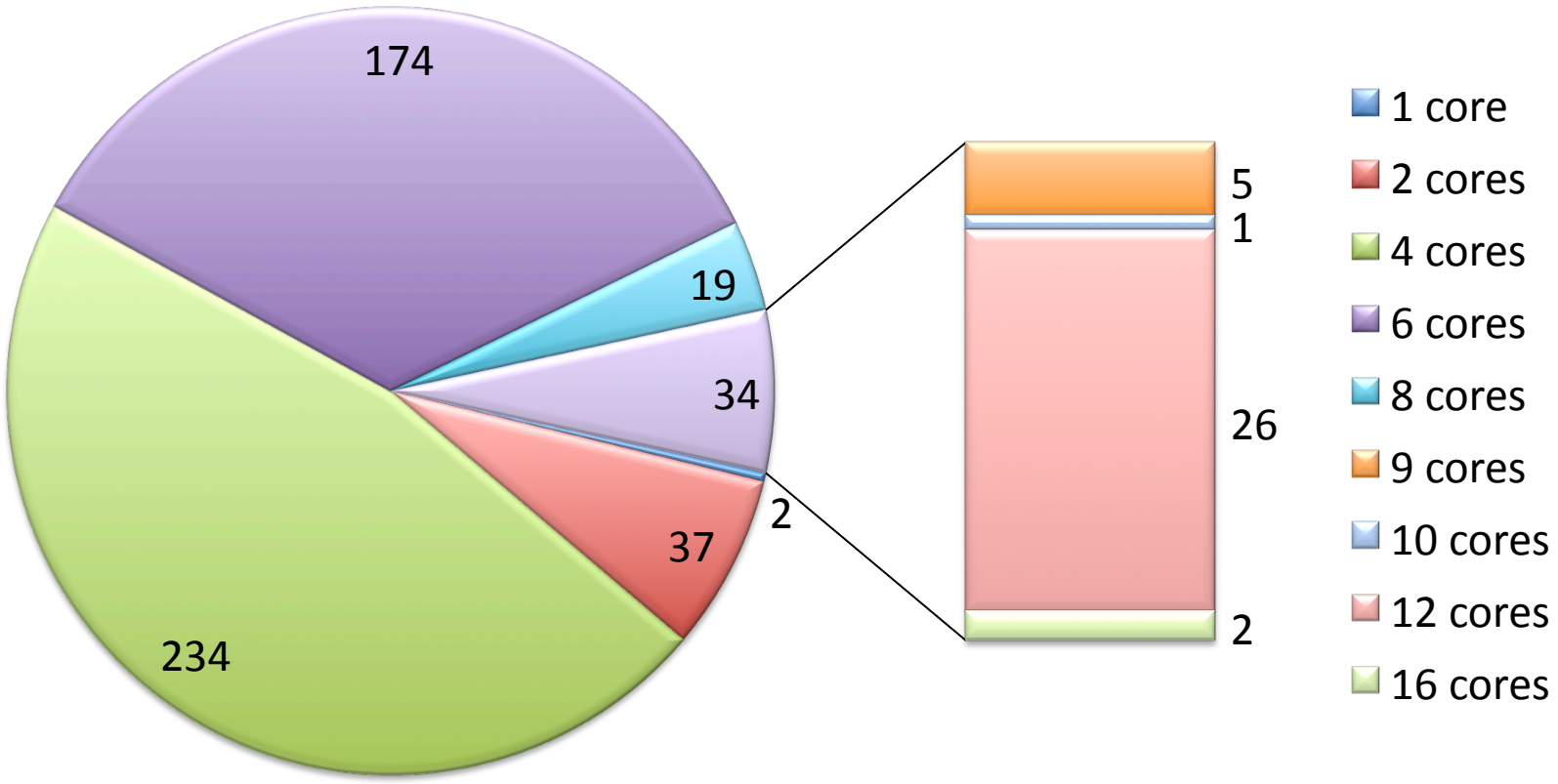
Photo: desktop die with two cores disabled

Intel Sandy Bridge: 8 cores / 16 threads
AMD Interlagos: 8 cores / 16 threads

TOP500: Cores per Socket

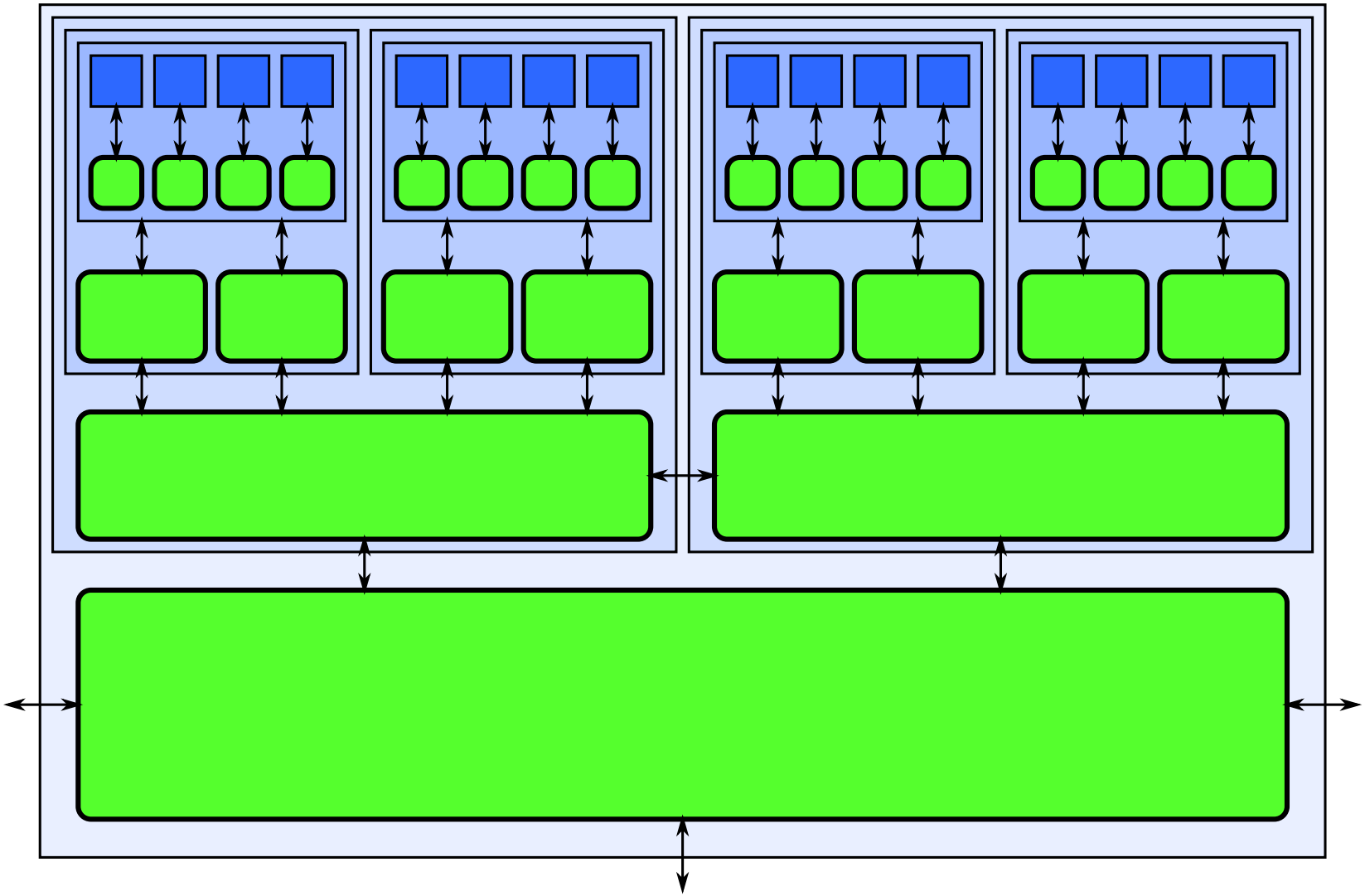


TOP500: Cores per Socket

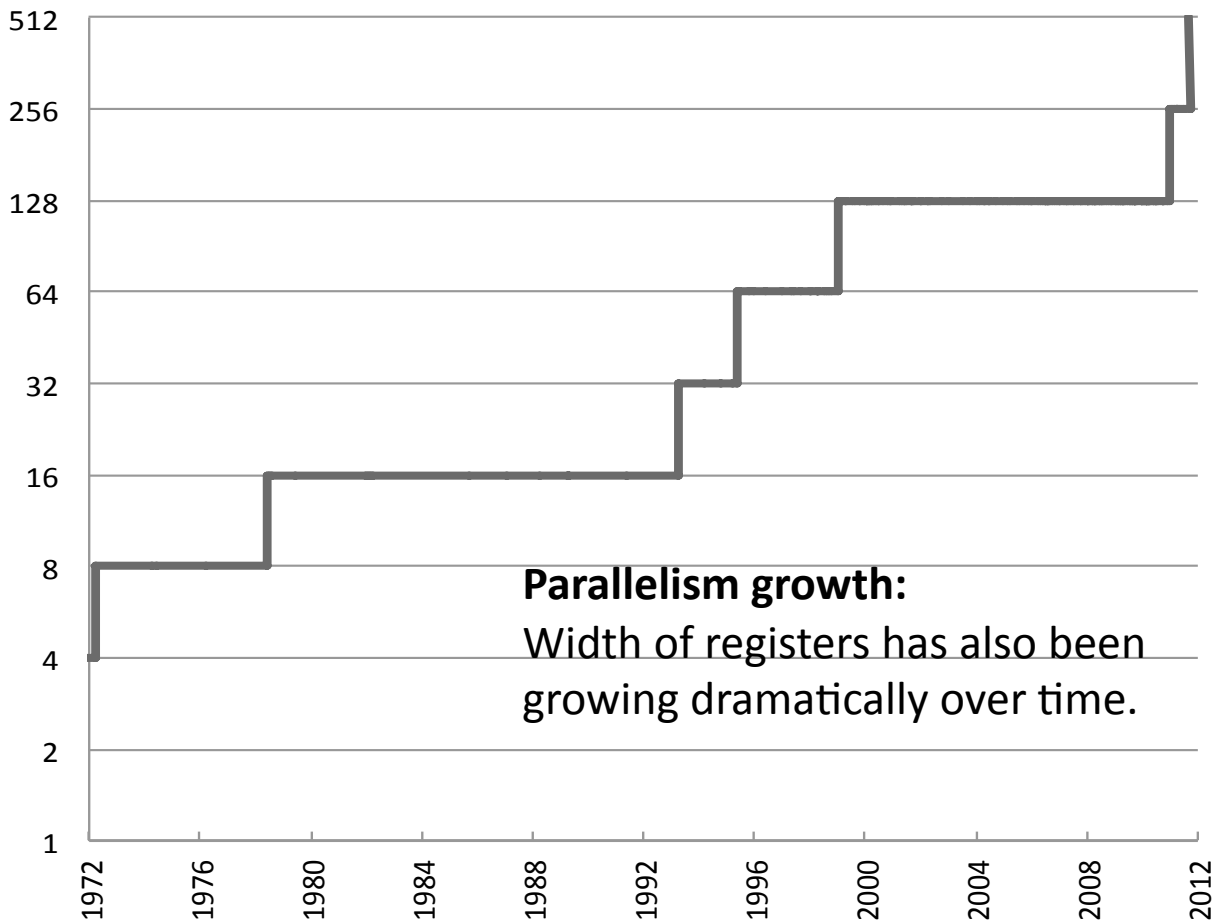


500 systems total

More Cores: Cache Hierarchy

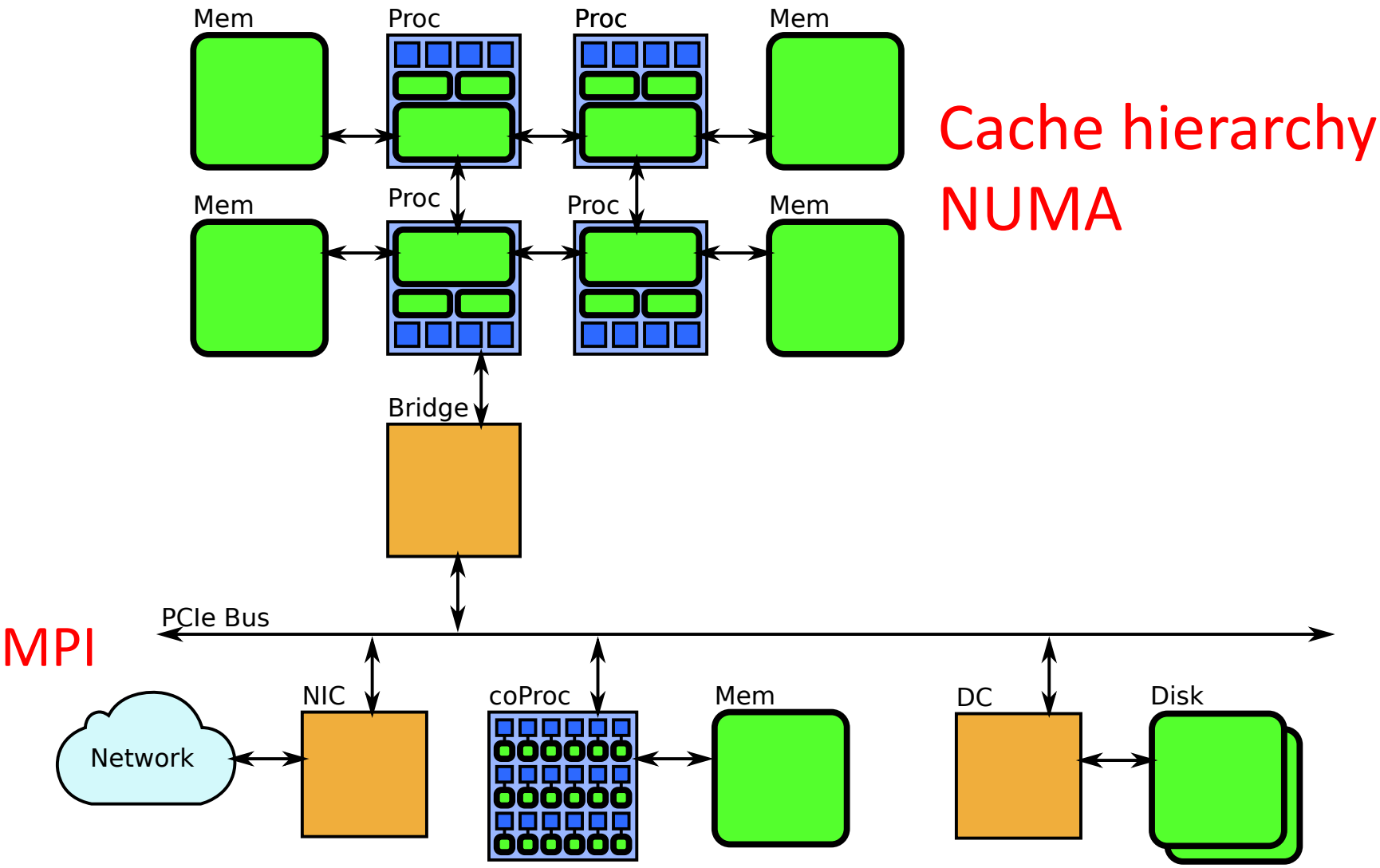


Vector Parallelism: Register Width



Vector Instruction sets e.g.: SSE4.x, AVX, FMA

Typical System Architecture



Cache hierarchy

NUMA

MPI

Heterogeneous (GPU) computing

Key Factors

Compute: Parallelism

What mechanisms do processors provide for using parallelism?

- Implicit: instruction pipelines, superscalar issues
- Explicit: cores, hyperthreads, vector units

- *How to map potential parallelism to actual parallelism?*

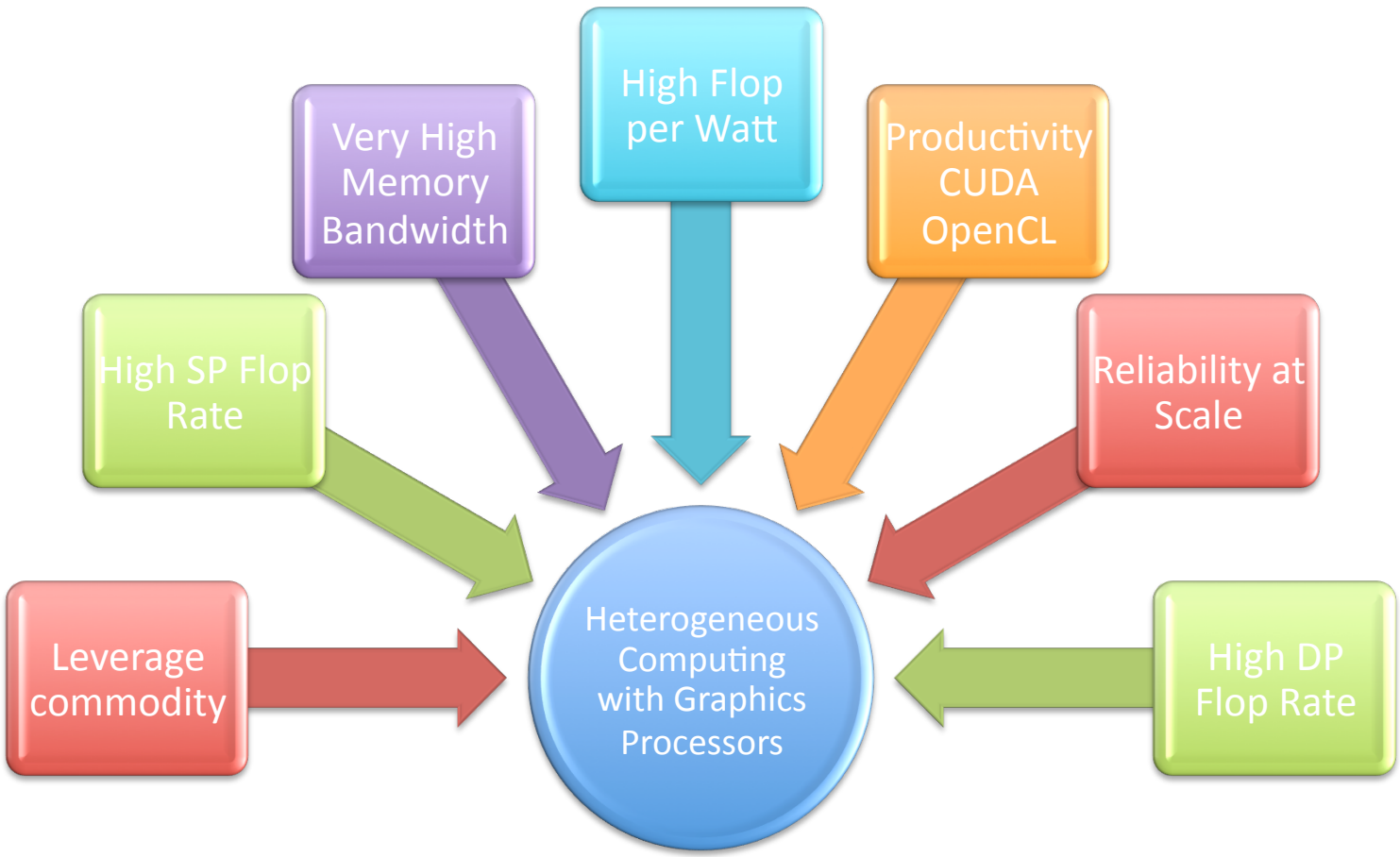
Data: Locality

How is data managed and accessed, and what are the performance implications?

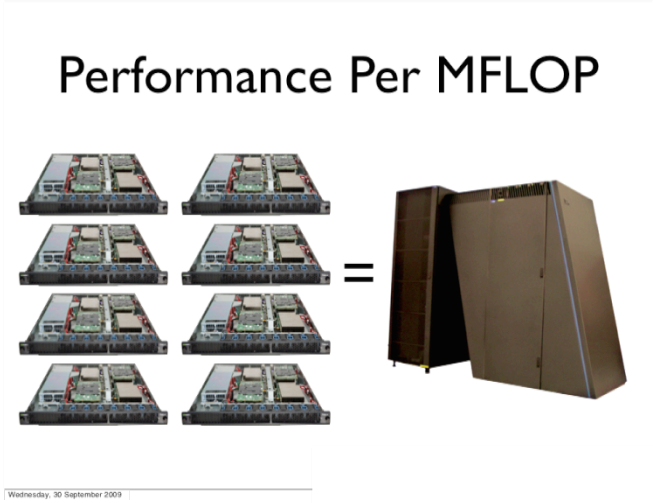
- Cache behavior, conflicts, sharing, coherency, (false) sharing; alignments with cache lines, pages, vector lanes
- NUMA

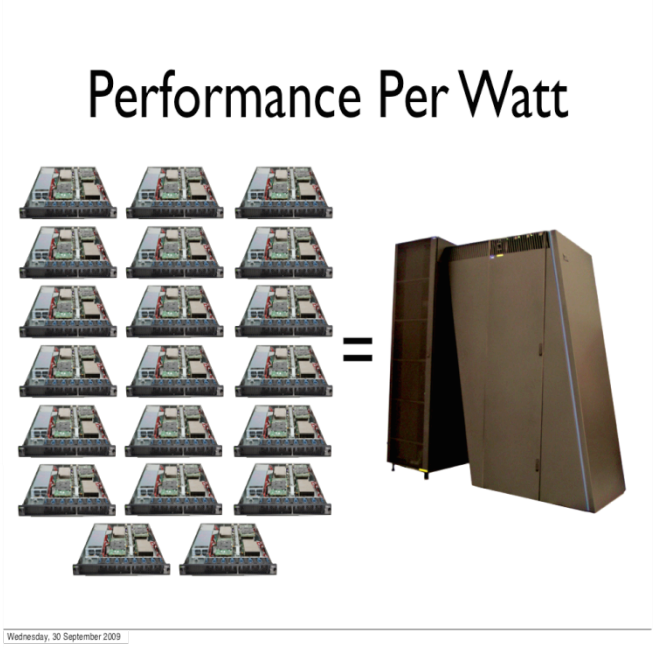
- *How to design algorithms that have good data locality?*

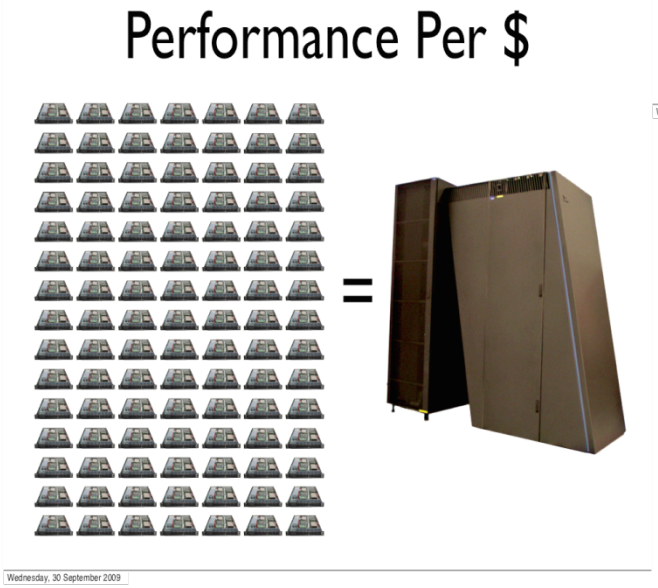
Heterogeneous Computing the GPU Rationale



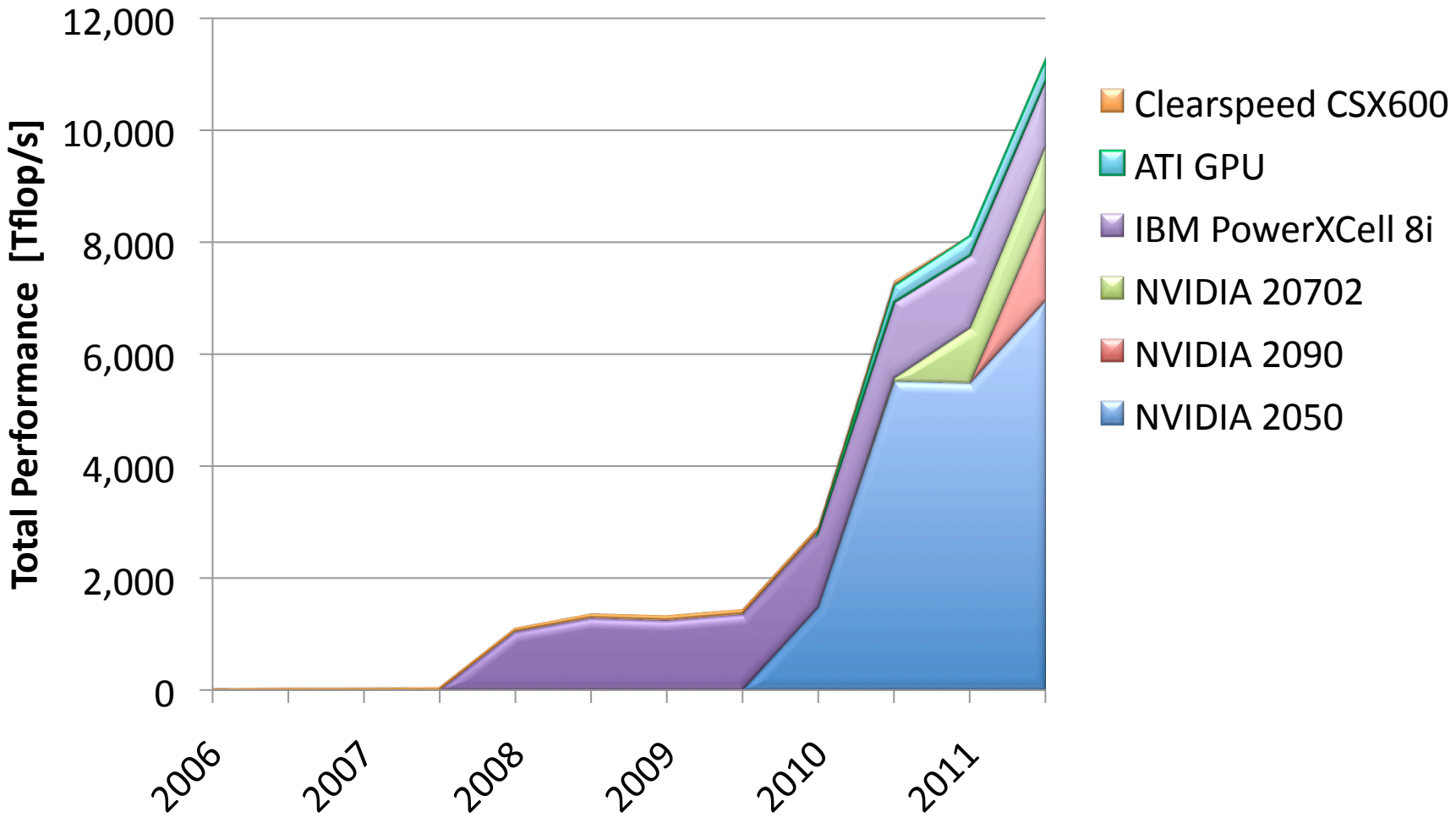
GPU capacity comparisons



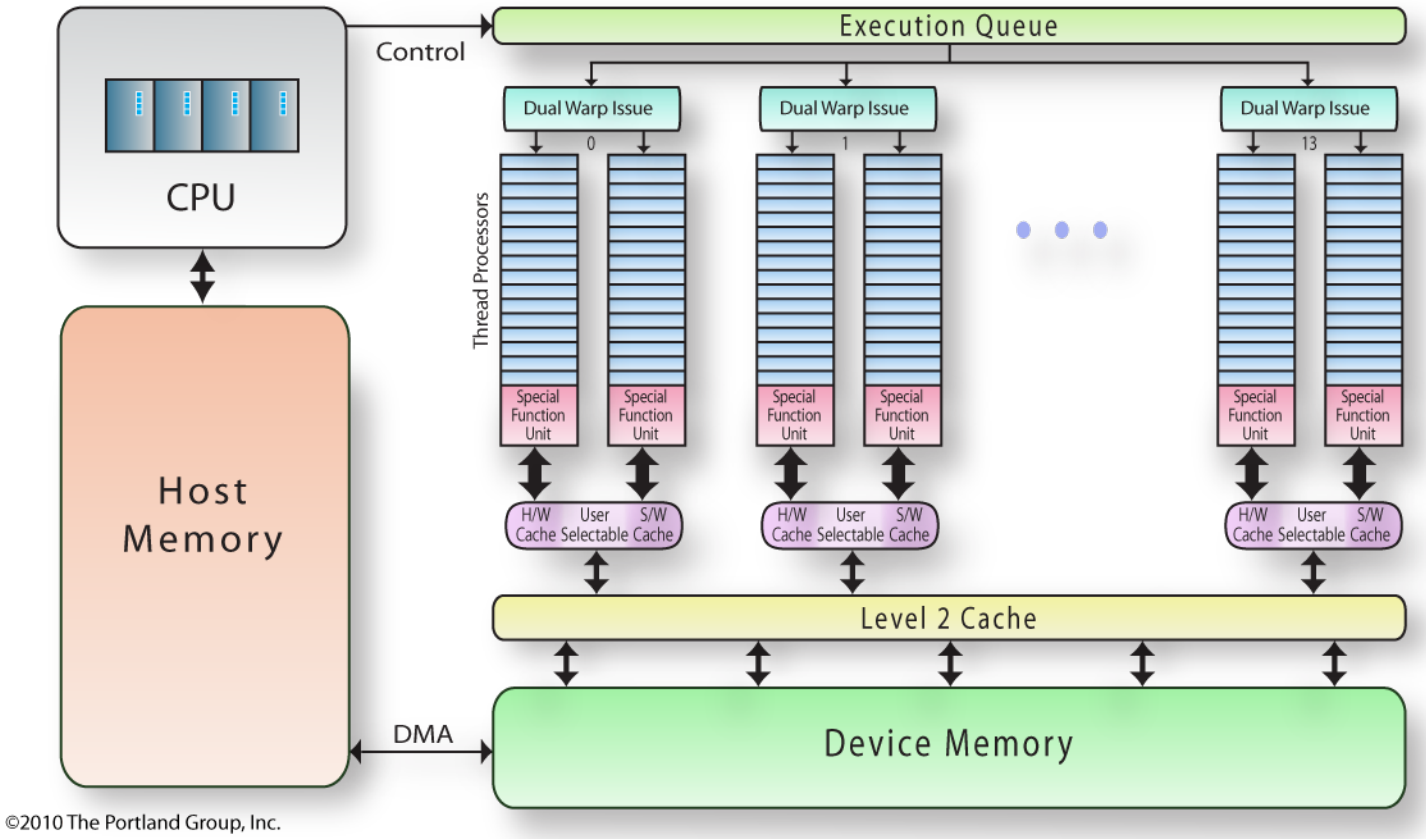




TOP500: Accelerators



Abstracted x64+ Nvidia Fermi Architecture



GPU Architecture Features

- Optimized for high degree of regular parallelism
- Classically optimized for low precision
  - Fermi supports double precision at ½ single precision bandwidth
- High bandwidth memory (Fermi supports ECC)
- Highly multithreaded (slack parallelism)
- Hardware thread scheduling
- Non-coherent software-managed data caches
  - Fermi has two-level hardware data cache
- No multiprocessor memory model guarantees
  - some guarantees with fence operations

Tesla-10 Features Summary

- Massively parallel thread processors
  - Organized into multiprocessors
    - up to 30, see deviceQuery or pgaccelinfo
  - Physically: 8 thread processors per multiprocessor
  - ISA: 32 threads per warp
  - Logically: Thread block is quasi-SIMD
- Memory hierarchy
  - host memory, device memory, constant memory, shared memory, register
- Queue of operations (kernels) on device

Fermi (Tesla-20) Features Summary

- Massively parallel thread processors
  - Organized into multiprocessors
    - up to 16, see deviceQuery or pgaccelinfo
  - Physically: two groups of 16 thread processors per multiprocessor
  - ISA: still 32 threads per warp, dual issue for 32-bit code
- Memory hierarchy
  - host memory, device memory (two level hardware cache), constant memory, (configurable) shared memory, register
- Queue of operations (kernels) on device
- ECC memory protection (supported, not default)
- Much improved double precision performance
- Hardware 32-bit integer multiply

Parallel Programming on CPUs

- Instruction level parallelism (ILP)
  - Loop unrolling, instruction scheduling
- Vector parallelism
  - Vectorized loops (or vector intrinsics)
- Thread level / Multiprocessor / multicore parallelism
  - Parallel loops, parallel tasks
  - Posix threads, OpenMP, Cilk, TBB, ...
- Large scale cluster / multicomputer parallelism
  - MPI (& HPF, co-array Fortran, UPC, Titanium, X10, Fortress, Chapel)

Parallel Programming on GPUs

- Instruction level parallelism (ILP)
  - Loop unrolling, instruction scheduling
- Vector parallelism
  - CUDA Thread Blocks, OpenCL Work Groups
- Thread level / Multiprocessor / multicore parallelism
  - CUDA Grid, OpenCL