Parallel Computer Architecture
- Basics -

Christian Terboven <terboven@rz.rwth-aachen.de>

29.07.2013 / Aachen, Germany

Stand: 22.07.2013

Version 2.3

Agenda

- Overview: HPC Systems
- Processor Microarchitecture
- Shared-Memory Parallel Systems
- Distributed-Memory Parallel Systems (Cluster)
- General Purpose Graphic Processing Units (GPGPUs)
- Intel Xeon Phi Coprocessor
- Summary and Conclusions

Overview: HPC Systems

Performance Metrics

- **FLOPS = Floating Point Operation per Second**
  - Megaflops = 10^6 FLOPS
  - Gigaflops = 10^9 FLOPS
  - Teraflops = 10^{12} FLOPS
  - Petaflops = 10^{15} FLOPS

- **Memory Bandwidth: Rate at which data can be read from or stored into a semiconductor memory by a processor.**

- **Memory Latency: Delay incurred when a processors accesses data inside the main memory (data not in the cache).**

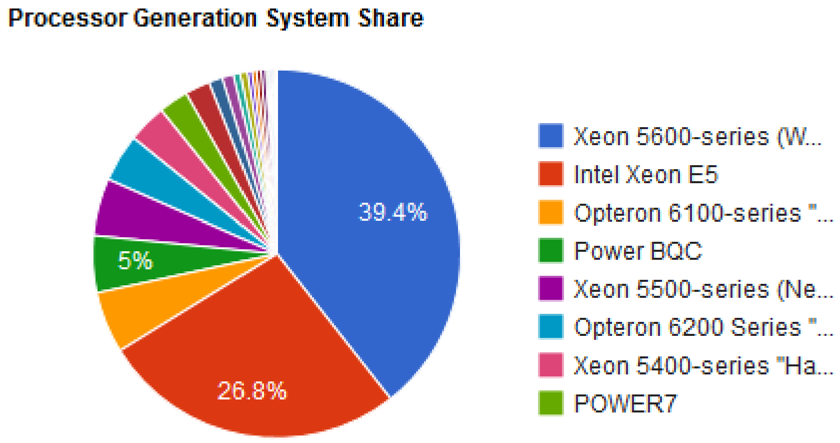
LINPACK Benchmark

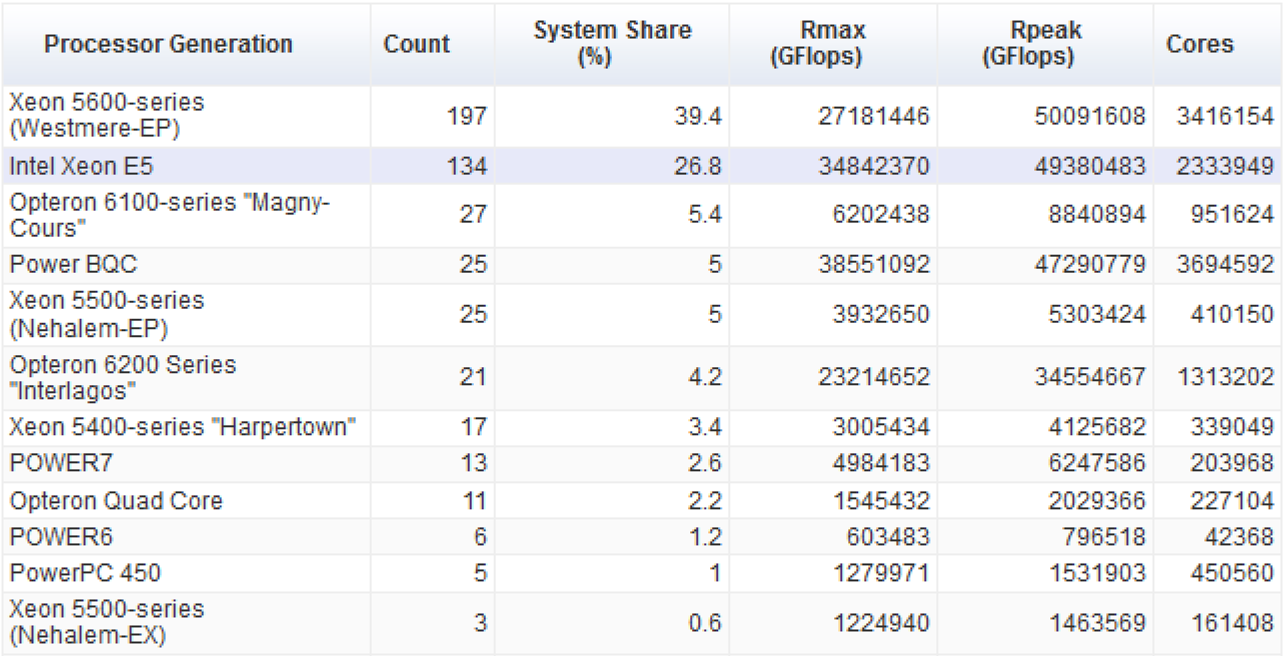
- **The theoretical *peak performance* is defined by the clock rate and cannot be achieved by a real application.**
- **The 500 fastest computer systems of the world are compared in the *Top500 list*:**
  - Updated in June at ISC (Germany) and November at SC (USA)
  - LINPACK benchmark (www.top500.org): Parallel solution

    of a dense (random) linear equation system, performance measured in

    FLOPS
  - Currently fastest system: *MilkyWay-2* (China), 22.86 PF, 17.8 MW power
    - Cluster of >32.000 cpus (Intel Xeon E5-2600) with >48.000 Intel Xeon Phi
- **Example: Our Bull Cluster with about 1500 nodes**
  - Peak: 292.135,94 GFLOPS, 25448 cores (3.0 GHz Intel Xeon)
  - Linpack: 219,8 TFLOPS (ranked 32 in 06/2011, ranked 111 in 11/2012)

Top500 List Statistics: Processor Family

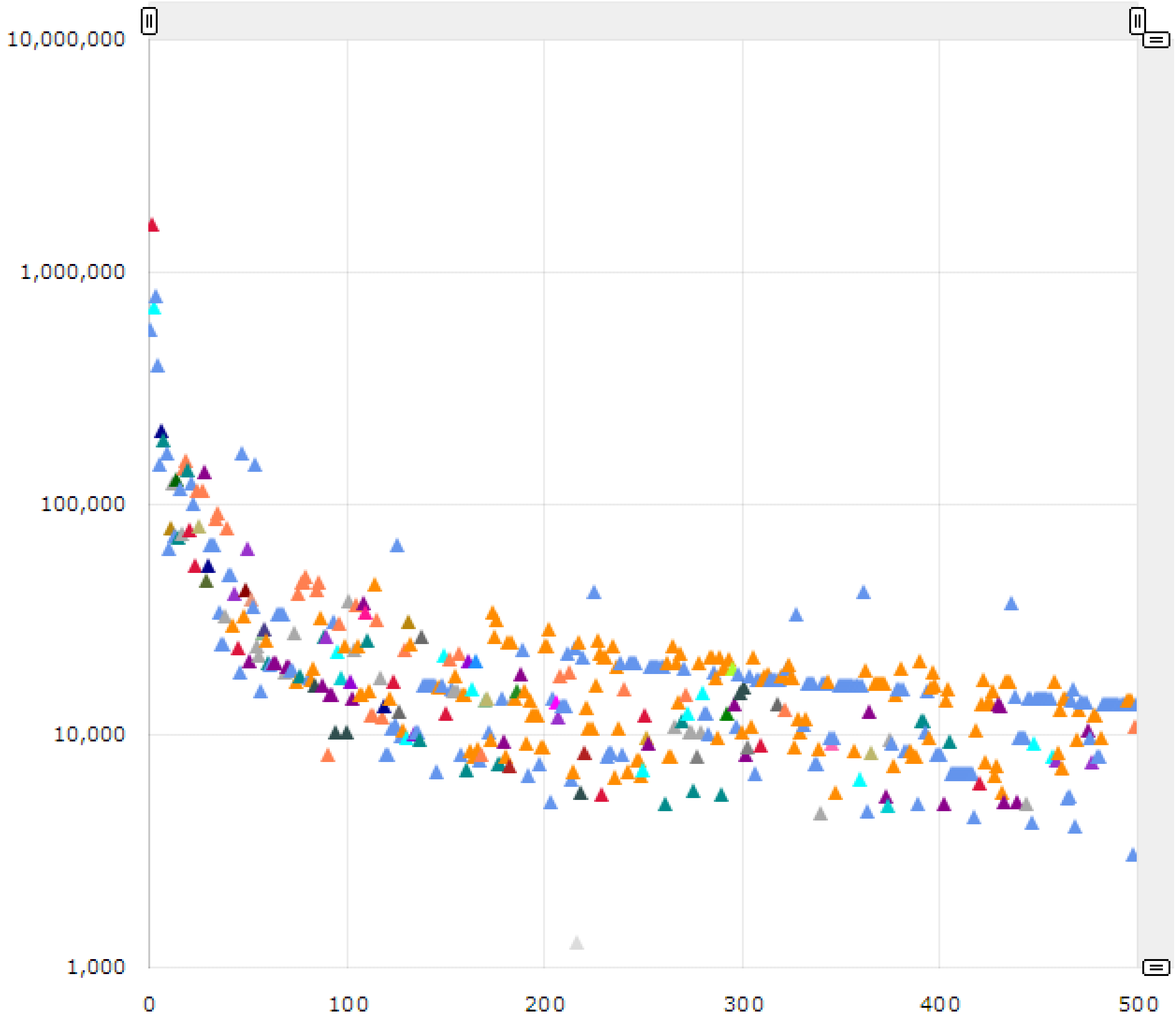




Processor Generation	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
Xeon 5600-series (Westmere-EP)	197	39.4	27181446	50091608	3416154
Intel Xeon E5	134	26.8	34842370	49380483	2333949
Opteron 6100-series "Magny-Cours"	27	5.4	6202438	8840894	951624
Power BQC	25	5	38551092	47290779	3694592
Xeon 5500-series (Nehalem-EP)	25	5	3932650	5303424	410150
Opteron 6200 Series "Interlagos"	21	4.2	23214652	34554667	1313202
Xeon 5400-series "Harpertown"	17	3.4	3005434	4125682	339049
POWER7	13	2.6	4984183	6247586	203968
Opteron Quad Core	11	2.2	1545432	2029366	227104
POWER6	6	1.2	603483	796518	42368
PowerPC 450	5	1	1279971	1531903	450560
Xeon 5500-series (Nehalem-EX)	3	0.6	1224940	1463569	161408

X86 has evolved as the main architecture for HPC systems.

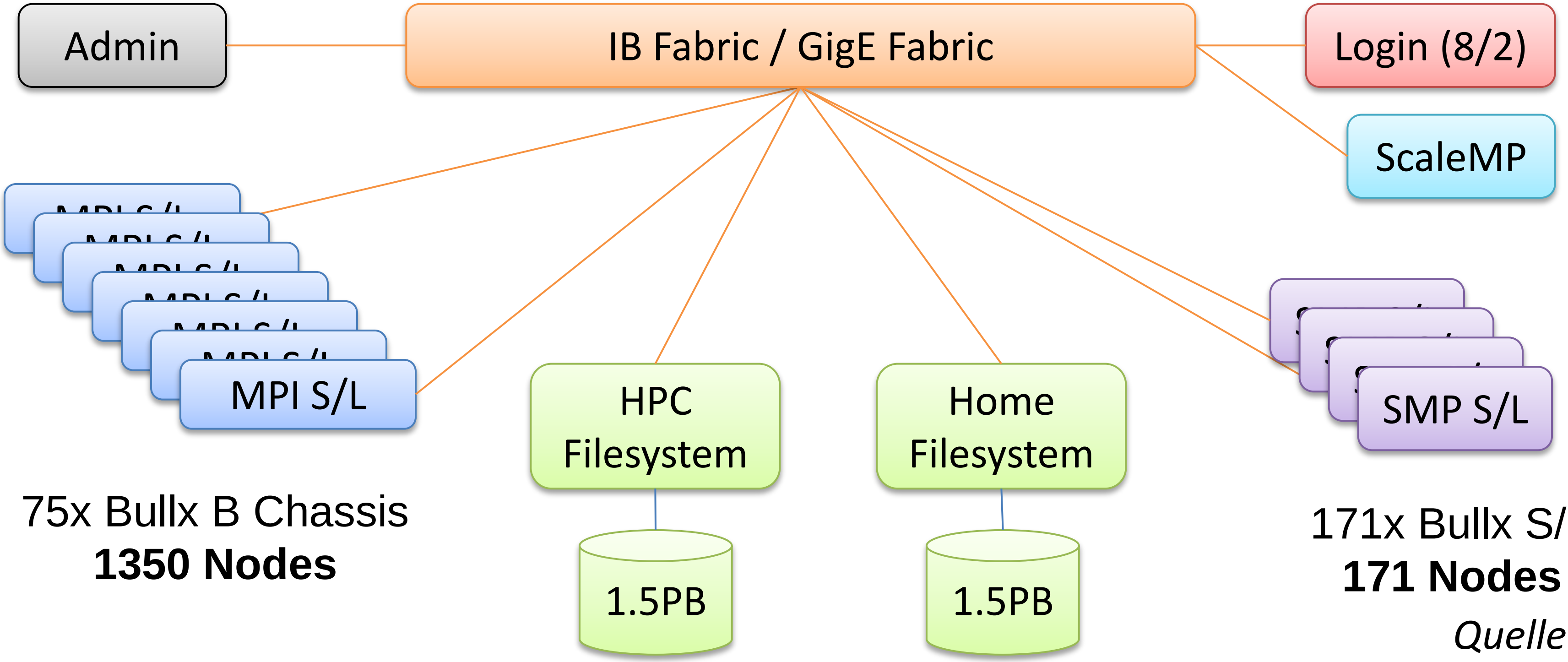
Top500 List Statistics: Number of Cores



The number of processor cores per system is exploding.

NEC/HP, IBM, Raytheon/Aspen Systems, NRCPCET, HP, Megware, RSC SKIF, Atipa, Itautec,HP/WIPRO, Adtech, Clustervision/Supermicro, Dell/Sun/IBM, IPE, Nvidia, Tyan, Dell, SGI, Appro,Cray Inc., Xenon Systems, Bull, NUDT, Acer Group, Lenovo, Intel, NEC, Self-made, Fujitsu,Oracle, Inspur, Dawning, Supermicro, Hitachi, Eurotech, ManyCoreSoft, T-Platforms, RSC Group

Our HPC Cluster from Bull: Overview



75x Bullx B Chassis
1350 Nodes

171x Bullx S/8
171 Nodes

Quelle: Bull

Group	# Nodes	Sum TFLOP	Sum Memory	# Procs per Node	Memory per Node
MPI-Small	1098	161	26 TB	2 x Westmere-EP (3,06 GHz, 6 Cores, 12 Threads)	24 GB
MPI-Large	252	37	24 TB		96 GB
SMP-Small	135	69	17 TB	4x Nehalem-EX (2,0 GHz, 8 Cores, 16 Threads) BCS: 4, vSMP: 16	128 GB
SMP-Large	36	18	18 TB		512 GB
ScaleMP-vSMP	8 gekoppelt	4	4 TB		4 TB gekoppelt

Processor Microarchitecture

Properties of modern Microarchitectures

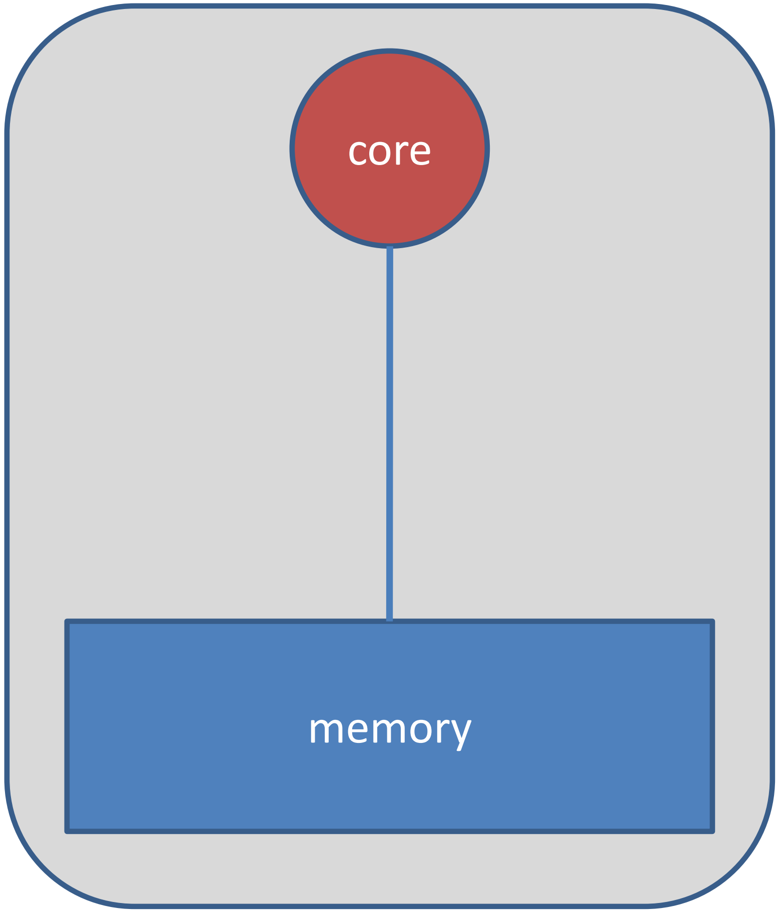
- **The program code (instructions) of the high level language (i.e. C/C++, Fortran) is translated into machine code by a compiler.**

- **The instructions are fetched from the main memory, decoded and executed in multiple steps (Pipelining). Conflicts are detected automatically and might lead to stall cycles.**

- **Modern (superscalar) processors are capable of executing multiple instructions in parallel (ILP = Instruction Level Parallelism).**
  - CPI = Clocks per Instruction, usually 0.5 to 2.0
  - Loops have the highest potential for efficient execution

Single Processor System (dying out) (1/2)

- **Processor**
  - Fetch program from memory
  - Execute program instructions
  - Load data from memory
  - Process data
  - Write results back to memory
- **Main Memory**
  - Store program
  - Store data
- **Input / Output is not covered here!**



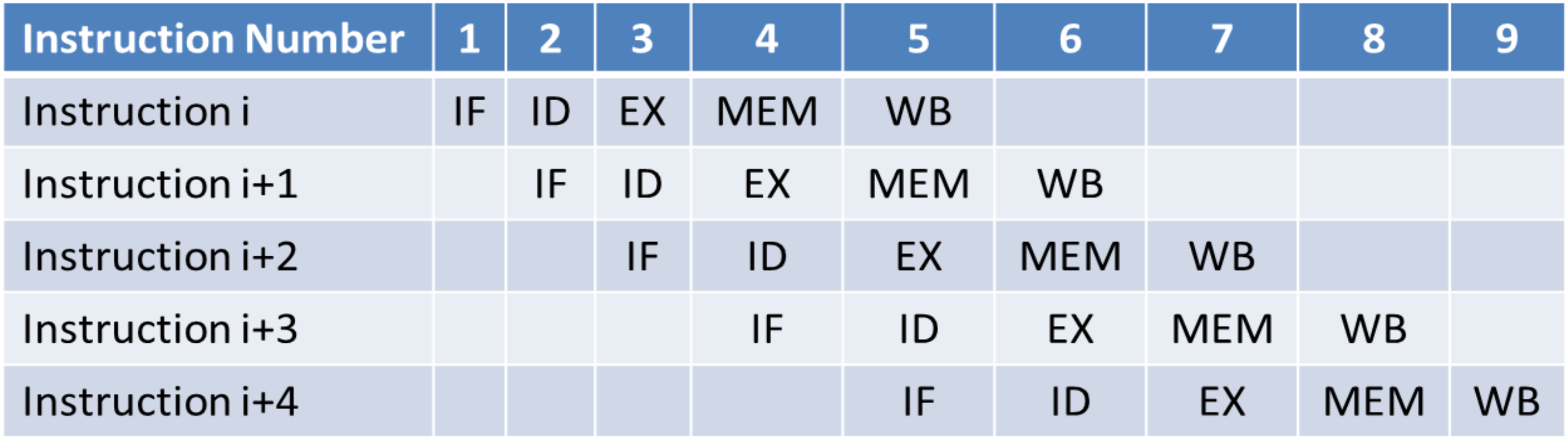
Pipelining (1/2)

- ***Pipelining*: An implementation technique whereby multiple instructions are overlapped in execution (think of an assembly line for automobiles).**
  - Throughput: Number of instructions per time interval
  - Speedup of pipelining: $\frac{\text{Time per instructions on unpipelined machine}}{\text{Number of pipeline stages}}$

- **Example: Assume any (RISC) instruction can be implemented in at most 5 clock cycles:**
  - Instruction fetch cycle (IF)
  - Instruction decode / register fetch cycle (ID)
  - Execution / effective address cycle (EX)
  - Memory access (MEM)
  - Write-back cycle (WB)

Pipelining (2/2)

- **Pipeline model of example architecture: On each clock cycle, another instruction is fetched and begins its 5-cycle exec.:**



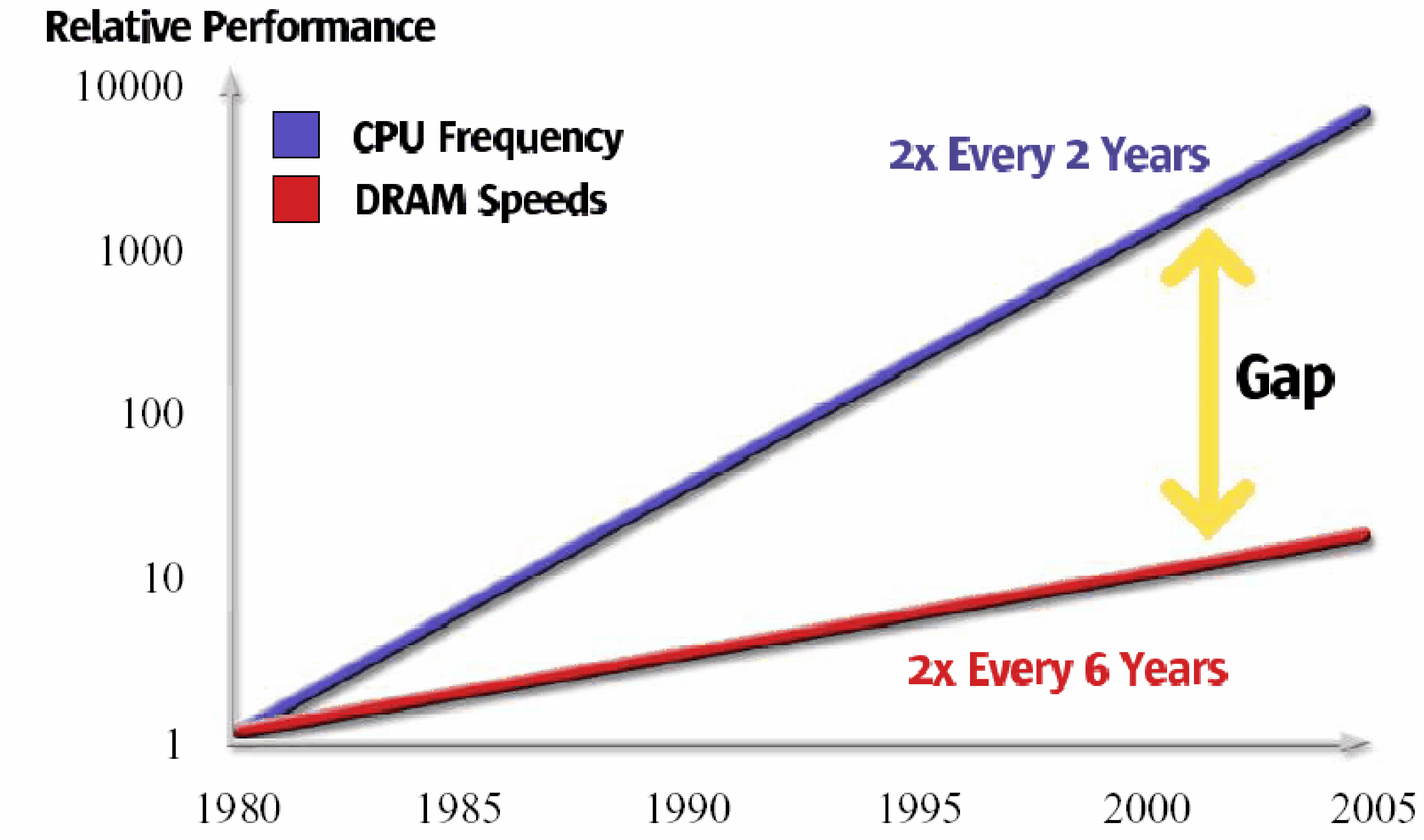
Instruction Number	1	2	3	4	5	6	7	8	9
Instruction i	IF	ID	EX	MEM	WB				
Instruction i+1		IF	ID	EX	MEM	WB			
Instruction i+2			IF	ID	EX	MEM	WB		
Instruction i+3				IF	ID	EX	MEM	WB	
Instruction i+4					IF	ID	EX	MEM	WB

- **The major problems of pipelines:**
  - *Structural hazards*: Resource conflicts when hardware cannot support all possible combinations of overlapped instructions
  - *Data hazards*: Instruction depends on result of previous instr.
  - *Control hazards*: Branches or other interrupting events

→ Any hazard leads to a pipeline stall.

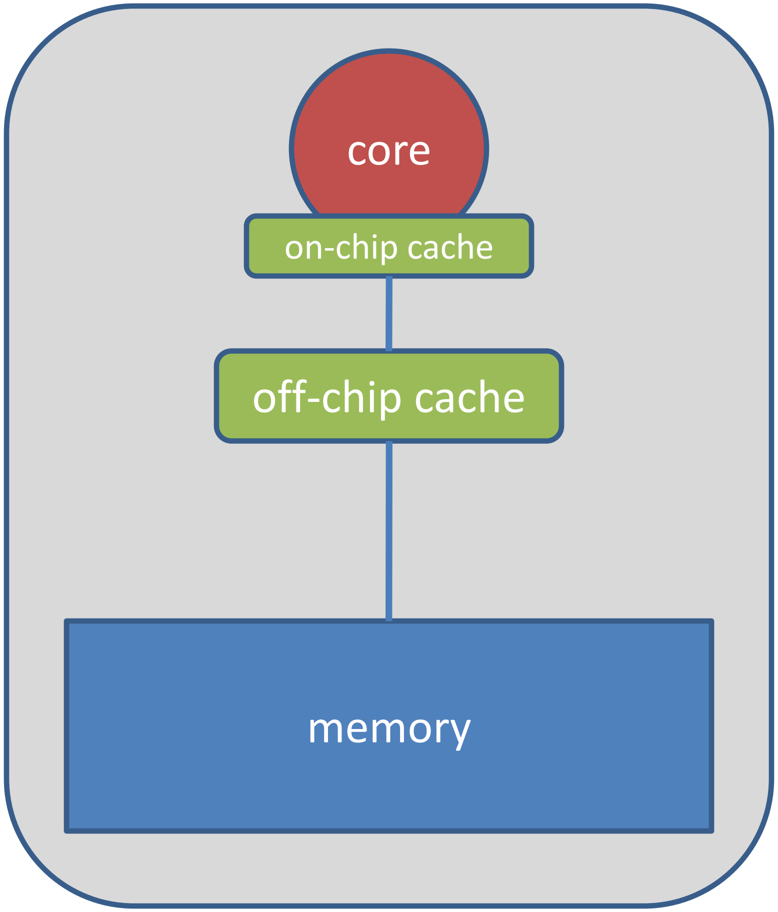
Memory Bottleneck

- There is a gap between core and memory performance.



Single Processor System (dying out) (2/2)

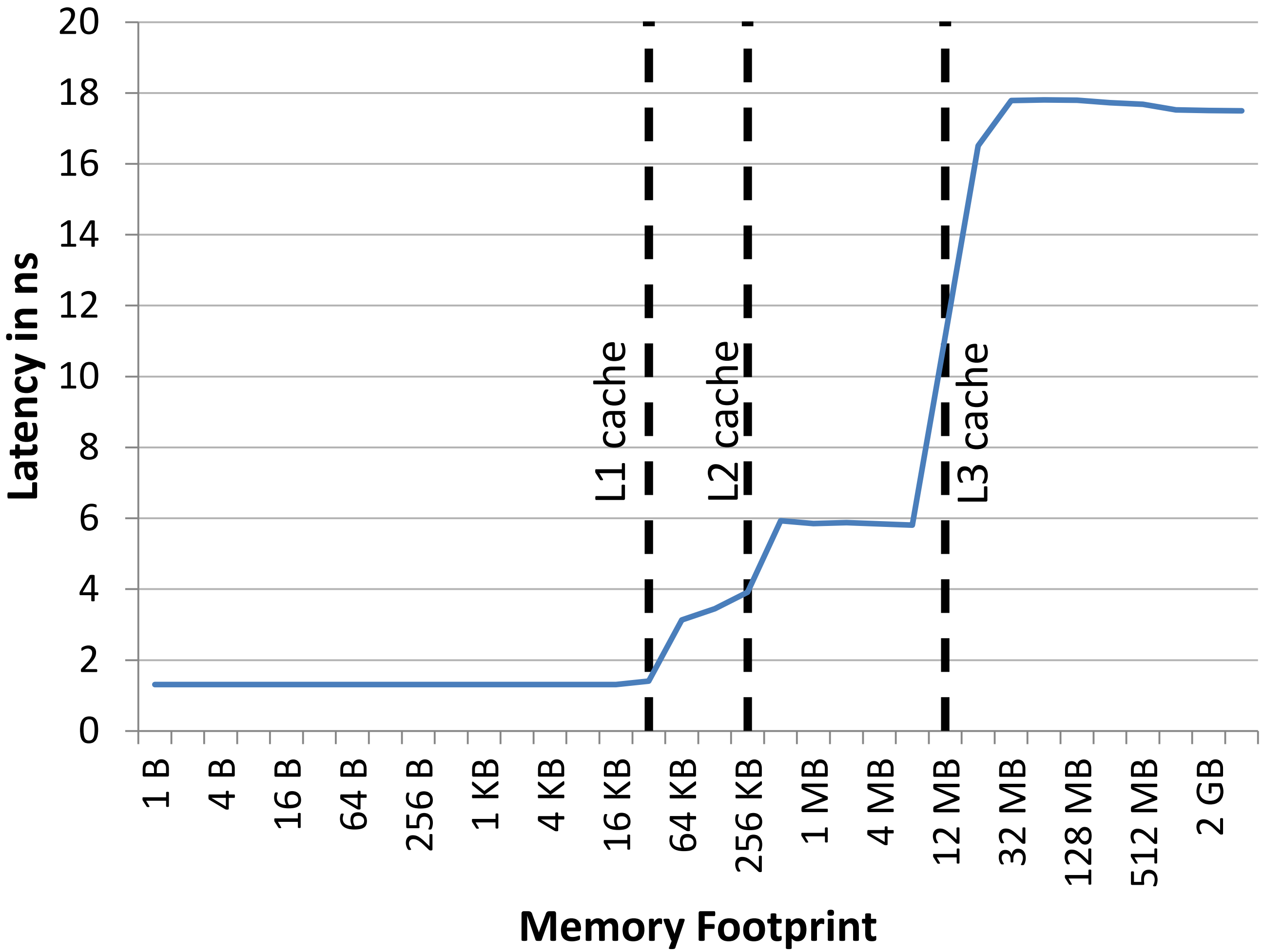
- CPU is fast
  - Order of 3.0 GHz
- Caches:
  - Fast, but expensive
  - Thus small, order of MB
- Memory is slow
  - Order of 0.3 GHz
  - Large, order of GB



- A good utilization of caches is crucial for good performance of HPC applications!

Visualization of the Memory Hierarchy

Latency on our Intel Westmere-EP systems



Memory Model: C/C++ vs. Fortran

- **The order of multi-dimensional arrays (= matrices!) in C/C++ is different from the order in Fortran:**
  - C: int a[2][3]

a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]	…	**physical memory**

  - Fortran: INTEGER, DIMENSION(2, 3) :: A

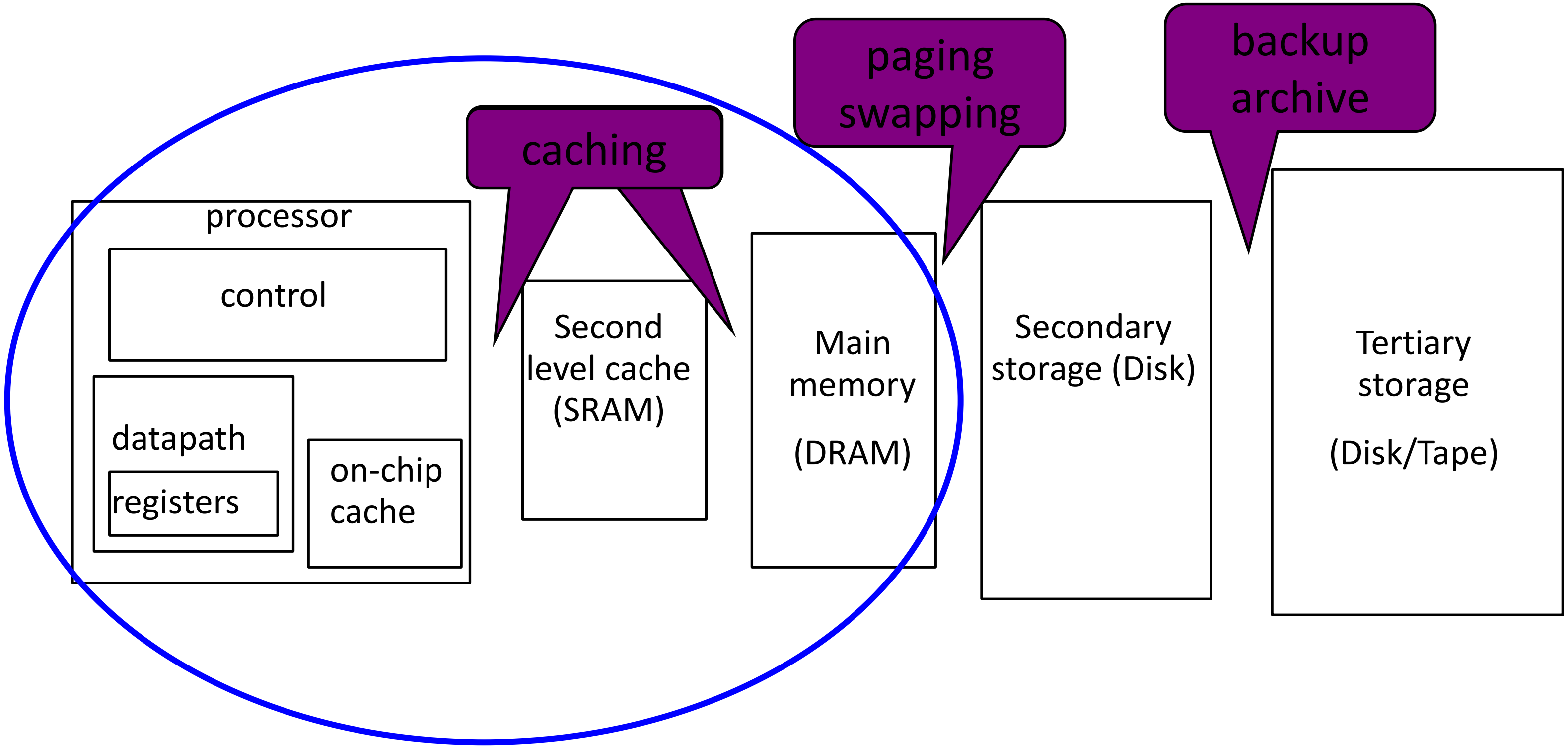
a(1,1)	a(2,1)	a(1,2)	a(2,2)	a(1,3)	a(2,3)	…	**physical memory**

- **Thus, the following is equivalent:**
  - C: int i[4][3][2]
  - Fortran: INTEGER, DIMENSION(2, 3, 4) :: I

- **C: Increment in the *rightmost* loop index for next element in cache**
- **Fortran: Incr. in the *leftmost* loop index for next element in cache**

Memory Hierarchy

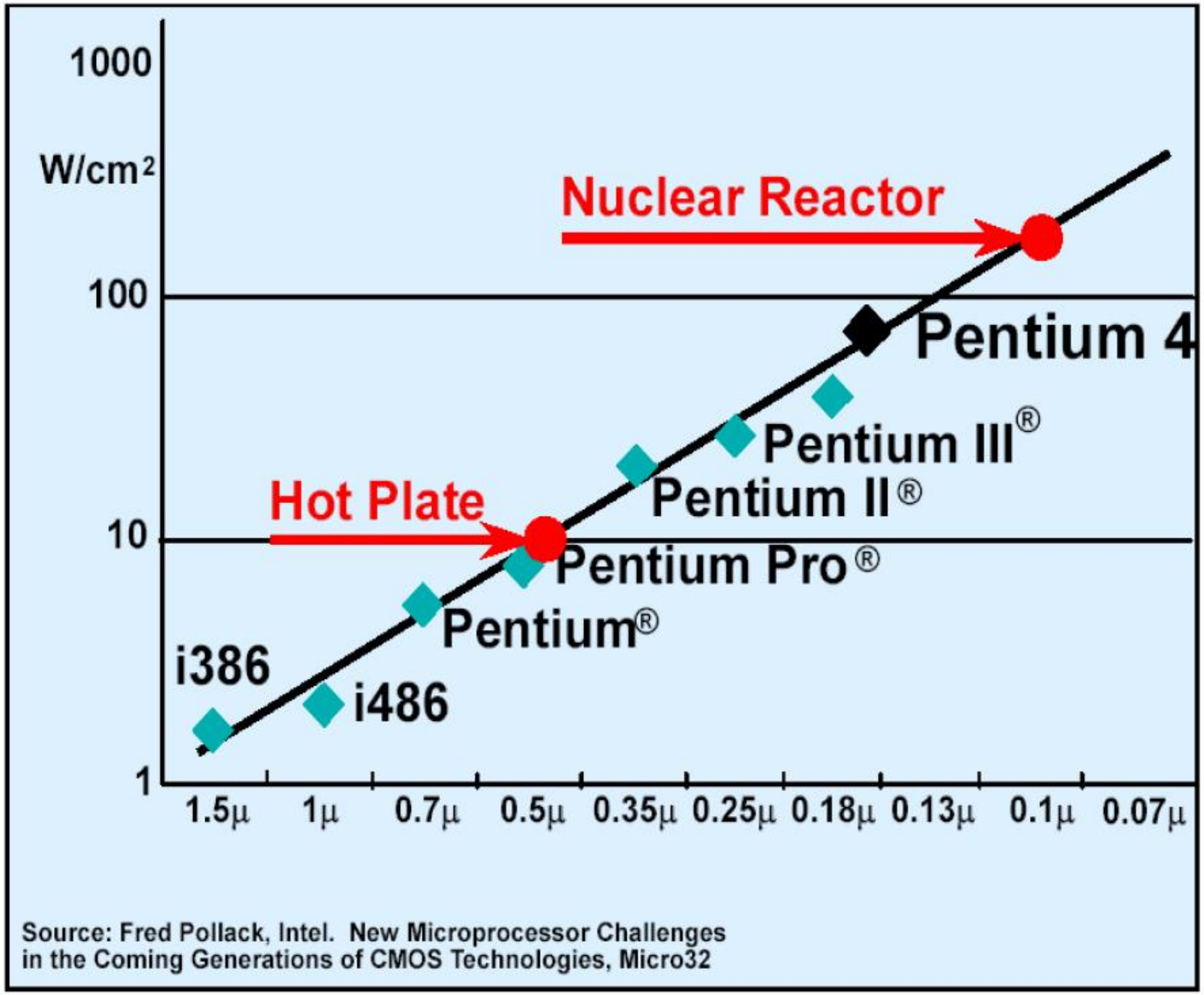
- **Since a large and fast memory is not feasible, the memory hierarchy has evolved and is getting deeper and deeper ...**



Latency Dimension:	nsec	10 nsec	100 nsec	10 msec	10 sec
Size:	~32 KB	1-8 MB	1-100 GB	Tera-/Petabytes	Peta-/Exabytes

Why can‘t you buy a 4.0 GHz x86 CPU?

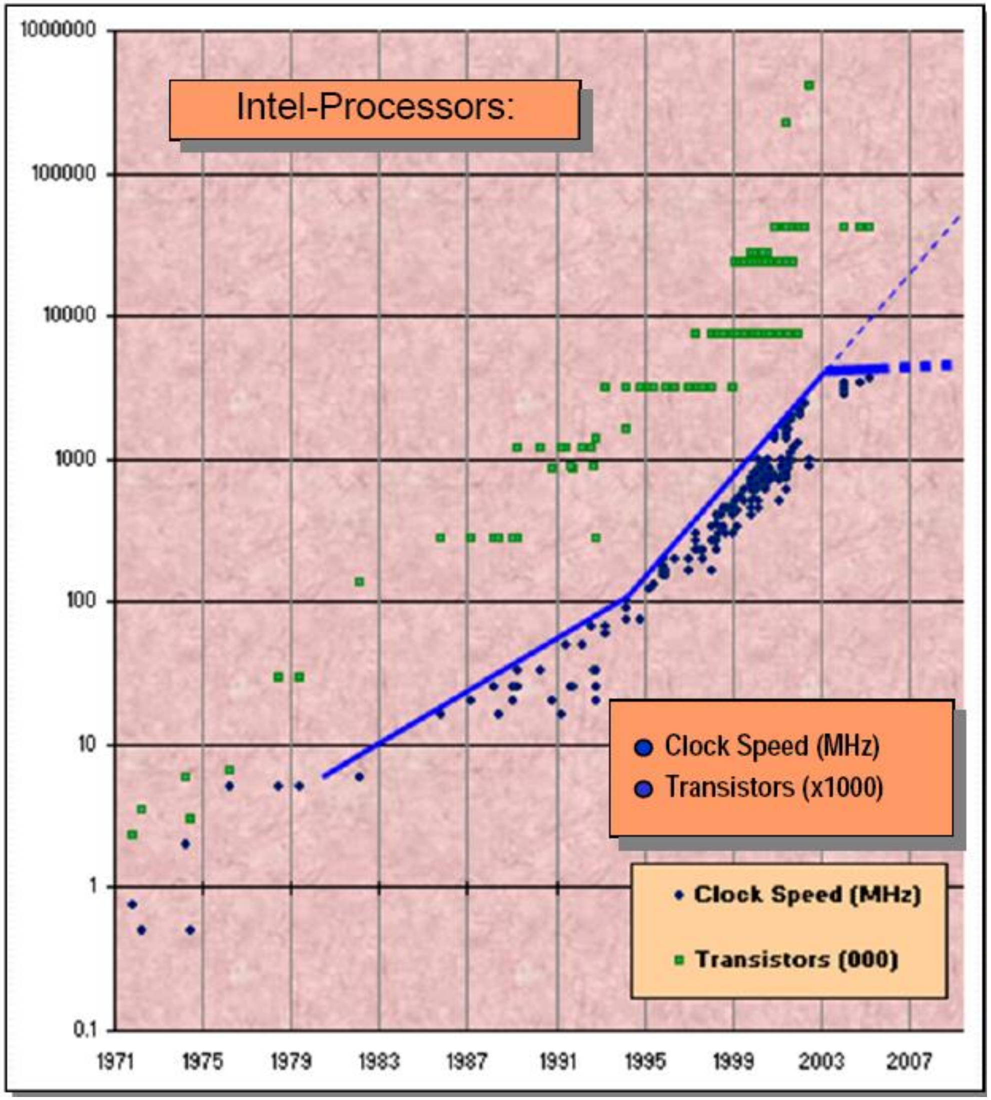
- Because that beast would get too hot!



Source: Fred Pollack, Intel. New Microprocessor Challenges in the Coming Generations of CMOS Technologies, Micro32

Fast clock cycles make processor chips more expensive, hotter and more power consuming.

Moore‘s Law still holds!



The number of transistors on a chip is still doubling every 24 months …

… but the clock speed is no longer increasing that fast!

Instead, we will see many more cores per chip!

Source: Herb Sutter

www.gotw.ca/publications/concurrency-ddj.htm

Chip Multi-Threading (CMT)

- **Traditional single-core processors can only process one thread at a time, spending a majority of time waiting for data from memory**

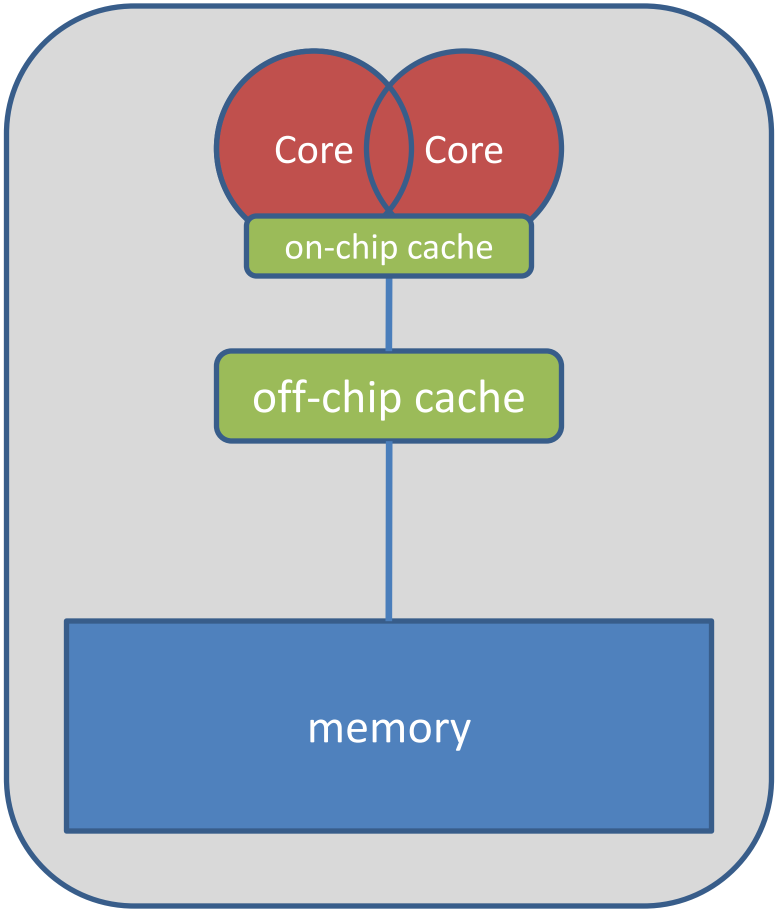
- **CMT refers to a processor‘s ability to process multiple software threads. Such capabilities can be implemented using a variety of methods, such as**
  - Having multiple cores on a single chip:

    Chip Multi-Processing (CMP)
  - Executing multiple threads on a single core:

    Simultaneous Multi-Threading (SMT)
  - A combination of both CMP and SMT.

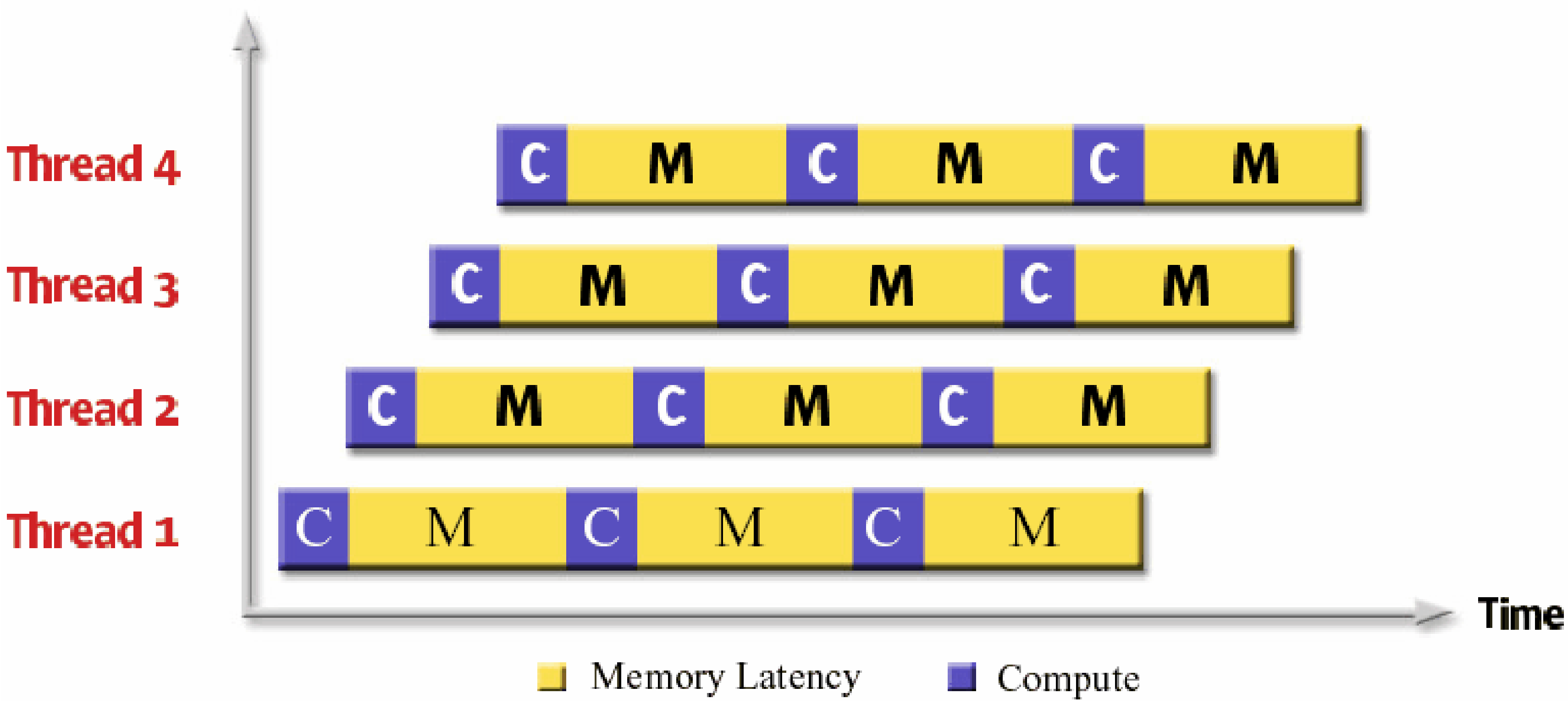
Dual-Core Processor System

- **Since 2005/2006 Intel and AMD are producing dual-core processors for the mass market!**
- **In 2006/2007 Intel and AMD introduced quad-core processors.**
- **→ Any recently bought PC or laptop is a multi-core system already!**



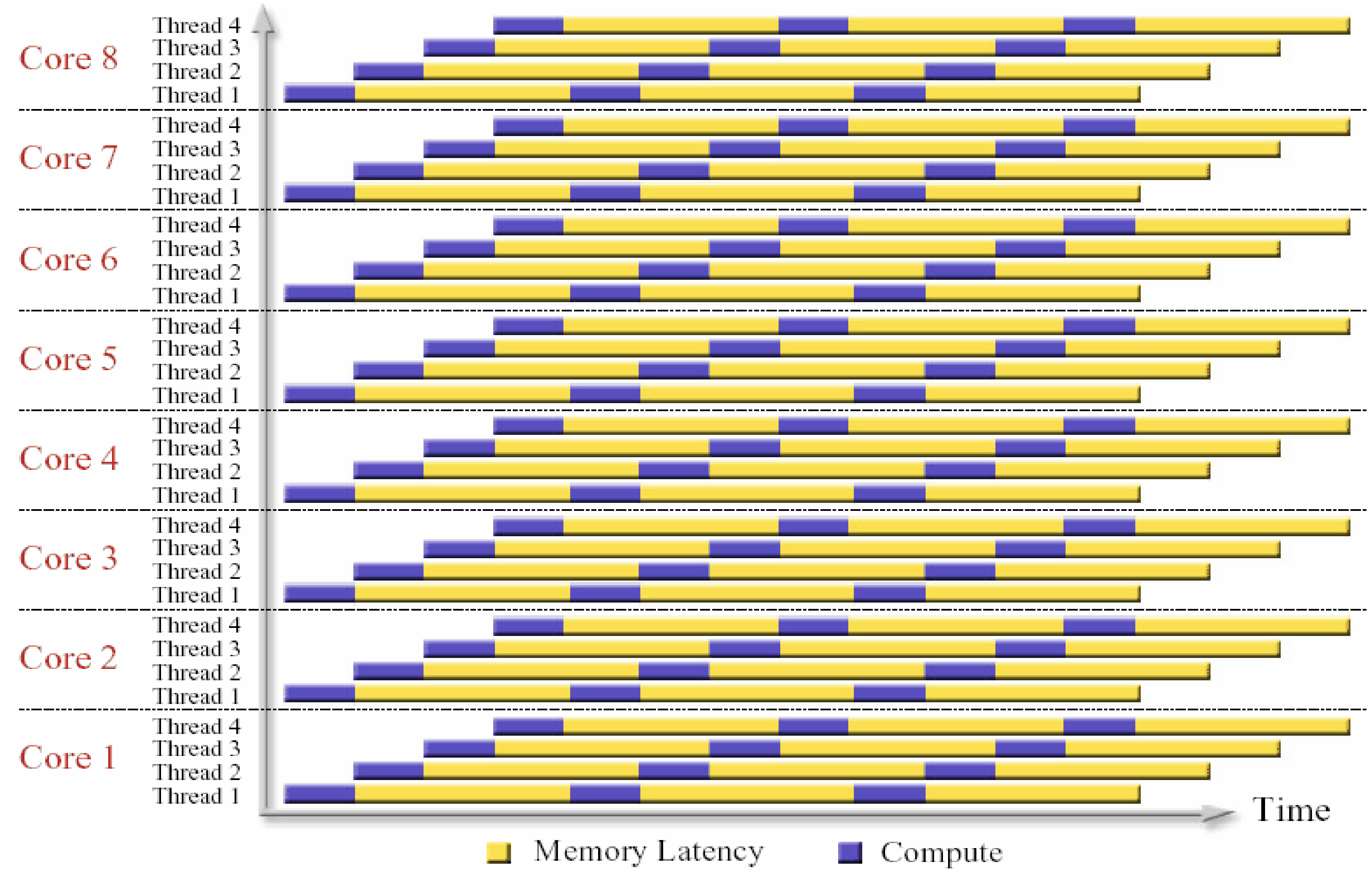
Simultaneous Multi-Threading (SMT)

- **Each Core executes multiple threads simultaneously**
  - Typically there is one register set thread per thread
  - But compute units are shared



Today: Multiple Multi-Threaded Cores

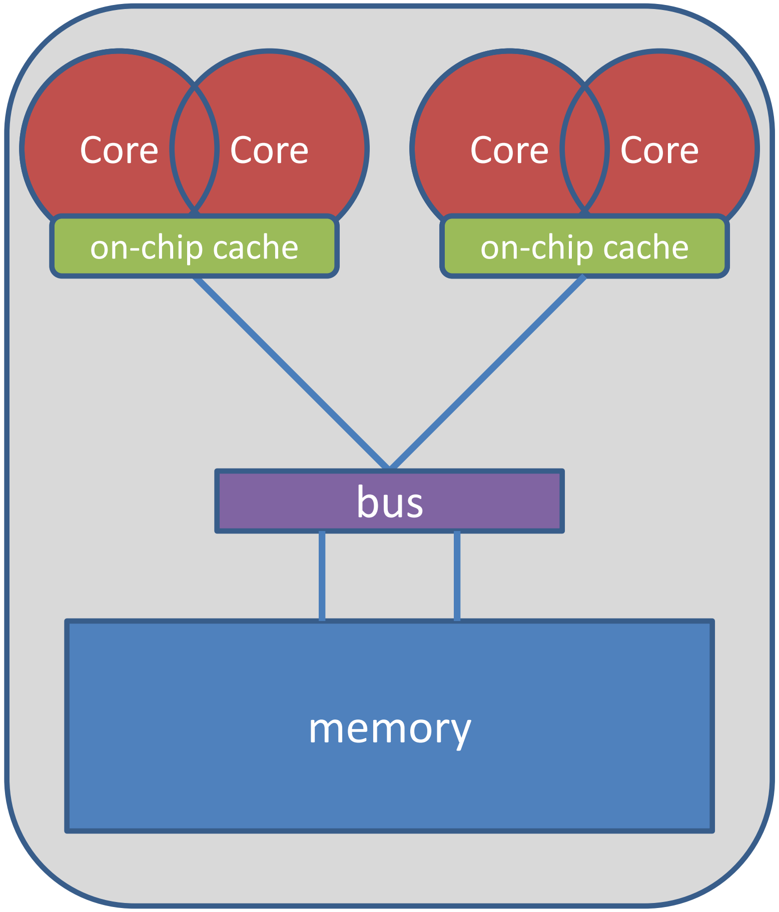
- Combination of CMP and SMT at work:



Shared-Memory Parallelism

Example for a SMP system

- **Dual-socket Intel Woodcrest (dual-core) system**
  - Two cores per chip, 3.0 GHz
  - Each chip has 4 MB of L2 cache on-chip, shared by both cores
  - No off-chip cache
  - Bus: Frontsidebus
- **SMP: Symmetric Multi Processor**
  - Memory access time is uniform on all cores
  - Limited scalabilty



Cache Coherence (cc)

- **If there are multiple caches not shared by all cores in the system, the system takes care of the cache coherence.**
- **Example:**

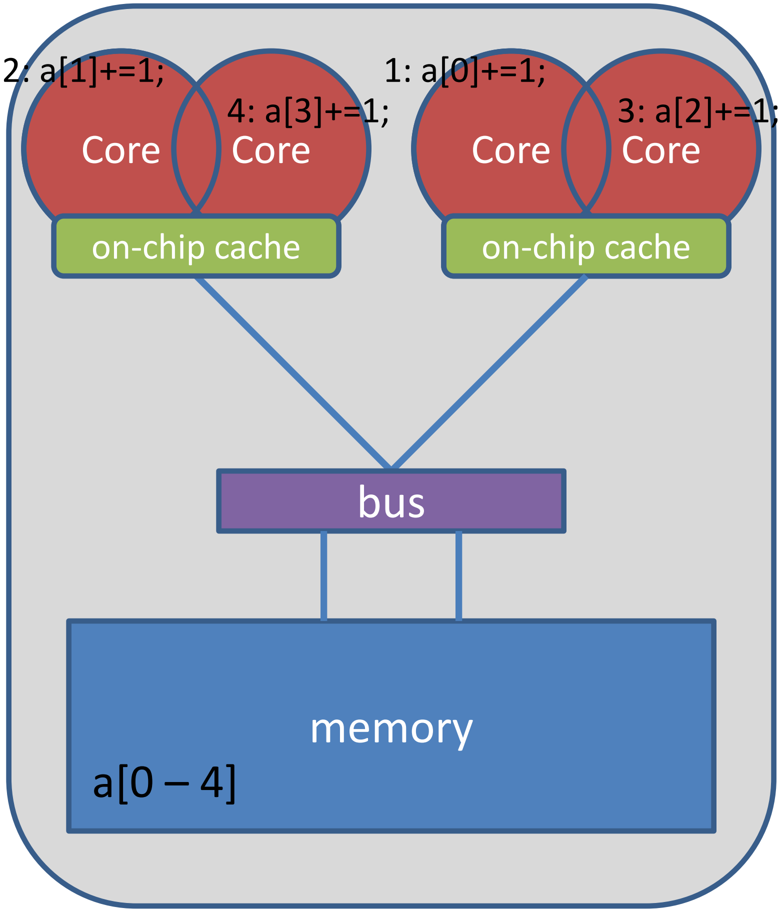
```
int a[some_number]; //shared by all threads
thread 1: a[0] = 23;     thread 2: a[1] = 42;
--- thread + memory synchronization (barrier) ---
thread 1: x = a[1];      thread 2: y = a[0];
```

  - Elements of array `a` are stored in continuous memory range
  - Data is loaded into cache in 64 byte blocks (cache line)
  - Both `a[0]` and `a[1]` are stored in caches of thread 1 and 2
  - After synchronization point all threads need to have the same view of (shared) main memory

- **False Sharing: Parallel accesses to the same cache line may have a significant performance impact!**

False Sharing

- **False Sharing: Parallel accesses to the same cache line may have a significant performance impact!**



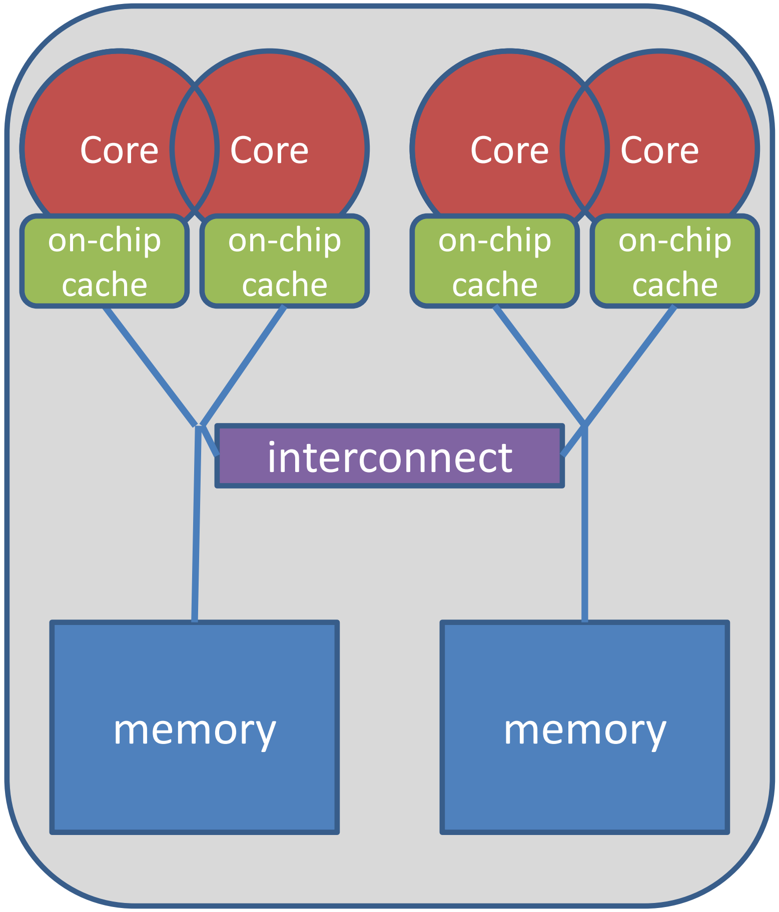
Caches are organized in lines of typically 64 bytes: integer array a[0-4] fits into one cache line.

Whenever one element of a cache line is updated, the whole cache line is invalidated.

Local copies of a cache line have to be re-loaded from the main memory and the computation may have to be repeated.

Example for a cc-NUMA system

- **Dual-socket AMD Opteron (dual-core) system**
  - Two cores per chip, 2.4 GHz
  - Each core has separate 1 MB of L2 cache on-chip
  - No off-chip cache
  - Interconnect: HyperTransport

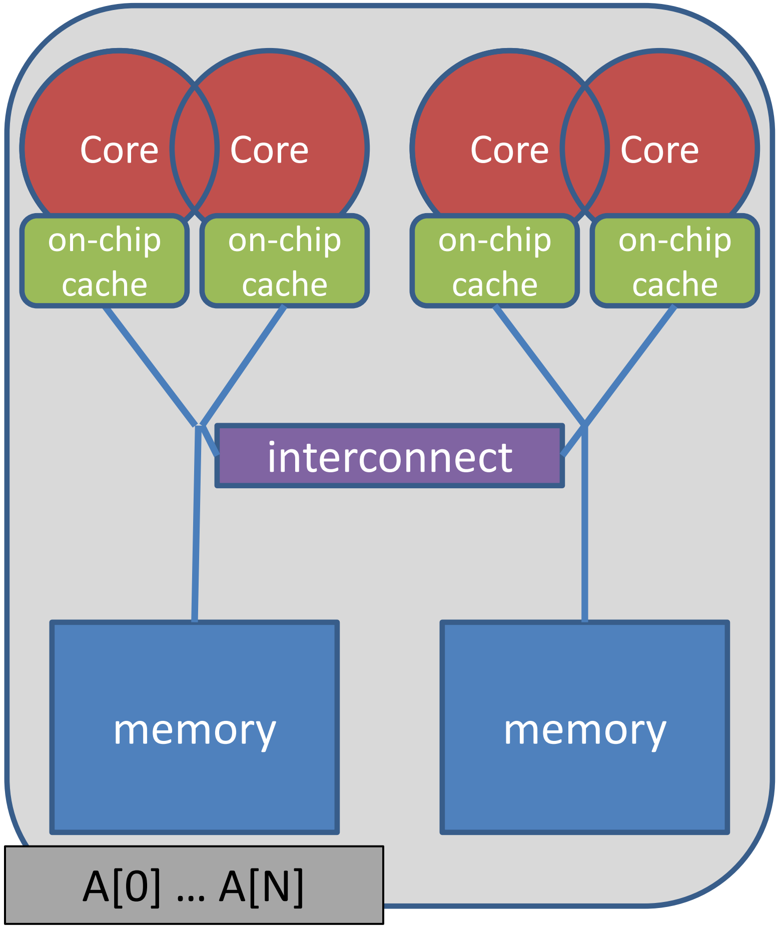


- **cc-NUMA:**
  - Memory access time is non-uniform
  - Scalable (only if you do it right, as we will see)

Non-uniform Memory

- **Serial code: all array elements are allocated in the memory of the NUMA node containing the core executing this thread**

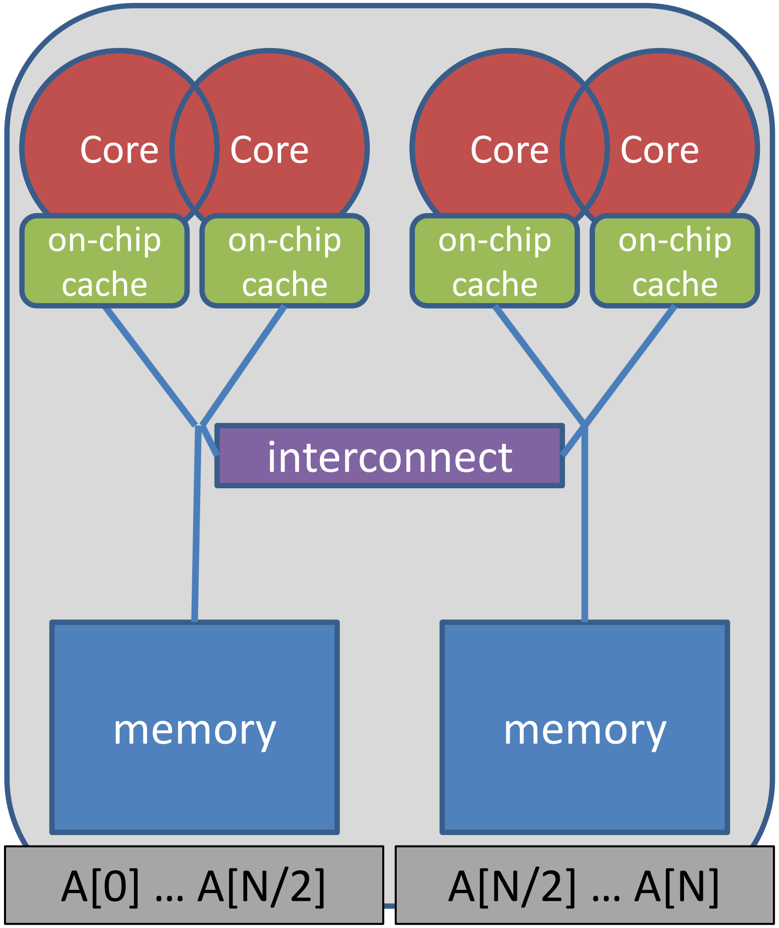
```
double* A;
A = (double*)
    malloc(N * sizeof(double));



for (int i = 0; i < N; i++) {
   A[i] = 0.0;
}
```



First Touch Memory Placement

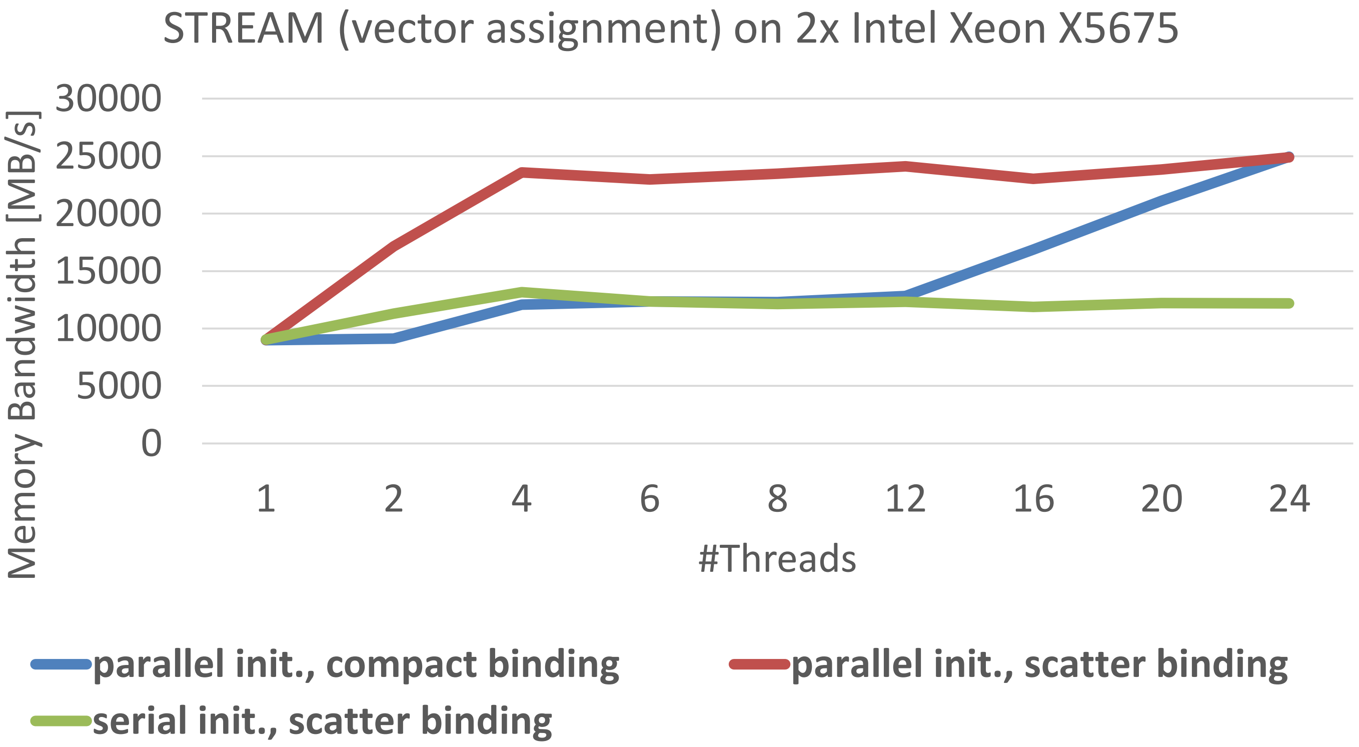
- **First Touch w/ parallel code: all array elements are allocated in the memory of the NUMA node containing the core executing the thread initializing the respective partition**

```
double* A;
A = (double*)
    malloc(N * sizeof(double));

omp_set_num_threads(2);

#pragma omp parallel for
for (int i = 0; i < N; i++) {
   A[i] = 0.0;
}
```

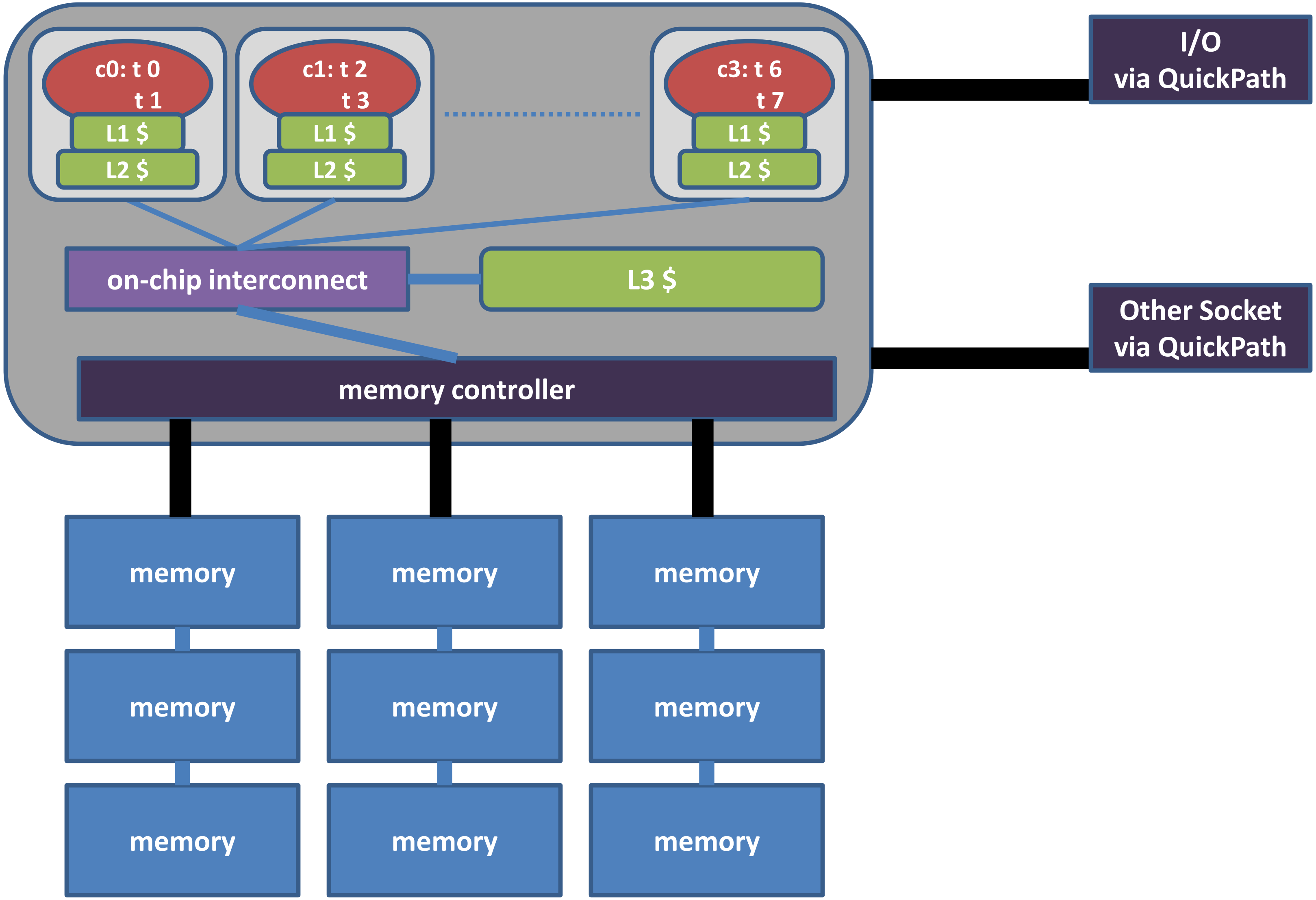


Serial vs. Parallel Initialization

- **Performance of OpenMP-parallel STREAM vector assignment measured on 2-socket Intel® Xeon® X5675 („Westmere“) using Intel® Composer XE 2013 compiler with different thread binding options:**

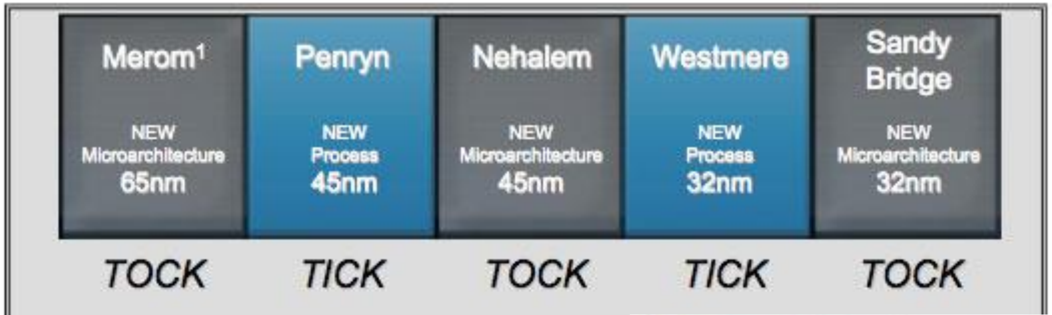


Intel‘s Nehalem (1/3)

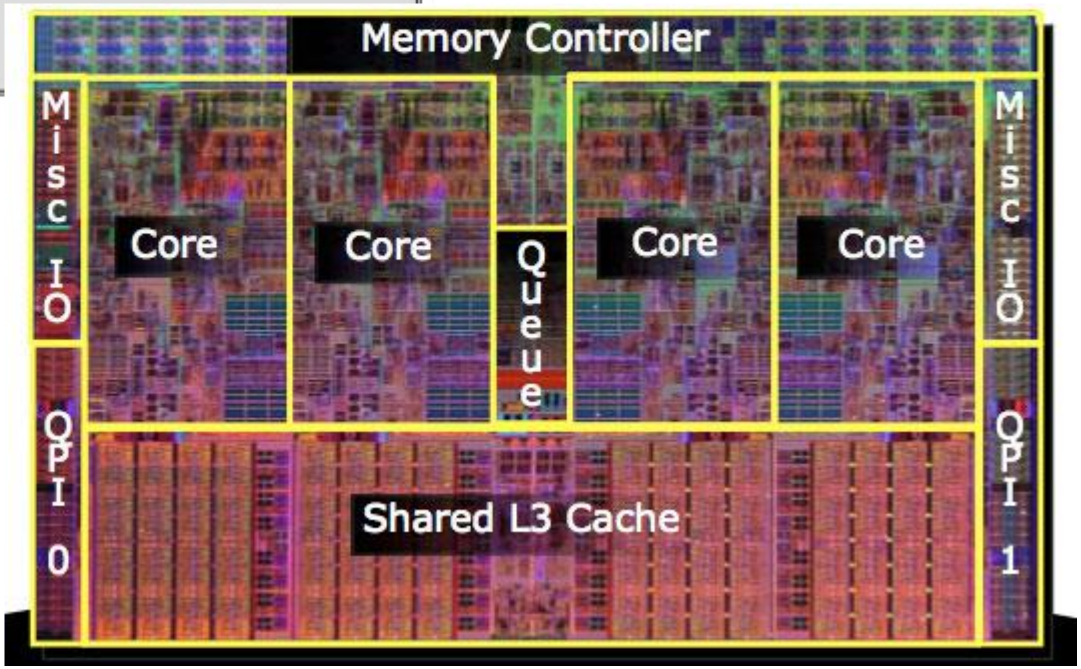


Intel‘s Nehalem (2/3)

- With each improvement in production technology the number of cores per chip increases.



- Minor modification to cache hierarchy: L3 cache is shared by all cores, L1 and L2 caches are per core.



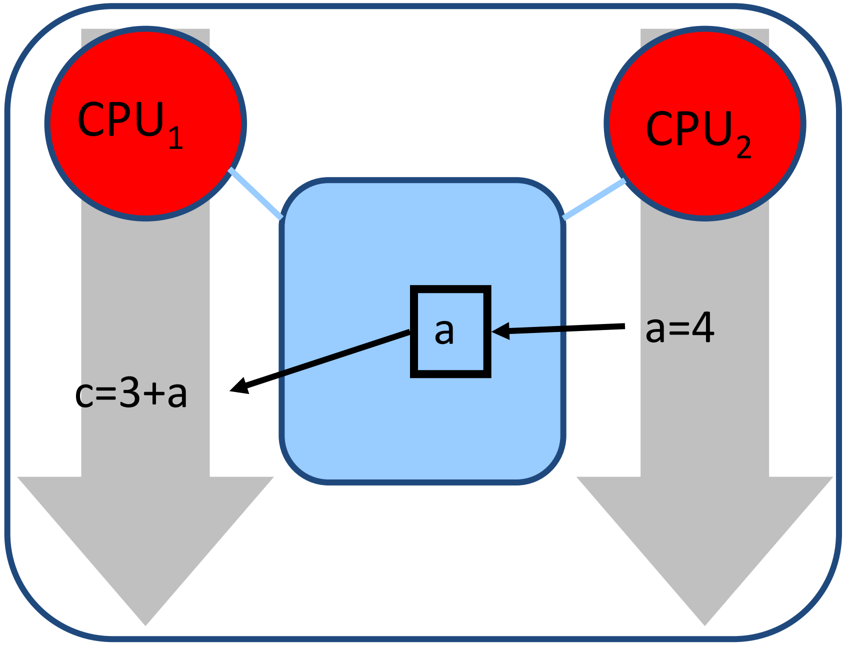
Intel‘s Nehalem (3/3)

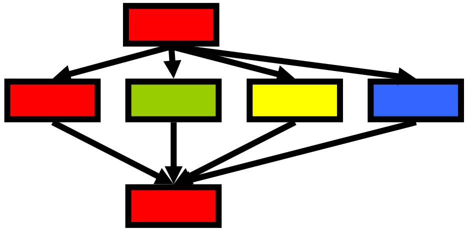
Technology

- 45 nm manufacturing process
- Integrated memory controller
  - Intel QuickPath Interconnect replaces FrontsideBus
  - Will offer cc-NUMA characteristics (see below)
- Simultaneous Multi-Threading (SMT) = Hyper-Threading
- Cache Hierarchy
  - 32 KB L1 instruction cache + 32 KB L1 data cache per core
  - 256 KB L2 cache per core
  - 2 or 3 MB L3 cache per core, but shared by all cores
- Number of pipeline stages: Core microarchitecture has only 12 stages (compared to 30 in latest Netburst architecture)

Shared Memory Parallelization

- Memory can be accessed by several threads running on different cores in a multi-socket multi-core system:



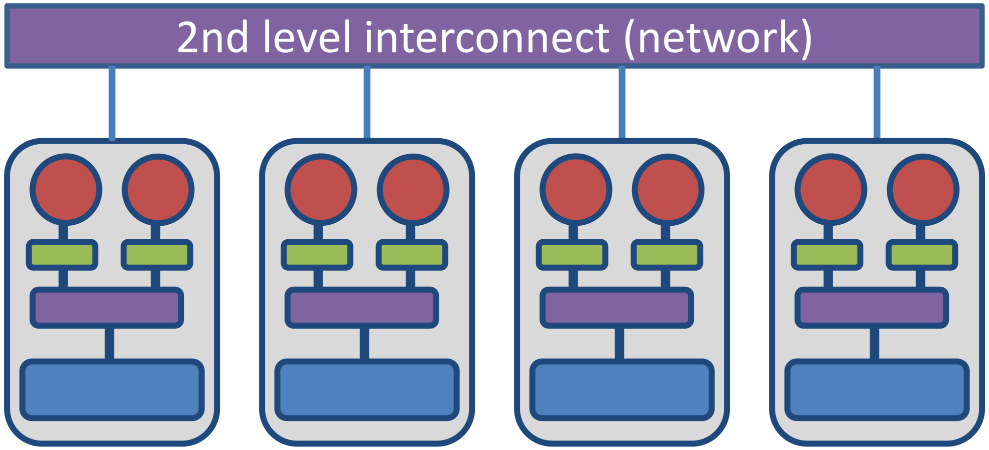


Look for tasks that can be executed simultaneously (task parallelism)

Distributed-Memory Parallelelism

Example for a Cluster of SMP nodes

- **Second level interconnect (network) is not cache coherent**
  - Typically used in High Performane Computing: InfiniBand
    - Latency: <= 5 us
    - Bandwidth: >= 1200 MB/s
  - Also used: GigaBit Ethernet:
    - Latency: <= 60 us
    - Bandwidth: >= 100 MB/s

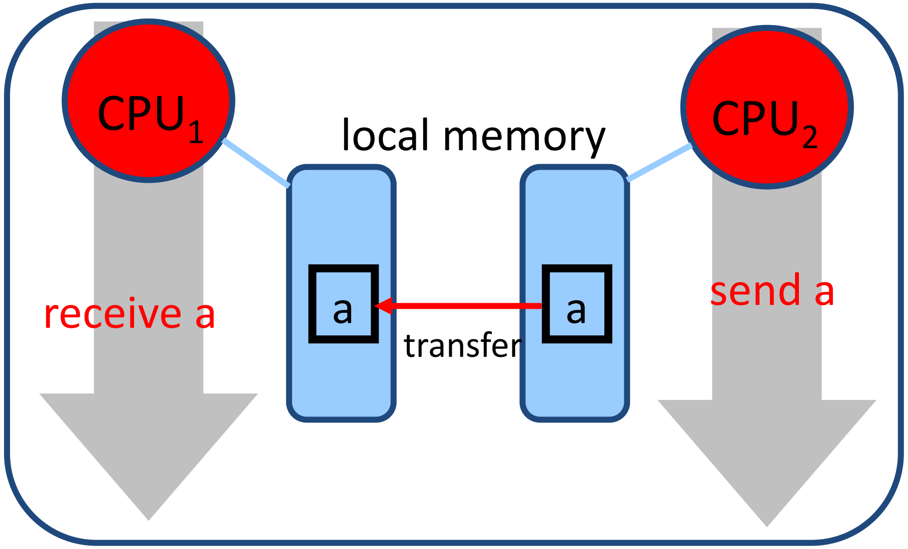


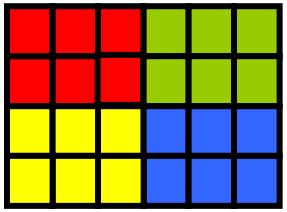
Latency: Time required to send a message of size zero (that is: time to setup the communication)

Bandwidth: Rate at which large messages (>= 2 MB) are transferred

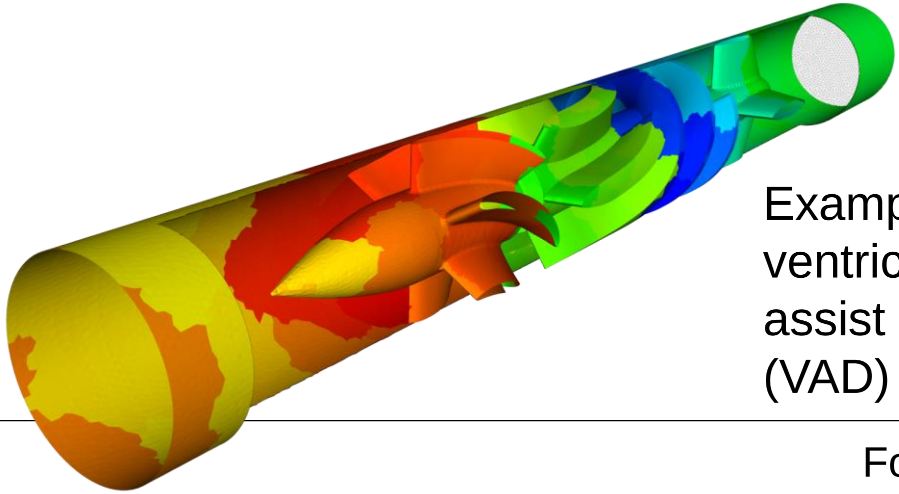
Distributed Memory Parallelization

- Each process has it‘s own distinct memory
- Communication via Message Passing



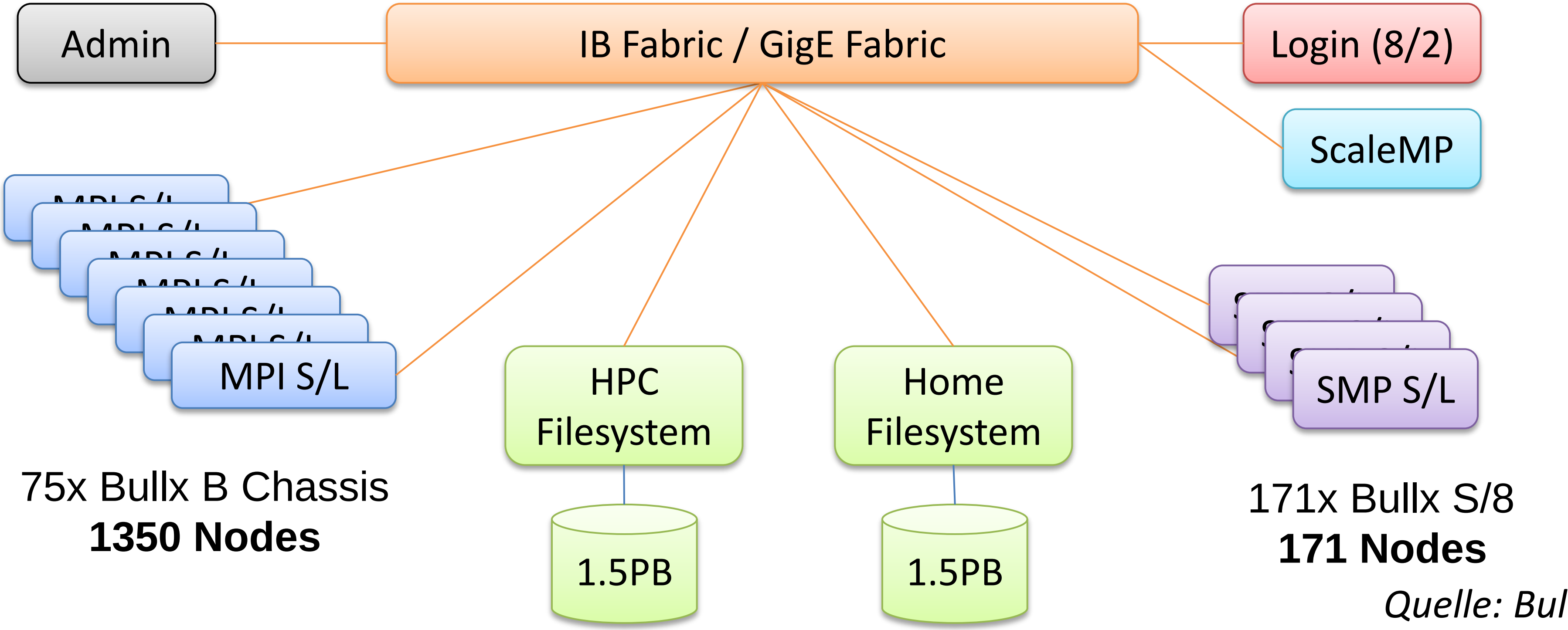


Decompose data into distinct chunks to be processed independently (data parallelism)



Example: ventricular assist device (VAD)

Our HPC Cluster from Bull: Overview



Admin

IB Fabric / GigE Fabric

Login (8/2)

ScaleMP

MPI S/L

HPC Filesystem

1.5PB

Home Filesystem

1.5PB

SMP S/L

75x Bullx B Chassis
1350 Nodes

171x Bullx S/8
171 Nodes

Quelle: Bull

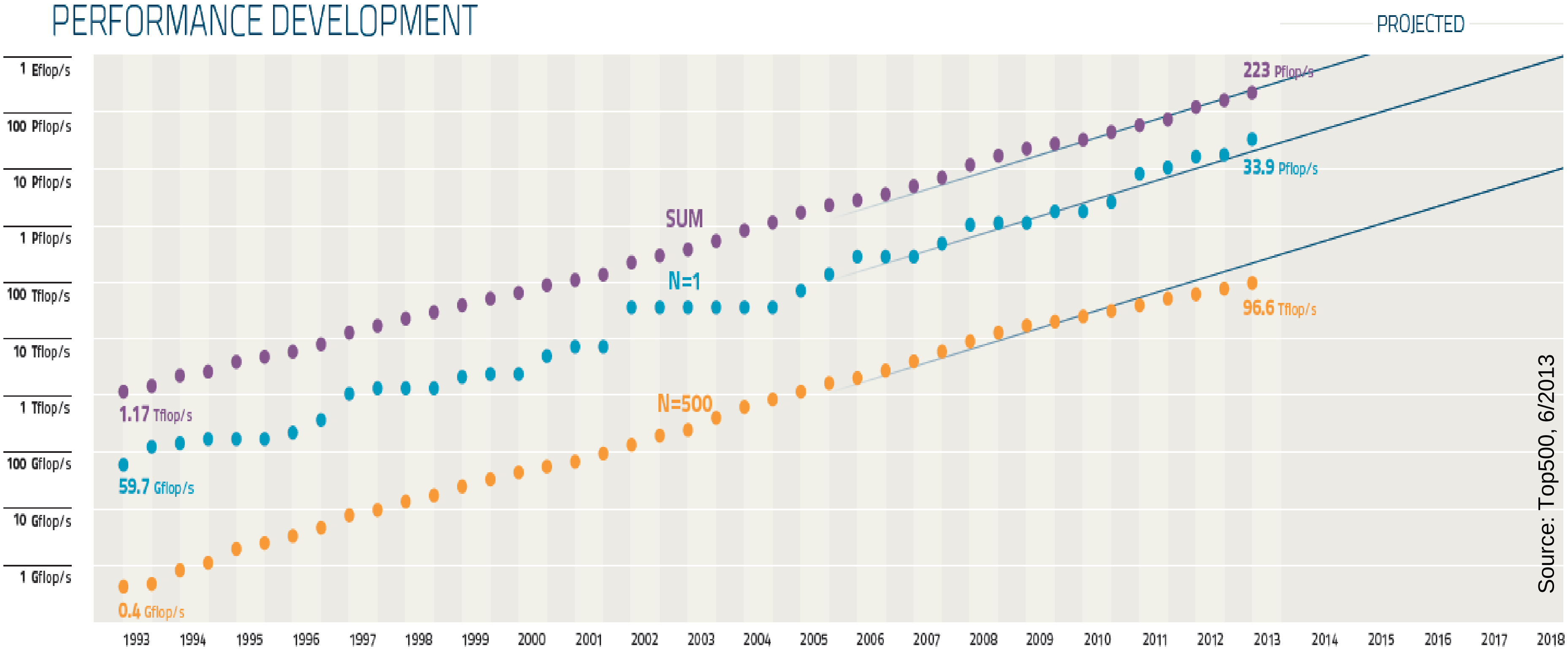
Group	# Nodes	Sum TFLOP	Sum Memory	# Procs per Node	Memory per Node
MPI-Small	1098	161	26 TB	2 x Westmere-EP (3,06 GHz, 6 Cores, 12 Threads)	24 GB
MPI-Large	252	37	24 TB		96 GB
SMP-Small	135	69	17 TB	4x Nehalem-EX (2,0 GHz, 8 Cores, 16 Threads)	128 GB
SMP-Large	36	18	18 TB		512 GB
ScaleMP-vSMP	8 gekoppelt	4	4 TB	BCS: 4, vSMP: 16	4 TB gekoppelt

General Purpose Graphic Processing Units (GPGPUs)

Motivation

Why to care about accelerators?

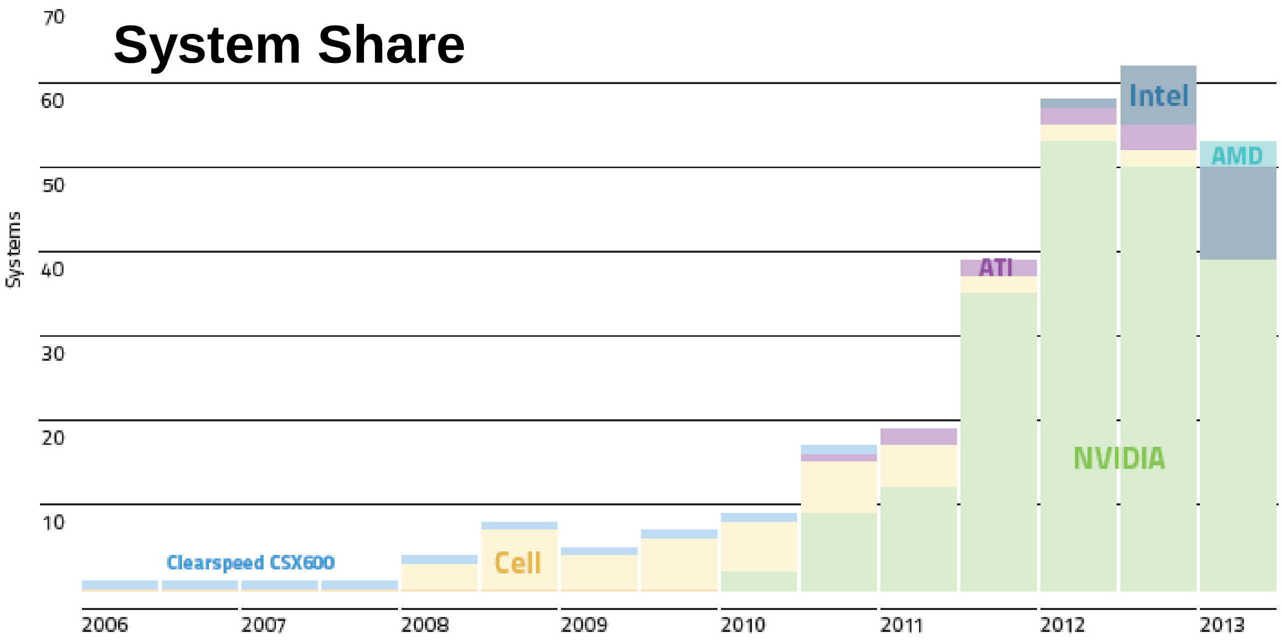
- Towards exa-flop computing (performance gain, but power constraints)
- Accelerators provide good performance per watt ratio (first step)



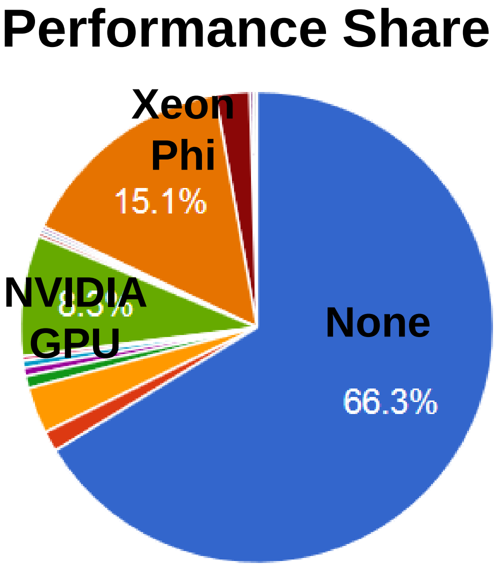
Source: Top500, 6/2013

Motivation

- **Accelerators/ co-processors**
  - GPGPUs (e.g. NVIDIA, AMD)
  - Intel Many Integrated Core (MIC) Arch. (Intel Xeon Phi)
  - FPGAs (e.g. Convey), …

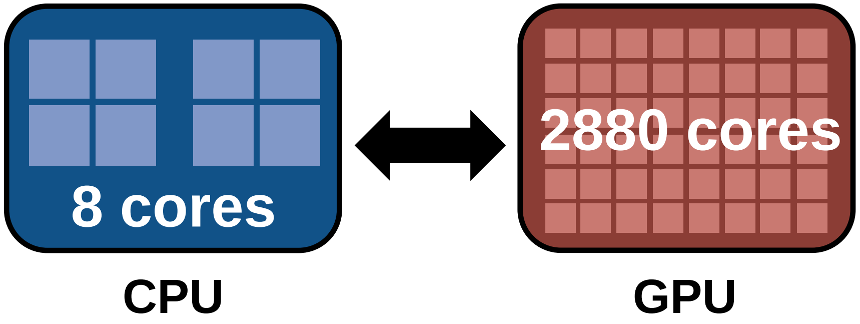


Source: Top500, 6/2013



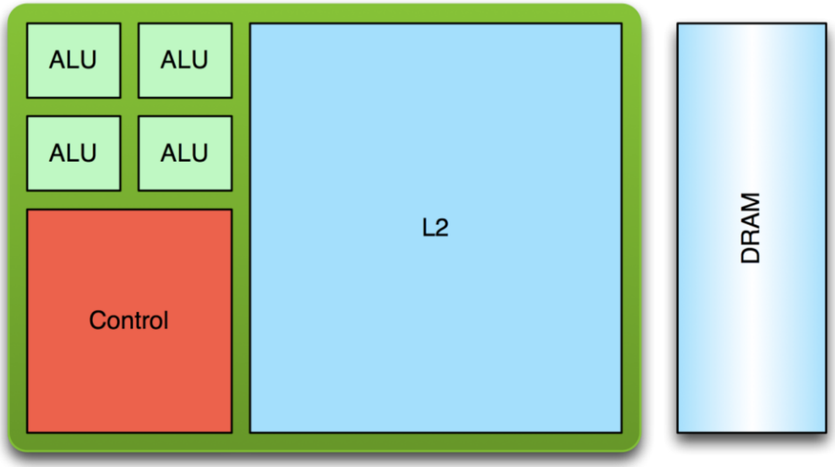
GPGPUs

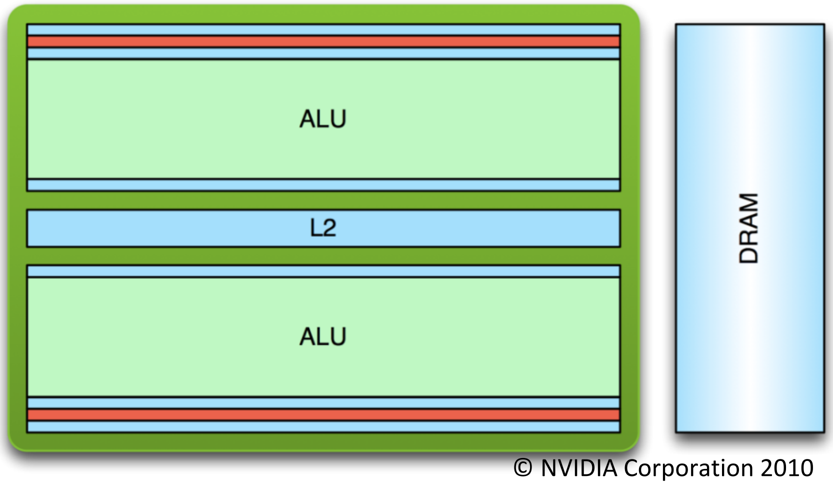
- **GPGPUs = General Purpose Graphics Processing Units**
  - From fixed-function graphics pipeline to programmable processors for general purpose computations
- **Programming paradigms**
  - CUDA, OpenCL, OpenACC, OpenMP 4.0,…
- **Main vendors**
  - NVIDIA, e.g. Quadro, Tesla, Fermi, Kepler
  - AMD, e.g. FireStream, Radeon
- **“Manycore architecture”**



Comparison CPU ⇔ GPU

Different design





© NVIDIA Corporation 2010

CPU

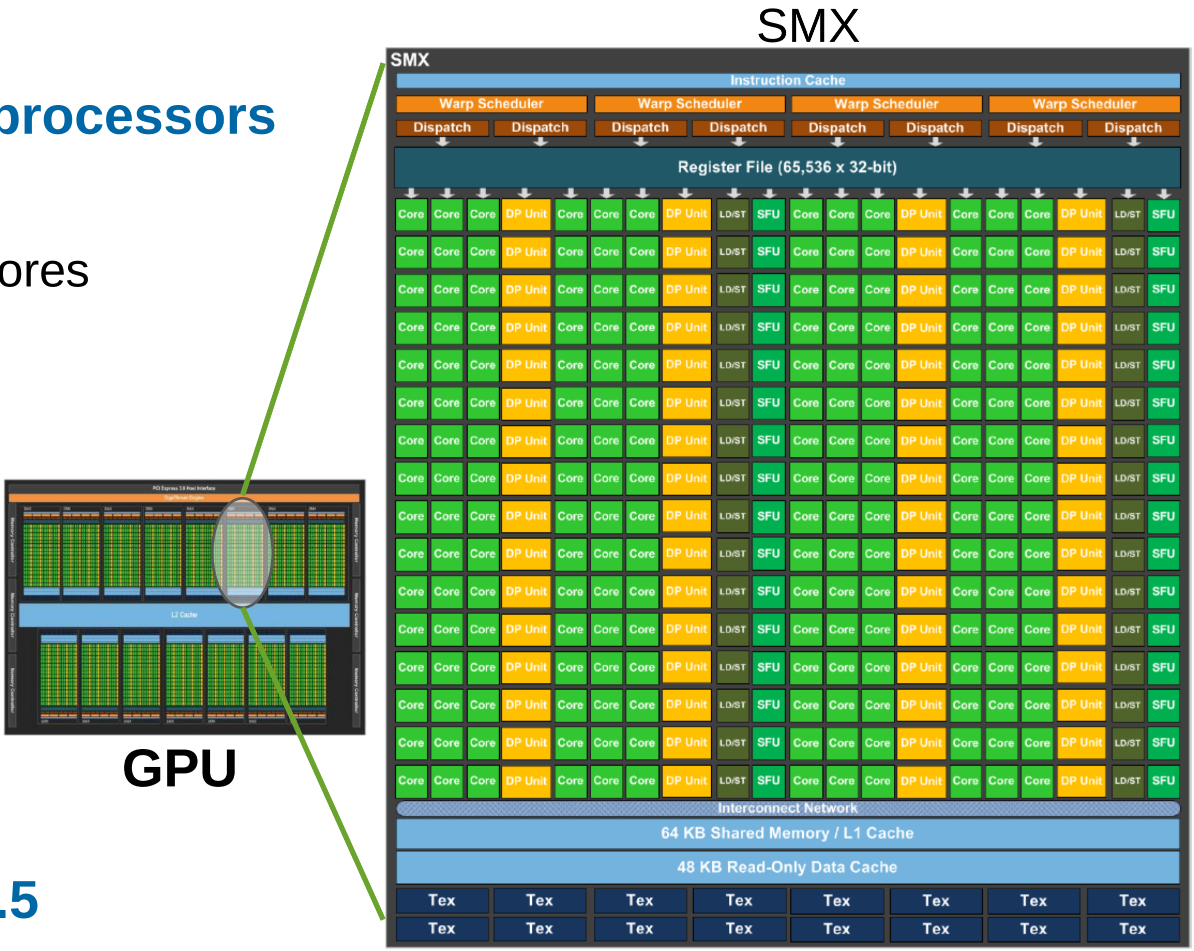
- Optimized for low latencies
- Huge caches
- Control logic for out-of-order and speculative execution

GPU

- Optimized for data parallel throughput
- Architecture tolerant of memory latency
- More transistors dedicated to computation

GPU architecture: NVIDIA Kepler

- **7.1 billion transistors**
- **13-15 streaming multiprocessors extreme (SMX)**
  - Each comprises 192 cores
- **2496-2880 cores**
- **Memory hierarchy**
- **Peak performance**
  - SP: 3.52 TFlops
  - DP: 1.17 TFlops
- **ECC support**
- **Compute capability: 3.5**
  - E.g. dynamic parallelism = possibility to launch dynamically new work from GPU



SMX

Instruction Cache

Warp Scheduler | Warp Scheduler | Warp Scheduler | Warp Scheduler

Dispatch | Dispatch | Dispatch | Dispatch | Dispatch | Dispatch | Dispatch | Dispatch

Register File (65,536 x 32-bit)

Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU
Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU Core Core Core DP Unit Core Core Core DP Unit LD/ST SFU

Interconnect Network

64 KB Shared Memory / L1 Cache

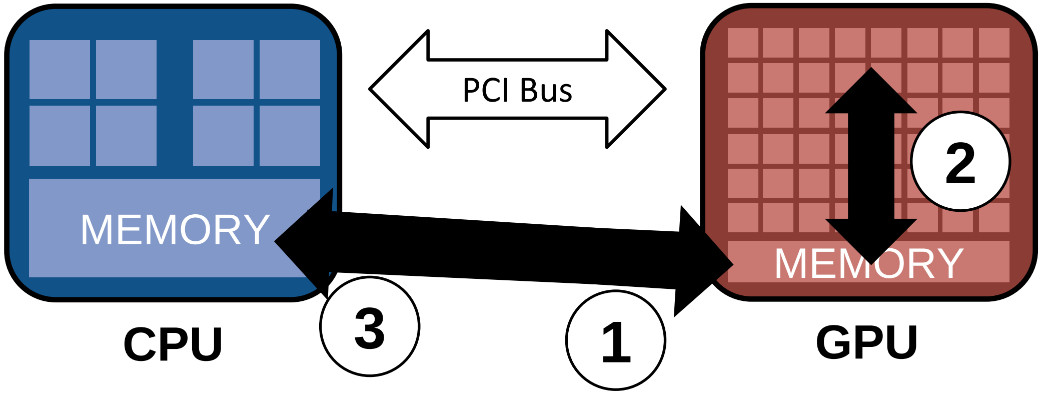
48 KB Read-Only Data Cache

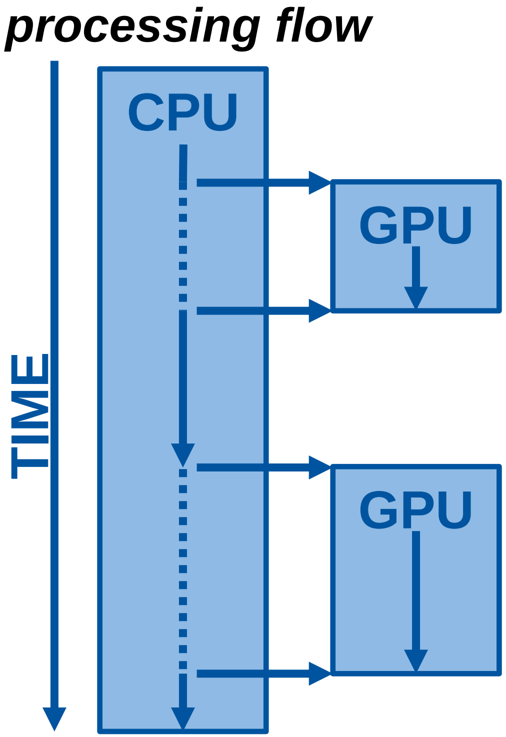
Tex Tex Tex Tex Tex Tex Tex Tex
Tex Tex Tex Tex Tex Tex Tex Tex

http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

Memory & Execution Model

- **(Weak) memory model**
  - Host + device memory = separate entities
  - No coherence between host + device
    - Data synchronization/transfers (triggered by host)



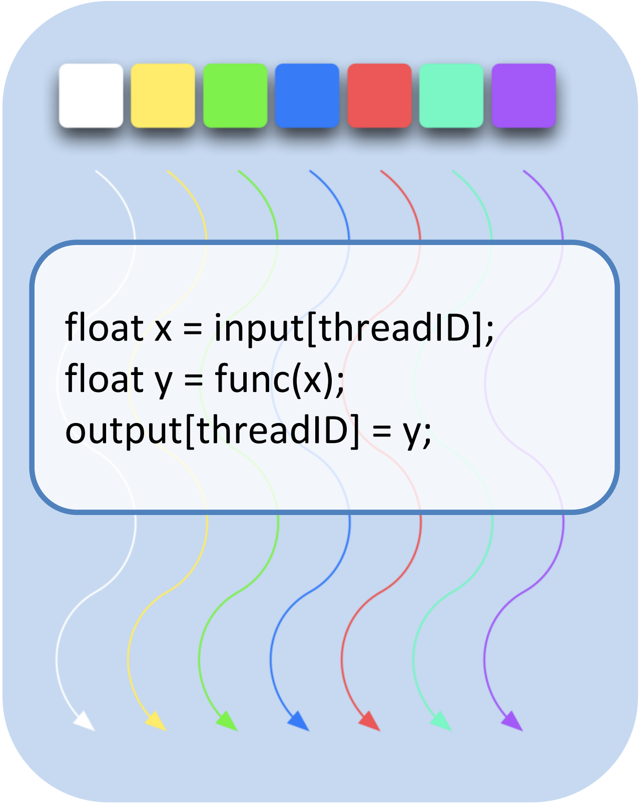


- **Host-directed execution model**
  1. Copy input data from CPU memory to GPU memory.
  2. Execute the GPU program.
  3. Copy results from GPU memory to CPU memory.

Programming Model

- **Parallel portion of application is executed as kernel on the device**
  - Kernel is executed as an array of threads
  - All threads execute the same code
- **GPU is a simple architecture**
  - Only few applications suit onto GPUs
  - Optimize data accesses

    (e.g. SoA instead of AoS, use on-chip memory)
  - Launch many many threads to hide latencies
    - GPU-“threads” are lightweight and fast switching
    - Stalls are hidden by switching to another without a stall



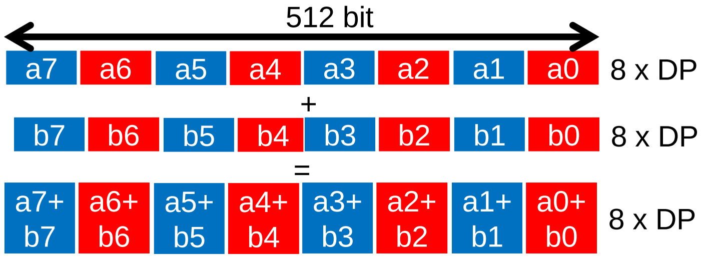
```
float x = input[threadID];
float y = func(x);
output[threadID] = y;
```

© NVIDIA Corporation 2010

Intel Xeon Phi Coprocessor

MIC Architecture

- Many Integrated Core (MIC) Architecture
- PCI Express Card (“co-processor”)
- 60 cores @ 1090 MHz
  - 4 HW threads per core
- 512-bit SIMD vectors



512 bit

a7	a6	a5	a4	a3	a2	a1	a0	8 x DP

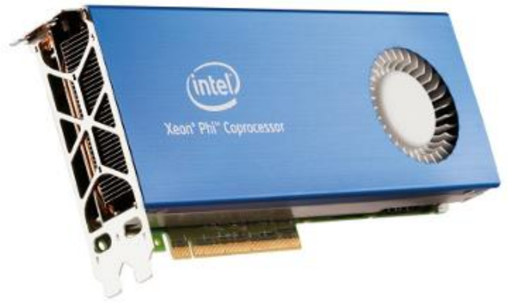
+

b7	b6	b5	b4	b3	b2	b1	b0	8 x DP

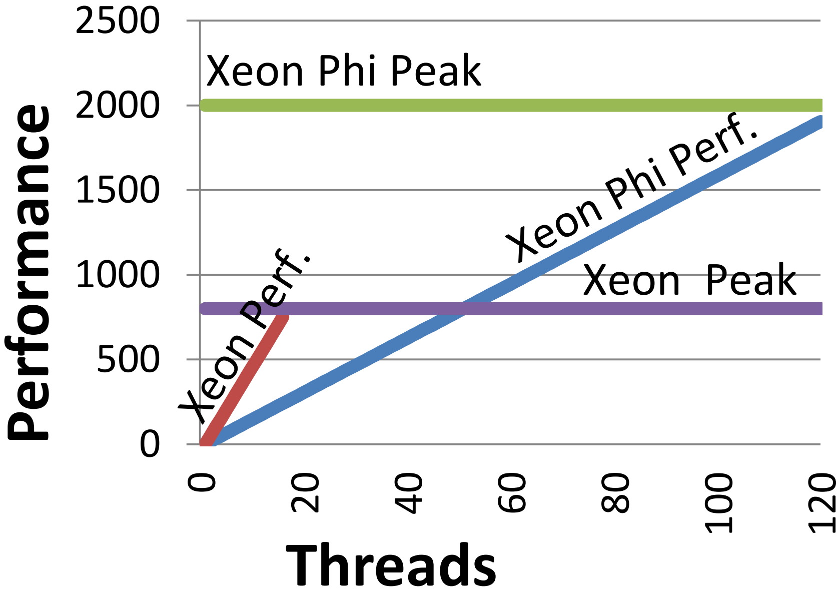
=

a7+ b7	a6+ b6	a5+ b5	a4+ b4	a3+ b3	a2+ b2	a1+ b1	a0+ b0	8 x DP

- Peak Performance
  - DP: ~ 1 TFlops
- Memory hierarchy



Source: Intel

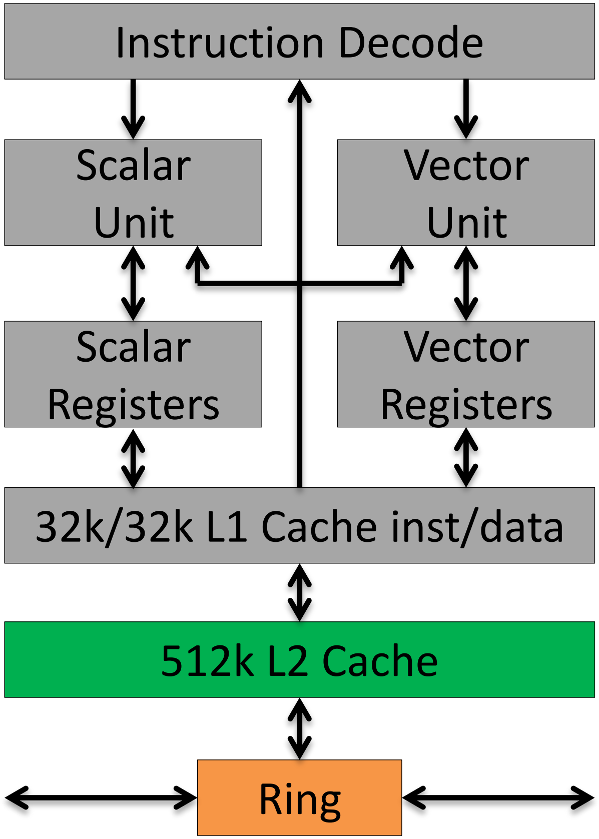


Source: James Reinders, Intel

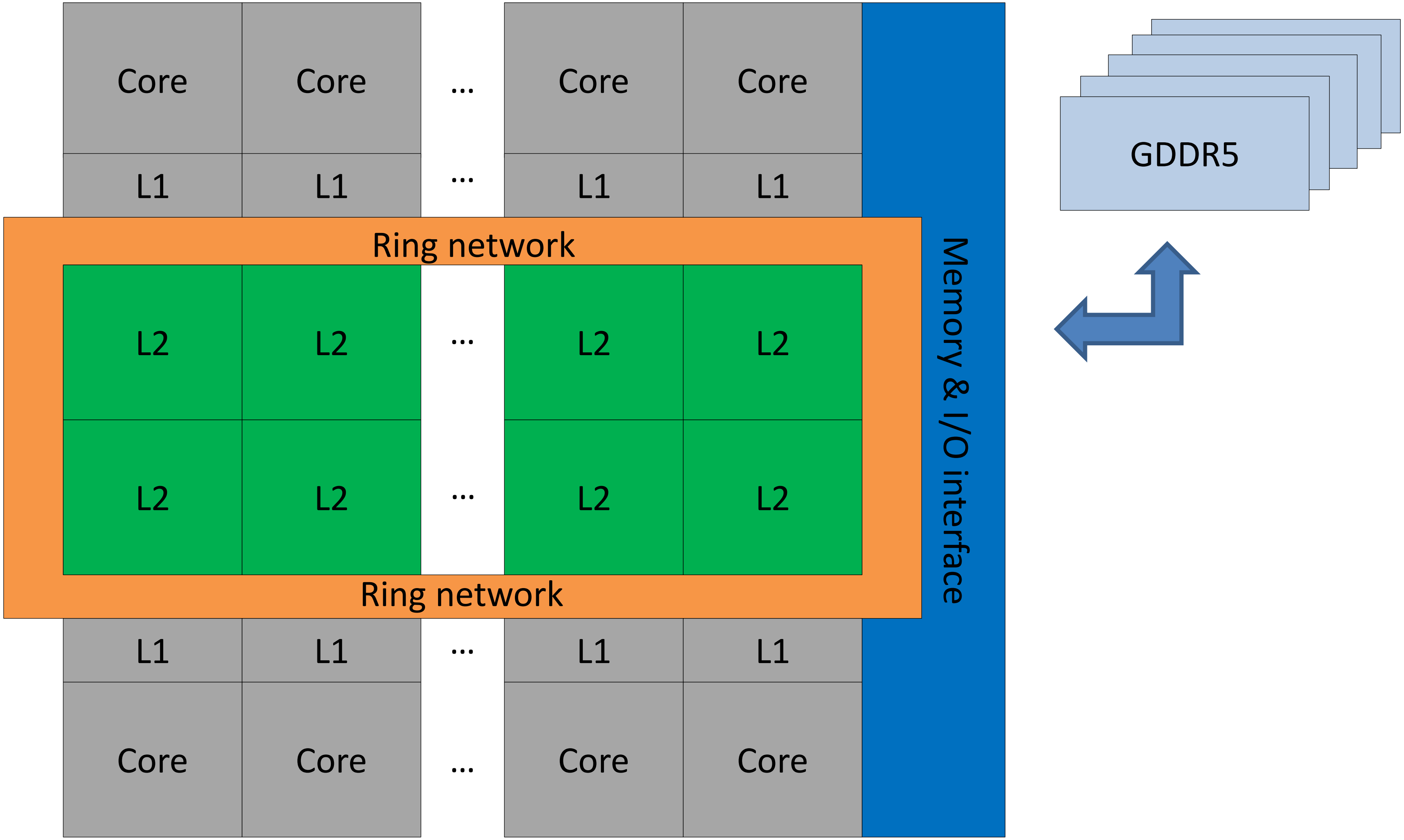
Architecture (1/2)

Source: Intel

Intel Xeon Phi Coprocessor
- 1 x Intel Xeon Phi @ 1090 MHz
- 60 Cores (in-order)
- ~ 1 TFLOPS DP Peak
- 4 hardware threads per core
- 8 GB GDDR5 memory
- 512-bit SIMD vectors (32 registers)
- Fully-coherent L1 and L2 caches
- Plugged into PCI Express bus



Architecture (2/2)



Core Core … Core Core

L1 L1 … L1 L1

Ring network

L2 L2 … L2 L2

L2 L2 … L2 L2

Ring network

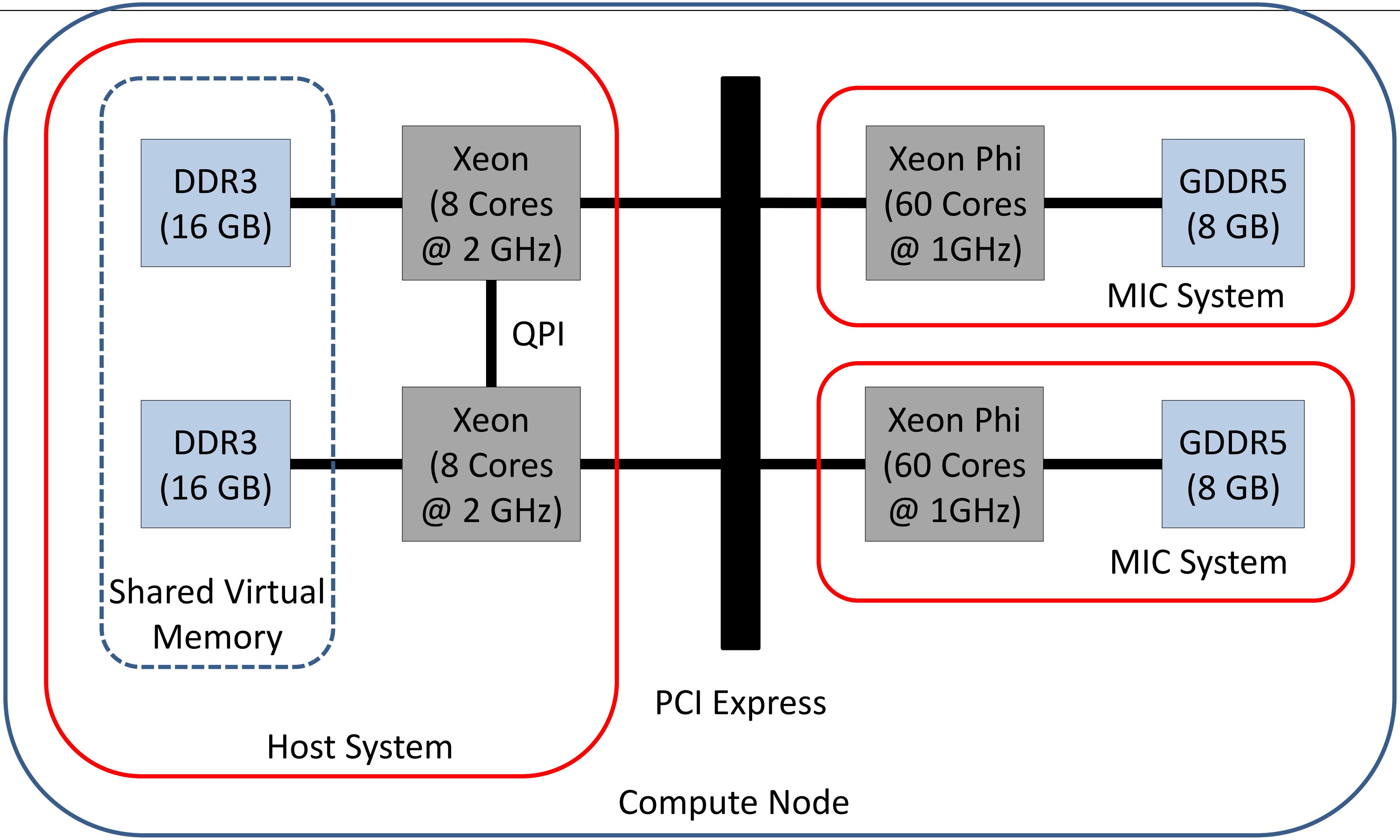
L1 L1 … L1 L1

Core Core … Core Core

Memory & I/O interface

GDDR5

Xeon Phi Nodes at RWTH Aachen

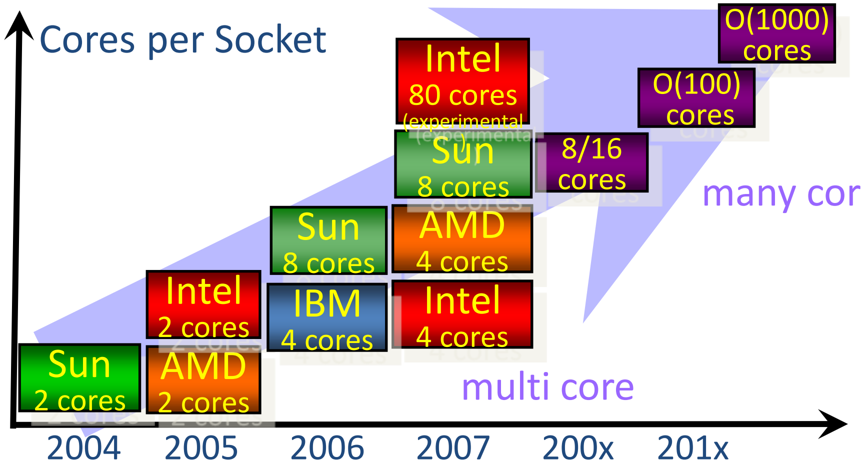


Summary and Conclusions

Summary and Conclusion

- **With a growing number of cores per system, even the desktop or notebook has become a parallel computer.**
  - Only parallel programs will profit from new processors!

- **SMP boxes are building blocks of large parallel systems.**
- **Memory hierarchies will grow further.**
- **CMP + SMT architecture is very promising for large, memory bound problems.**



- **Program optimization strategies:**
  - Memory Hierarchy
  - Many Cores
  - Tree-like networks

The End

Thank you for your attention.