High Performance and Parallel Computing (MA 6241)

Lecture 2

Basic Computer Technology

and Elements of Computer Architecture

Ulrich Rüde
Lehrstuhl für Simulation
Universität Erlangen-Nürnberg

visiting Professor at
Department of Mathematics
National University of Singapore

Organization of High Performance and Parallel Computing

- weekly Lecture: Tuesday 19:00-22:00
- Syllabus with more information will come soon
- No final exam
- 4 homework assignments
- grading (preliminary)
  - 10%
  - 20%
  - 20%
  - 50% (final project)
- Textbooks
  - Hager, G., & Wellein, G. (2010). Introduction to high performance computing for scientists and engineers. CRC Press.
  - Goedecker, S., & Hoisie, A. (2001). Performance optimization of numerically intensive codes. Siam.

First homework (10%)

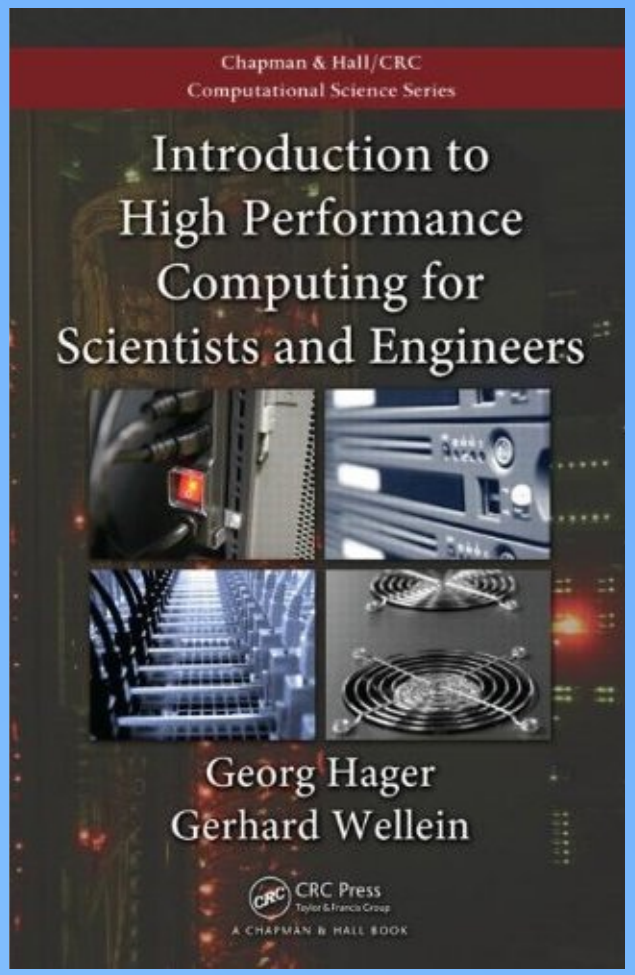
- Inform yourself in the internet about
  - LINPACK
  - TOP 500
  - what is the LINPACK performance of your own PC/ Laptop and predict how long will it therefore take to solve a dense system on your PC with N=1000 unknowns.
- Starting from there search for web sites with information on
  - computer performance and its evolution
  - benchmarking computer performance for scientific comp.
  - critique of LINPACK and TOP 500
  - technological trends
- Summarize your findings in an essay
  - with $2 \leqq p \leqq 4$ pages, and
  - submit as pdf file, due date Jan 25.

Homework

- I have not yet found time to establish my computer access/ e-mail/ web pages at NUS
- please submit homework as pdf and
- mail to

ulrich.ruede@fau.de

New HPC textbook



Georg Hager and Gerhard Wellein:

Introduction to High Performance Computing for Scientists and Engineers

CRC Press, ISBN 978-1439811924
356 pages
July 2010

"Georg Hager and Gerhard Wellein have developed a very approachable introduction to high performance computing for scientists and engineers. Their style and descriptions are easy to read and follow. … This book presents a balanced treatment of the theory, technology, architecture, and software for modern high performance computers and the use of high performance computing systems. The focus on scientific and engineering problems makes it both educational and unique. I highly recommend this timely book for scientists and engineers. I believe it will benefit many readers and provide a fine reference."
— *From the Foreword by Jack Dongarra, University of Tennessee, Knoxville, USA*

Georg Hager & Gerhard Wellein:

Introduction to High Performance Computing for Scientists and Engineers

- Covers basic sequential optimization strategies and the dominating parallelization paradigms, including shared-memory parallelization with OpenMP and distributed-memory parallel programming with MPI
- Highlights the importance of performance modeling of applications on all levels of a system's architecture
- Contains numerous case studies drawn from the authors' invaluable experiences in HPC user support, performance optimization, and benchmarking
- Explores important contemporary concepts, such as multicore architecture and affinity issues
- Includes code examples in Fortran and, if relevant, C and C++
- Provides end-of-chapter exercises with solutions in an appendix
- http://www.hpc.rrze.uni-erlangen.de/HPC4SE/

Contents of High Performance and Parallel Computing

- Introduction
  - Computational Science and Engineering
  - High Performance Computing
- Computer Architecture
  - memory hierarchy
  - pipeline
  - instruction level parallelism
  - multi-core systems
  - parallel clusters
- Efficient Programming
  - computational complexity
  - efficient coding
  - architecture aware programming
  - shared memory parallel programming (OpenMP)
  - distributed memory parallel programming (MPI)
  - parallel programming with Graphics Cards

So What?

Example application:

- Flow with moving objects on the micro-scale (suspensions)
- 1μm lattice resolution, object size 5-10 μm (example: blood cells)
- We can simulate (with this resolution)
- on a laptop: 2 Million lattice cells = 0.002 mm^3 of liquid.
- on a supercomputer with 40 TByte of memory: 80 Billion (8×10^{10}) lattice cells = 80 mm^3= 0.08 ml of liquid
  - for blood additionally required: 400 Million blood cells
- on systems that reach peta scale peak performance (will become available in the next generation of supercomputers)
  - 2 Trillion lattice cells = 2 cm^3 = 2 ml of liquid
  - ~ 10 Billion blood cells
- on an systems that will reach peta scale performance on real applications (should become available next decade)
  - 20 Trillion lattice cells = 20 ml of liquid

Is this good for anything?

Performance Requirements for Molecular Dynamics Simulations

In molecular dynamics (MD), the forces between atoms must be computed in each time step. This must be kept as cheap as possible, in particular we must avoid that each atom/molecule interacts individually with each other one.

In some applications (e.g. in material science), we would like to compute ensembles of many millions of atoms.

(Remember Loschmidt's constant = $2{,}687 \times 10^{25}\ \mathrm{m}^{-3}$)

Due to physical restrictions, a time step is very short: Picoseconds (10^{-12}) or Femtoseconds (10^{-15}) since we must resolve thermal oscillations. Therefore even to simulate a nanosecond (10^{-9}), we may need thousands to millions of time steps.

Molecular dynamics has an insatiable hunger for computer performance!

Current and Future High Performance (Super-)Computers

What is computer performance?

Relevant performance metrics:

- Floating Point operations/ second (Flops)
- Memory capacity (Byte)
- Speed of Memory

Throughput vs. Latency:

- Throughput = number of operations performed per second
- Latency = how long does it take for an operation to complete

(operation = e.g. floating point operation, memory read operation, communication operation, etc.)

Parallel execution $\Rightarrow$ Throughput $\neq$ 1/Latency

Especially important on (all) computer systems:

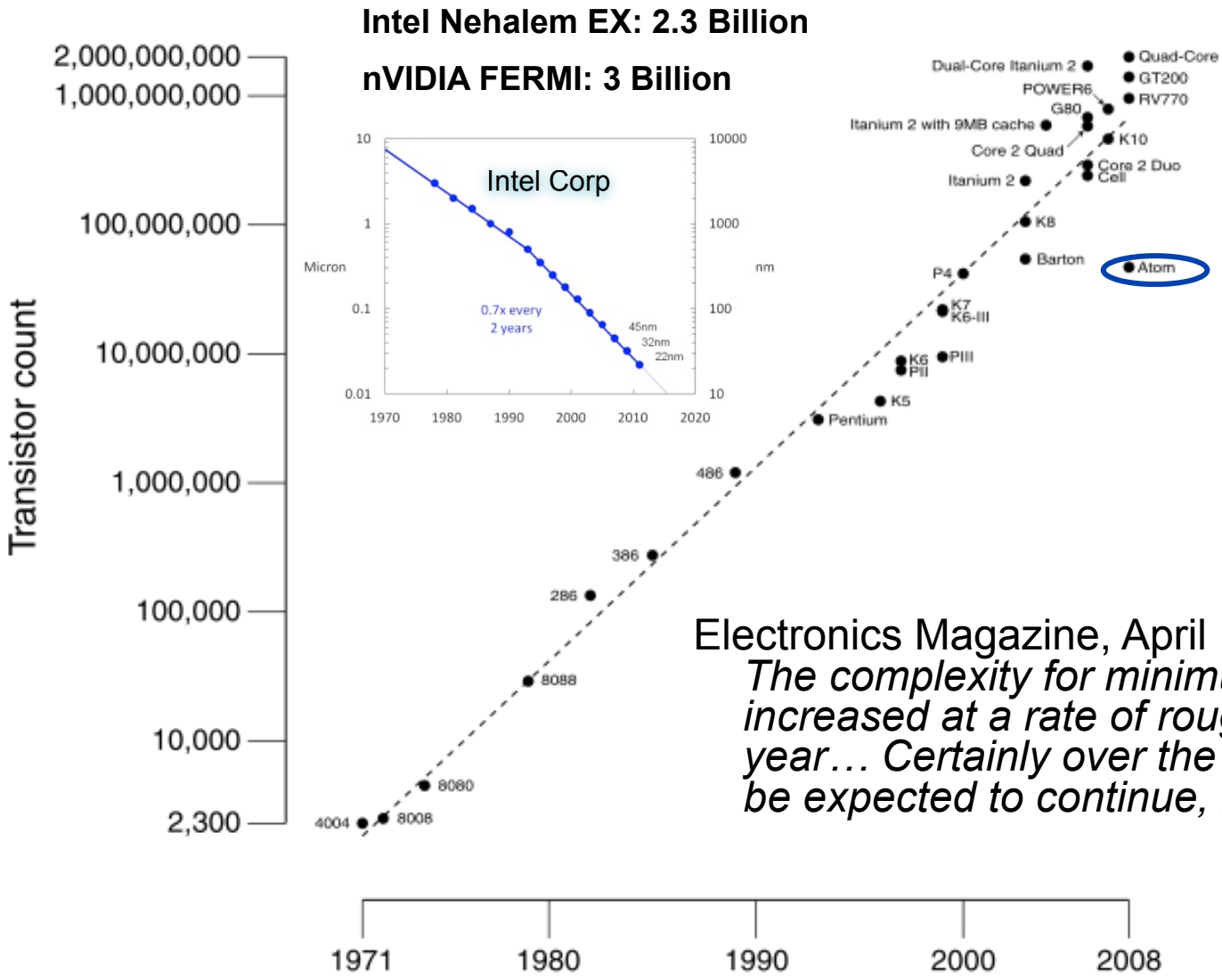
- Throughput and Latency of the
  - memory system and of the
  - network connecting different parts of the computer system.
- The throughput of data transfer operations is also called "bandwidth" (measured in bytes per second).

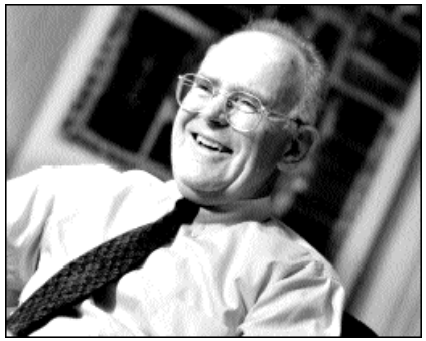
Evolution of Computer Technology?

Increase in processor performance:

- **Moore's law**
  is the observation that, over the history of computing hardware, the number of transistors in a dense integrated circuit doubles approximately every two years.
  - The observation is named after Gordon E. Moore, co-founder of the Intel Corporation
  - described the trend in 1965
  - ... double performance (for the same price) every 18 months (as long as clock speed went up)
- According to current development
  - this will continue for about one more decade
  - beyond ... we don't know
  - for the last 30 years, it has been predicted that Moore's law would last at least another decade

The driving force: Moore's law





Electronics Magazine, April 1965:

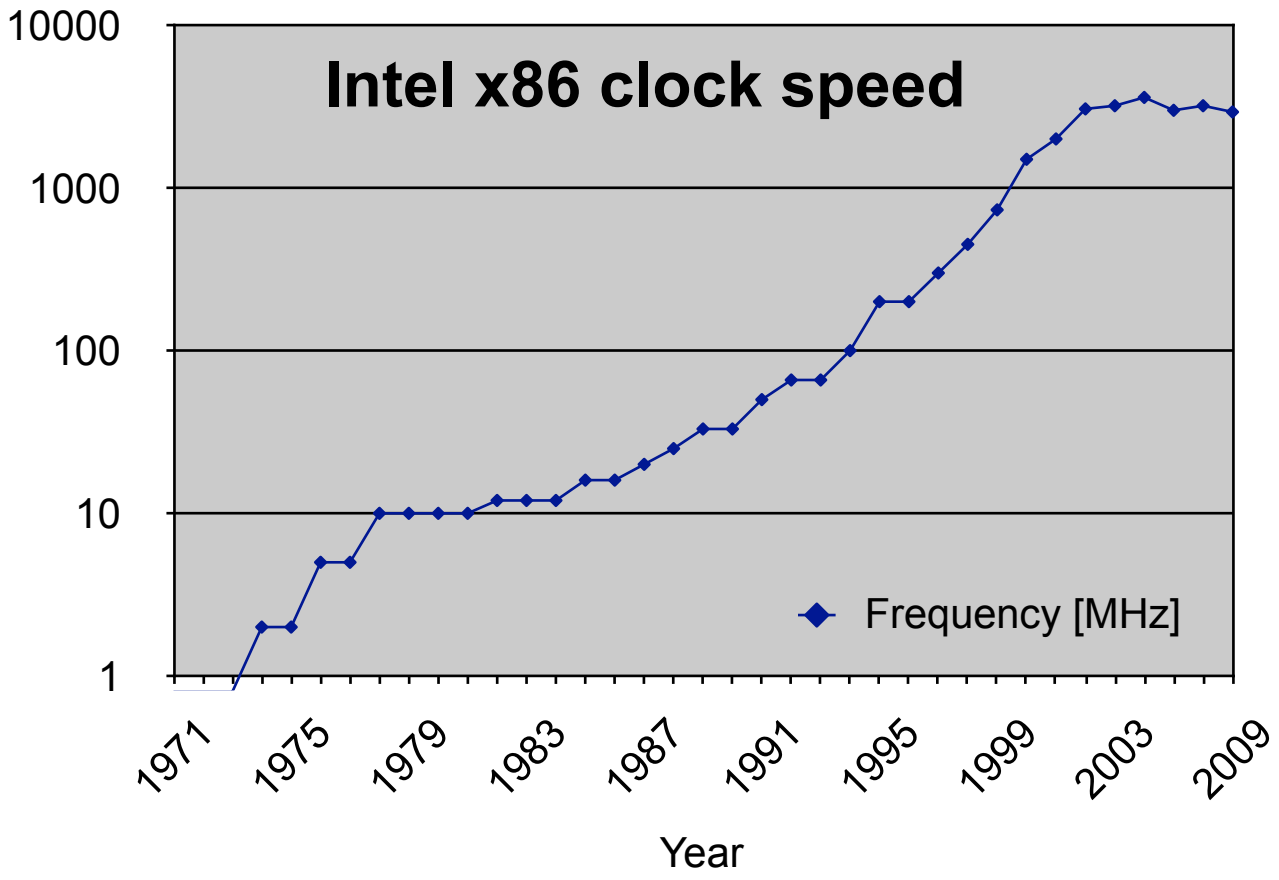
The complexity for minimum component costs has increased at a rate of roughly a factor of two per year… Certainly over the short term this rate can be expected to continue, if not to increase.

Evolution of Computer Technology?

- clock rate will increase only slowly
- but number of transistors per chip will continue to increase (possibl yslowing down some)
- parallelism on chip will continue to increase
  - used „transparently“
    - superscalar execution
    - pipelining
    - vectorization
    - ...
  - or burdening the programmer
    - vectorization
    - multi-core
- Memory wall
  - speed of memory (bandwith and latency do not keep up with CPU performance)
  - cache memory

The end of the clock speed race

Exponential growth of CPU clock speed for 15+ years



Better architectural features:

- **Pipelining**
- **Superscalarity**
- **SIMD / Vector ops**
- **Larger caches**

BUT

No fundamental architectural changes <u>until 2004/5</u>

Trading single thread performance for parallelism

- Power consumption (P) limits clock speed!
- At constant process technology: $P \sim V_C * f^2$ (potentially $V_C \sim f$)
- Core supply voltage approaches a lower limit: $V_C \sim 1$ V
- Thermal Design Power approaches economical limit: TDP ~ 80 W,…,130 W

P5 / 80586 (1993)	Pentium3 (1999)	Pentium4 (2003)	Core i7–960 (2009)
66 MHz	600 MHz	2800 MHz	3200 MHz
16 W @ V_C = 5 V	23 W @ V_C = 2 V	68 W @ V_C = 1.5 V	130 W @ V_C = 1.3
800 nm / 3 M	250 nm / 28 M	130 nm / 55 M	45 nm / 730 M
			Quad-Core

TDP / Core supply voltage

Process technology / Number of transistors in million

The Basic Semiconductor Technology

Integrated Circuits

Microprocessors and Memory

Evolution of Semiconductor Technology

- Collects trends of semiconductor technology
- Not easy to read, but excellent source of information
- See http://www.itrs.net/reports.html

GRAND CHALLENGES IN THE NEAR-TERM (THROUGH 2020) AND LONG-TERM (2021 AND BEYOND)

LOGIC DEVICE SCALING

The conventional path of scaling planar CMOS will face significant challenges set by performance and power consumption requirements.

...

Complementary metal–oxide–semiconductor (CMOS) is a technology for constructing integrated circuits. CMOS technology is used in microprocessors, microcontrollers, static RAM, and other digital logic circuits.
RAM = Random Access Memory

Evolution of Chip Technolgy

- From the semiconductor roadmap (old version, around 2002)

	2001	2003	2005	2008	2011	2014
nm	130	110	90	60	40	30
SRAM: M Tr/cm^2	70	128	234	577	1432	3510
Logic: M Tr/cm^2	13	24	44	109	265	664
M Tr/ Chip	122	244	488	1381	3907	11052
On Chip/local GHz	2.1	2.9	4.1	7.1	11.1	14.1
Chip-Board GHz	1.4	1.7	2	2.6	3.2	3.8
Number IO/chip	3000	3000	3200	3400	3800	4000

- From the semiconductor roadmap (2005 version/ 2006 update)

	2008	2011	2014	2020
nm	57	40	28	14
M Tr/ Chip	1106	2212	4424	17696
On Chip/local GHz	10	17	28	73
Chip-Board GHz	6	11	23	72
Number pins/chip	4400	5094	5896	7902

Summary 2013 Technology Trend Targets

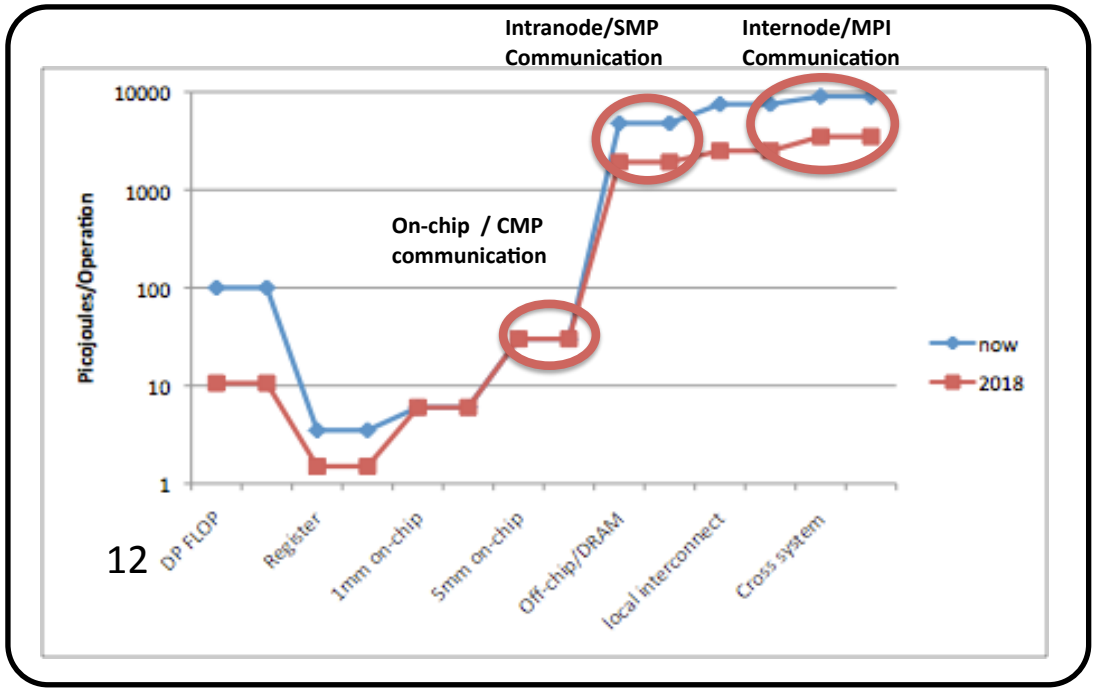
Year of Production	*2013*	*2015*	*2017*	*2019*	*2021*	*2023*	*2025*	*2028*
Logic Industry "Node Name" Label	**"16/14"**	**"10"**	**"7"**	**"5"**	**"3.5"**	**"2.5"**	**"1.8"**	
Logic ½ Pitch (nm)	**40**	**32**	**25**	**20**	**16**	**13**	**10**	**7**
Flash ½ Pitch [2D] (nm)	**18**	**15**	**13**	**11**	**9**	**8**	**8**	**8**
DRAM ½ Pitch (nm)	**28**	**24**	**20**	**17**	**14**	**12**	**10**	**7.7**
FinFET Fin Half-pitch (new) (nm)	**30**	**24**	**19**	**15**	**12**	**9.5**	**7.5**	**5.3**
FinFET Fin Width (new) (nm)	**7.6**	**7.2**	**6.8**	**6.4**	**6.1**	**5.7**	**5.4**	**5.0**
6-t SRAM Cell Size(um2) [@60f2]	**0.096**	**0.061**	**0.038**	**0.024**	**0.015**	**0.010**	**0.0060**	**0.0030**
MPU/ASIC HighPerf 4t NAND Gate Size(um2)	**0.248**	**0.157**	**0.099**	**0.062**	**0.039**	**0.025**	**0.018**	**0.009**
4-input NAND Gate Density (Kgates/mm) [@155f2]	**4.03E+03**	**6.37E+03**	**1.01E+04**	**1.61E+04**	**2.55E+04**	**4.05E+04**	**6.42E+04**	**1.28E+05**
Flash Generations Label (bits per chip) (SLC/MLC)	**64G /128G**	**128G /256G**	**256G / 512G**	**512G / 1T**	**512G / 1T**	**1T / 2T**	**2T / 4T**	**4T / 8T**
Flash 3D Number of Layer targets (at relaxed Poly half pitch)	**16-32**	**16-32**	**16-32**	**32-64**	**48-96**	**64-128**	**96-192**	**192-384**
Flash 3D Layer half-pitch targets (nm)	**64nm**	**54nm**	**45nm**	**30nm**	**28nm**	**27nm**	**25nm**	**22nm**
DRAM Generations Label (bits per chip)	**4G**	**8G**	**8G**	**16G**	**32G**	**32G**	**32G**	**32G**
450mm Production High Volume Manufacturing Begins (100Kwspm)				**2018**				
Vdd (High Performance, high Vdd transistors)[**]	**0.86**	**0.83**	**0.80**	**0.77**	**0.74**	**0.71**	**0.68**	**0.64**
1/(CV/I) (1/psec) [**]	**1.13**	**1.53**	**1.75**	**1.97**	**2.10**	**2.29**	**2.52**	**3.17**
On-chip local clock MPU HP [at 4% CAGR]	**5.50**	**5.95**	**6.44**	**6.96**	**7.53**	**8.14**	**8.8**	**9.9**
Maximum number wiring levels [unchanged	**13**	**13**	**14**	**14**	**15**	**15**	**16**	**17**
MPU High-Performance (HP) Printed Gate Length (GLpr) (nm) [**]	**28**	**22**	**18**	**14**	**11**	**9**	**7**	**5**
MPU High-Performance Physical Gate Length (GLph) (nm) [**]	**20**	**17**	**14**	**12**	**10**	**8**	**7**	**5**
ASIC/Low Standby Power (LP) Physical Gate Length (nm) (GLph)[**]	**23**	**19**	**16**	**13**	**11**	**9**	**8**	**6**

*** Note: from the PIDS working group data; however, the calibration of Vdd, GLph, and I/CV is ongoing for improved targets in 2014 ITRS work*

Why do we need to worry about energy and algorithms?

Thought Experiment

- 10^9 elements/nodes
- assume that every entity must contribute to any other (as is typical for an elliptic problem)
- equivalent to either
  - multiplication with inverse matrix
  - $n \times n$ Coulomb interaction
- Results in 10^{18} data movements
  - each 1 NanoJoule (optimistic)
- Together:
  10^9 Joule = 277 kWh



from: Exascale Programming Challenges, Report of the 2011 Workshop on Exascale Programming Challenges, Marina del Rey, July 27-29, 2011

Wh…al?

Tho…

- 1…
- a… m… (a… p…
- e…

- R… m…
- T…

…now
…2018

…of the
…nges,

Towards Exa-Scale!

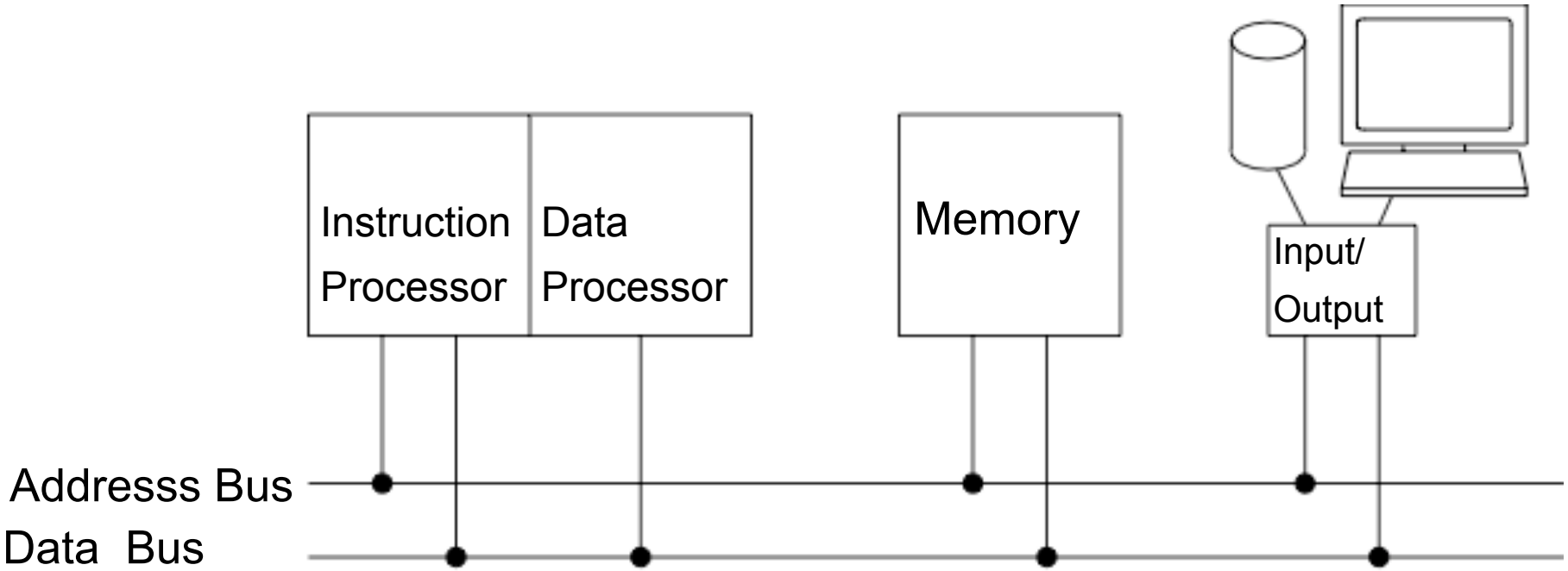
- The Leading Computer Architetcs are nowadays debating „Exa-Scale“ Computers: 10^{18} Flops/sec
- Supposedly ready by the end of the decade (2020?)
- „Peta-Scale was Easy“ (could be achieved by following existing trends)
- Exa-Scale requires many major research breakthroughs
  - „disruptive“ technology changes
  - Energy consumption
  - Fault tolerance
- Must reconsider algorithms and computational methods
- When an exascale computer shall remain below 10 MW power consumption, each Flop must use less than 10 PicoJoule = 10^{-11} Ws.
  - this means: all included, in particular mem access, cooling, ...

Elements of Computer Architecture

What is Computer Architecture?

- Functional design of computing machines here: for programmers of high performance (numerical) applications.
- A particular computer architecture may have an essential influence on the performance of a particular algorithm.
- Trend to more complex computer designs.
- Parallel computers
- superpipelined and superscalar
- Vector Processors
- Hierarchical Memory systems: Caches

Von-Neumann Computer Architecture



Levels of Parallelism

- Pipelining (in each functional unit)
  - fetch
  - decode
  - issue
  - execute
  - write back
  - commit
- Superscalar execution = multiple functional units (in each core)
  - 2 or more integer units
  - 2 or more load/store units
  - 2 or more floating point units
- multiple cores on each chip
  - either uniform
  - or dedicated special purpose
- Multiple Chips on same main board (e.g. sharing one main memory)
- Multiple main boards to build a large scale (physically distributed) memory parallel computer
- Multiple computers connected via the network (internet: grid computing)

Computer Architecture is Hierarchical

Core

- vectorization (SSE, AVX), i.e. vectors of 2-8 floating point numbers must be treated in blocks:
- may have their own cache memories
- access to local (cache) mem fast
- access to remote mem slow
- pipelining, superscalar execution
- each core may need several threads to hide memory access latency

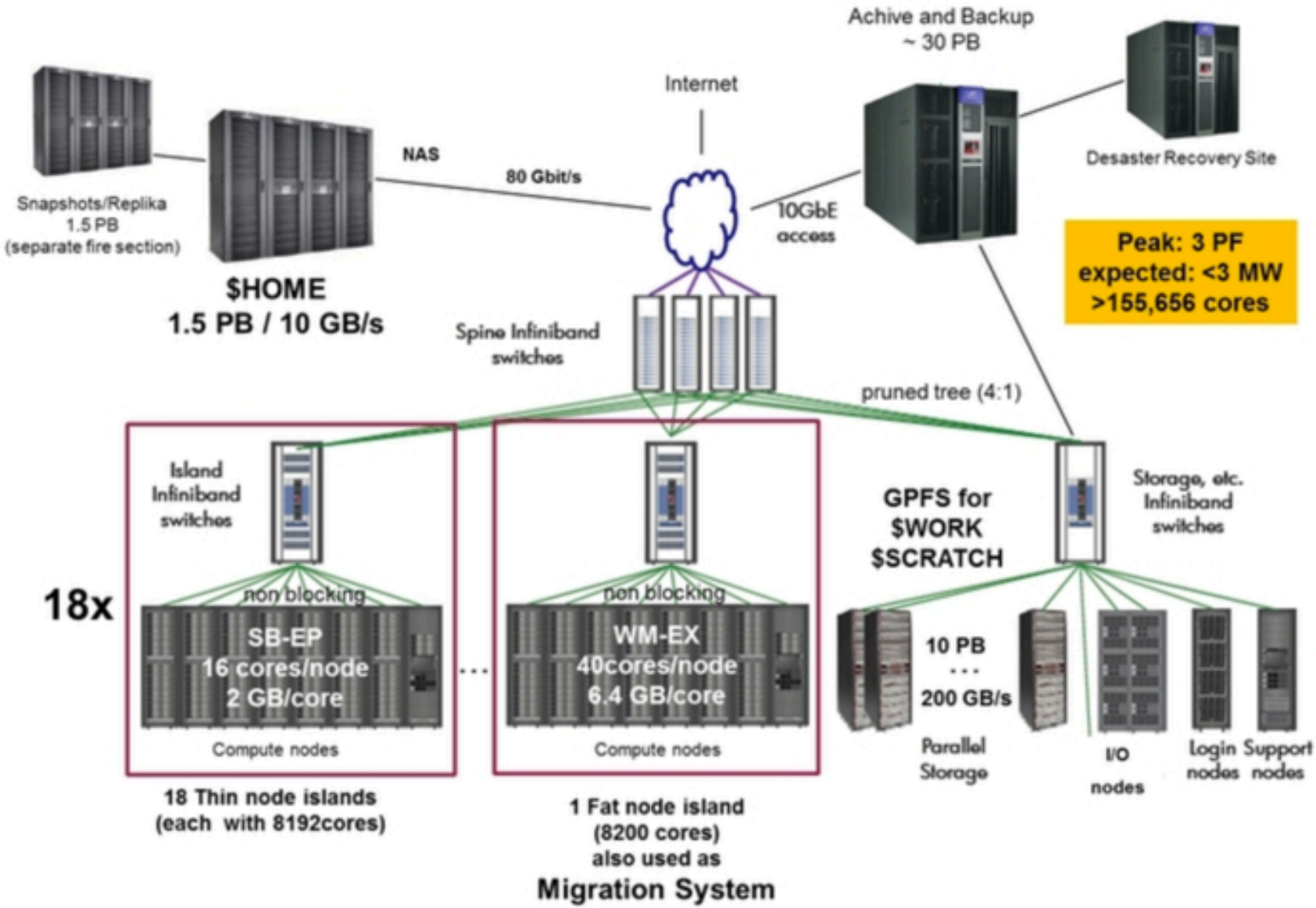
Node

- Several CPU chips (2) may be combined with local memory to become a node
- Several cores (8) are on a CPU chip
- Within a node we can use „shared memory parallelism“
- e.g. OpenMP
- Several cores may share second/third level caches
- Memory access bottlenecks may occur
- Sometimes nodes are equipped with accelerators (i.e. graphics cards)

Cluster

- thousands of nodes are connected by a fast network
- different network topologies
- between nodes message passing must be used
- e.g. MPI
- high latency
- low bandwidth

Schematic Overview of SuperMuc



Processors
Memory Architecture
Nodes

Bandwidth requirements

- 2 operands, 1 result per Flop (+,-*,/), 8 Bytes each
- 12 Gflops needs a bandwidth of 360 Gbyte/s
- Assume that a memory bus of 256 bits
  - used to transport 4 × 8 Bytes simultaneously
  - operates at 1 GHz
  - bandwidth is 32 Gbyte per second
  - this is only 1/10 of what is required
- To exploit the computational power, it is necessary to use data stored within the chip
  - registers
  - data caches

Latency requirements

- 2 Ghz corresponds to 0.5 ns per clock
- Typical memory systems have latencies 100-1000 times slower
- Remember:
  - light travels about 15 cm in 0.5 ns (in vaccuum)
  - It is impossible to build a large memory system that delivers data with latencies as short as the CPU clock
- In the future this will become worse

This is the

Memory Wall

which is sometimes also called the

von-Neumann Bottleneck.

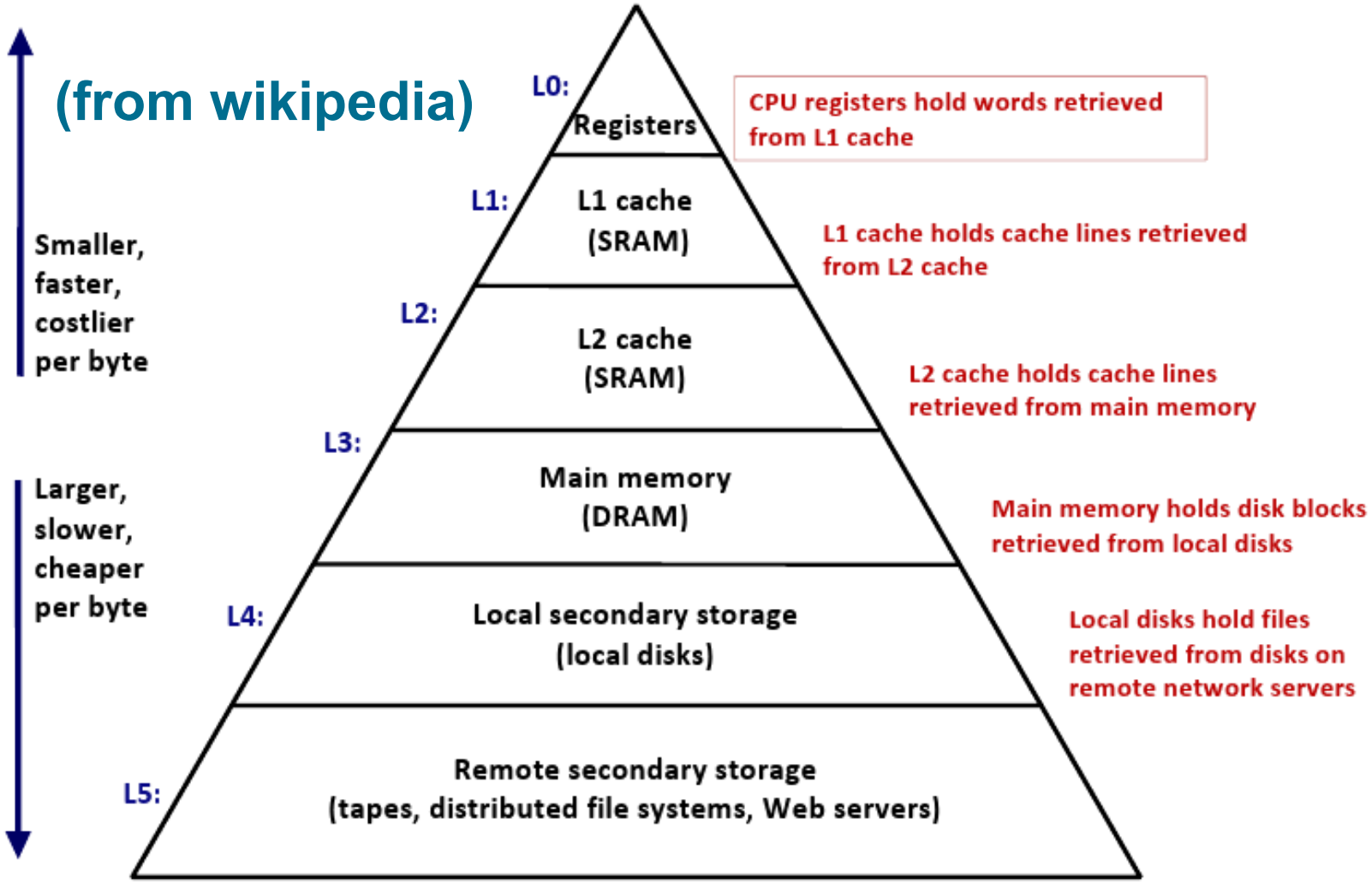
Neither latencies nor bandwith are sufficient to feed a high performance CPU chip if the data has to come from the memory system.

Memory wall

First of all, as chip geometries shrink and clock frequencies rise, the transistor leakage current increases, leading to excess power consumption and heat... Secondly, the advantages of higher clock speeds are in part negated by memory latency, since memory access times have not been able to keep pace with increasing clock frequencies. Third, for certain applications, traditional serial architectures are becoming less efficient as processors get faster (due to the so-called Von Neumann bottleneck), further undercutting any gains that frequency increases might otherwise buy. In addition, partly due to limitations in the means of producing inductance within solid state devices, resistance-capacitance (RC) delays in signal transmission are growing as feature sizes shrink, imposing an additional bottleneck that frequency increases don't address

taken from: Platform 2015: Intel® Processor and Platform Evolution for the Next Decade. March 2, 2005

Memory hierarchy

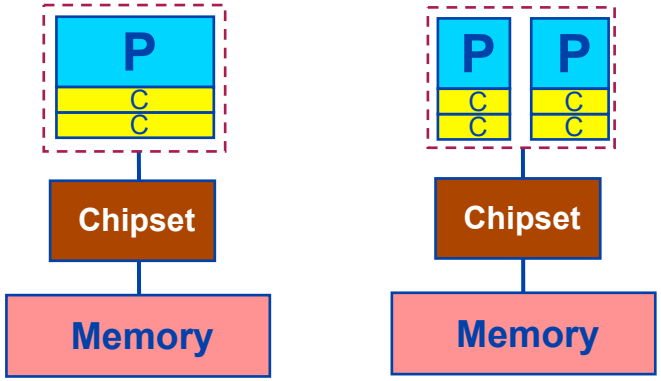


Multi-Core

The x86 multicore evolution

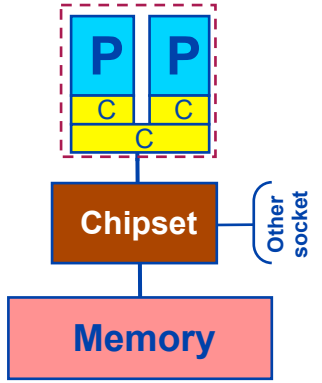
Intel Single-Dual-/Quad-/Hexa-/-Cores (one-socket view)

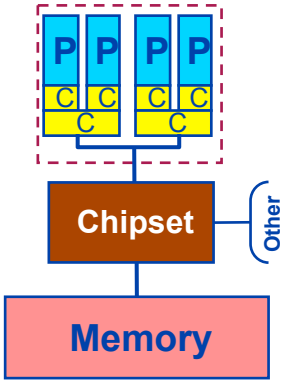
2005: "Fake" dual-core



2006: True dual-core

Woodcrest *"Core2 Duo"* 65nm

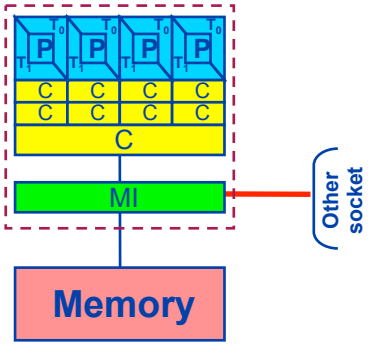




Harpertown *"Core2 Quad"* 45nm

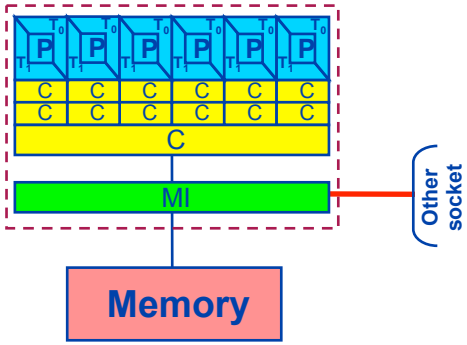
Approx. constant clock speed

2008: Hyperthreading/SMT is back!



Nehalem EP

"Core i7" 45nm



Westmere EP

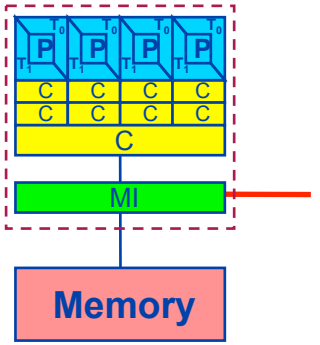
"Core i7"

32nm

2010/11: Wider SIMD units

SSE → AVX

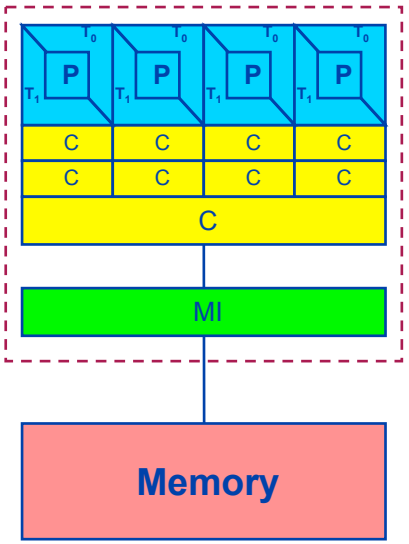
128 Bit → 256 Bit



Sandy Bridge (Desktop)

"Core i7"

32nm



Sandy Bridge
e.g.:
Intel Core i7-2600K

Total Floating Point (FP) Performance:

$P = n_{core} * F * S * \nu$

n_{core}	number of cores:	4
F	FP instructions per cycle: (1 MULT and 1 ADD)	2
S	FP operations / instruction: (256 Bit SIMD registers)	4 (dp)/ 8 (sp)
ν	Clock speed (Cycles per second):	3.4 GHz

P = 108.8 GF/s (dp) / 217.6 GF/s (sp)

But: P=6.8 GF/s (dp) for serial, "non-vectorized" code