High performance computing and numerical modeling

Volker Springel

Plan for my lectures

Lecture 1:	Collisional and collisionless N-body dynamics
Lecture 2:	Gravitational force calculation
Lecture 3:	Basic gas dynamics
Lecture 4:	Smoothed particle hydrodynamics
Lecture 5:	Eulerian hydrodynamics
Lecture 6:	Moving-mesh techniques
Lecture 7:	Towards high dynamic range
Lecture 8:	Parallelization techniques and current computing trends

Heidelberg Institute for Theoretical Studies HITS

Universität Heidelberg

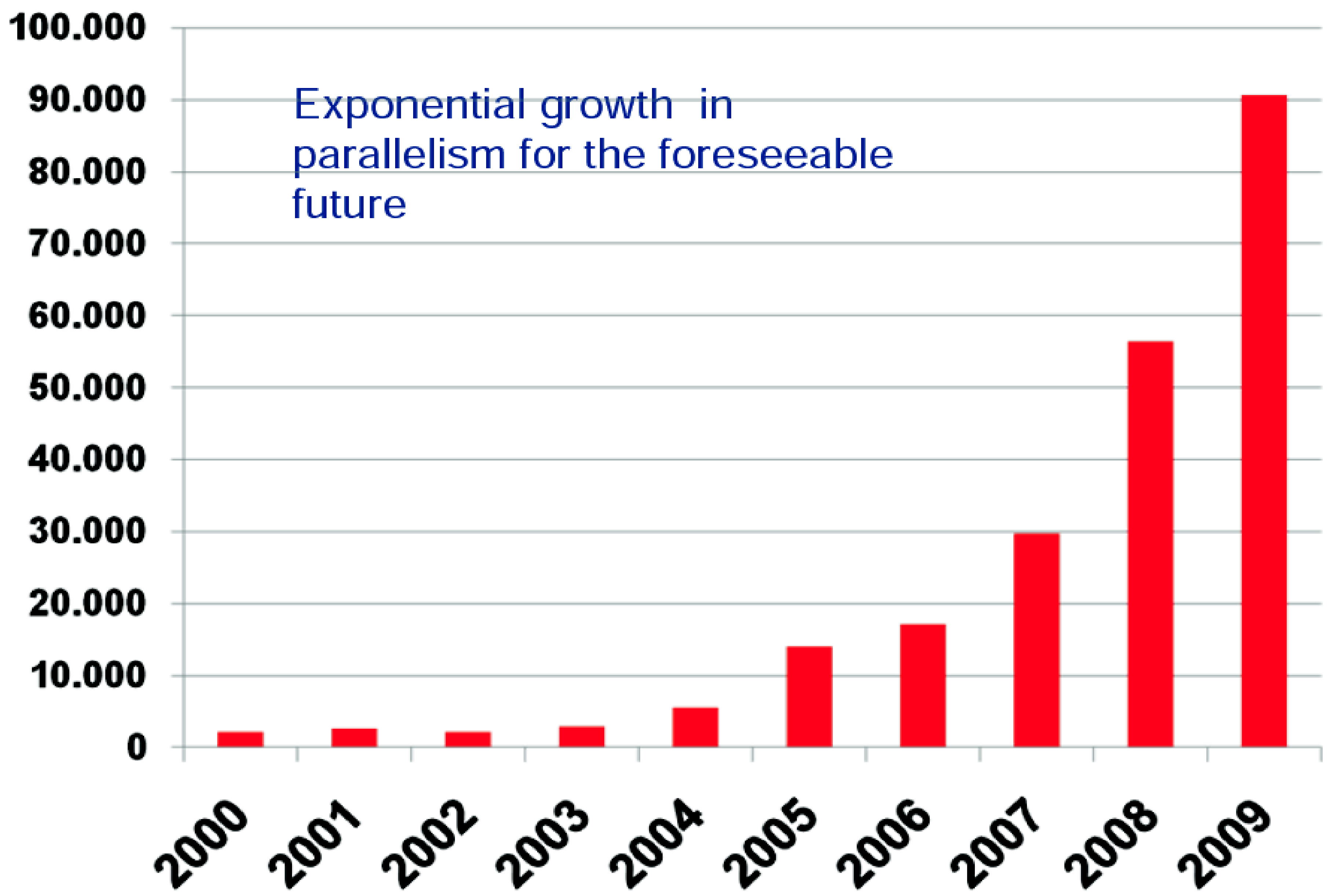
43rd Saas Fee Course
Villars-Sur-Ollon, March 2013

Future progress with cosmological simulations requires....

- **Better resolution (more computing power...)**
- **Higher accuracy of numerical codes**
- **More complete and realistic physics models**

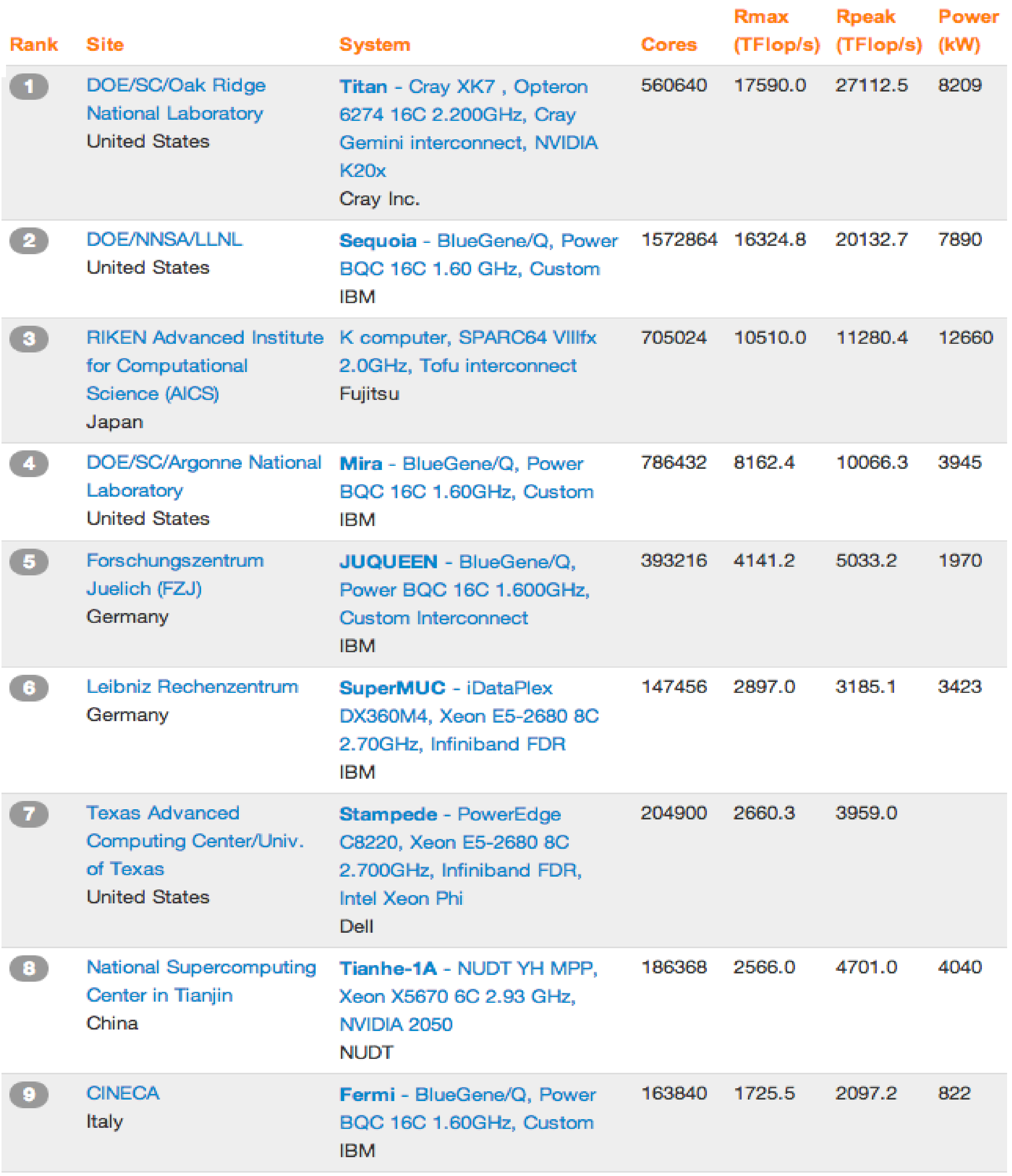
The number of cores on the top supercomputers grows exponentially

EXTREME GROWTH OF PARALLELISM



(figure by G. Sutmann)

Top500 List - November 2012



Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
2	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	16324.8	20132.7	7890
3	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
4	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8162.4	10066.3	3945
5	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4141.2	5033.2	1970
6	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi Dell	204900	2660.3	3959.0	
8	National Supercomputing Center in Tianjin China	**Tianhe-1A** - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT	186368	2566.0	4701.0	4040
9	CINECA Italy	**Fermi** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	163840	1725.5	2097.2	822

Currently typical supercomputers carry out about ~1-10 Petaflops

JUGENE IN JUELICH

Millennium-XXL

Largest high-resolution N-body simulation

303 billion particles

L = 3 Gpc/h

~700 million halos at z=0

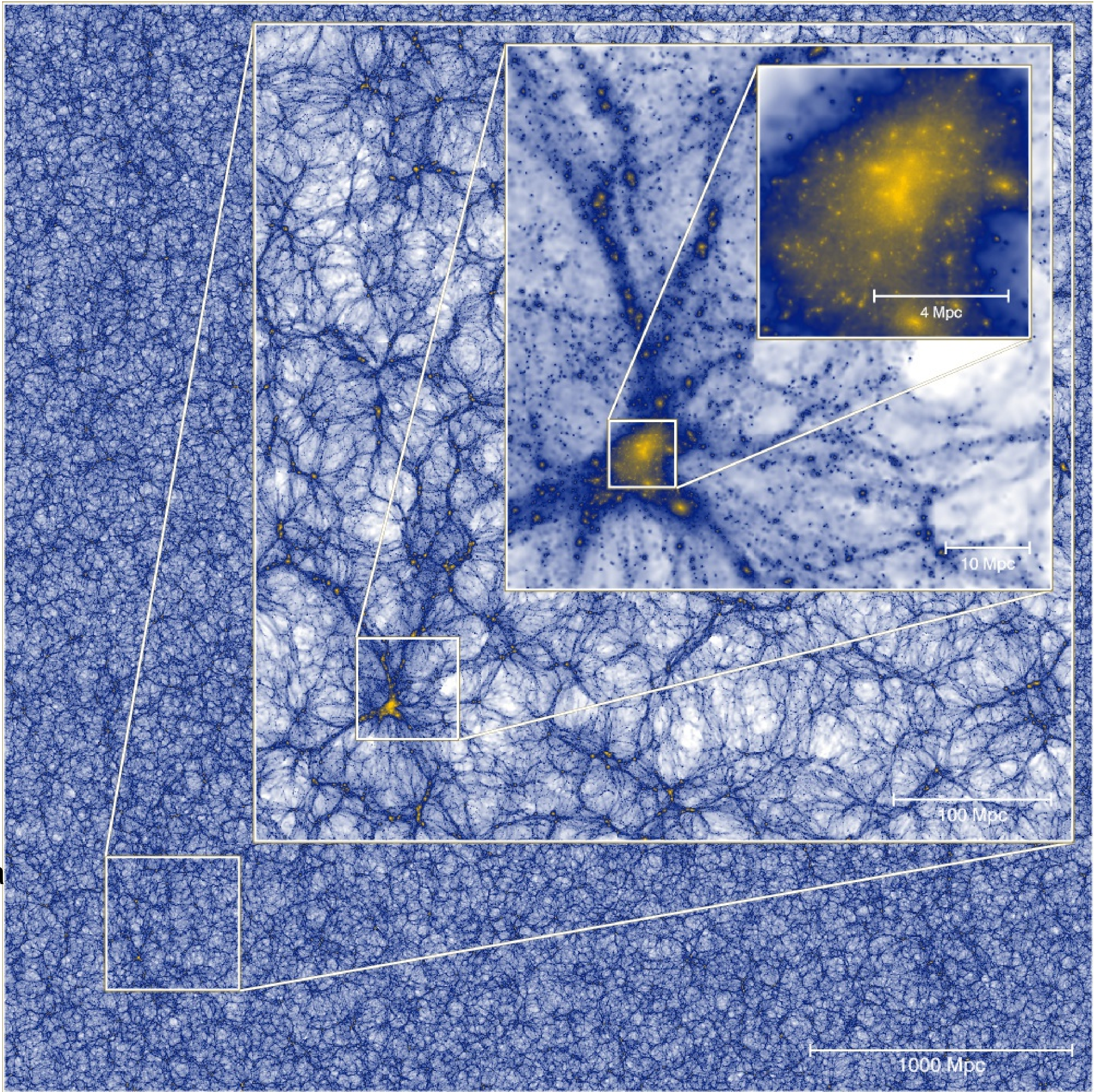
~25 billion (sub)halos in mergers trees

$m_p = 6.1 \times 10^9 \, M_\odot/h$

12288 cores, 30 TB RAM on Supercomputer JuRoPa in Juelich

2.7 million CPU-hours

Angulo et al. (2011)



Trouble ahead in the Exaflop regime ?

10^{18} in 2018

How long would the Millennium-XXL take on a Exaflop Supercomputer at peak performance?

15 min

One of the main problems:
Power Consumption

Petaflop Computer: **6 MW**

Exaflop Computer: **~ GW ?**

Need to get this down to **20-40 MW**

The Green500 List

Listed below are the November 2012 The Green500's energy-efficient supercomputers ranked from 1 to 10.

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	**2,499.44**	National Institute for Computational Sciences/University of Tennessee	Beacon - Appro GreenBlade GB824M, Xeon E5-2670 8C 2.600GHz, Infiniband FDR, Intel Xeon Phi 5110P	44.89
2	**2,351.10**	King Abdulaziz City for Science and Technology	SANAM - Adtech ESC4000/FDR G2, Xeon E5-2650 8C 2.000GHz, Infiniband FDR, AMD FirePro S10000	179.15
3	**2,142.77**	DOE/SC/Oak Ridge National Laboratory	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x	8,209.00
4	**2,121.71**	Swiss Scientific Computing Center (CSCS)	Todi - Cray XK7 , Opteron 6272 16C 2.100GHz, Cray Gemini interconnect, NVIDIA Tesla K20 Kepler	129.00
5	**2,102.12**	Forschungszentrum Juelich (FZJ)	JUQUEEN - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect	1,970.00
6	**2,101.39**	Southern Ontario Smart Computing Innovation Consortium/University of Toronto	BGQdev - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect	41.09
7	**2,101.39**	DOE/NNSA/LLNL	rzuseq - BlueGene/Q, Power BQC 16C 1.60GHz, Custom	41.09
8	**2,101.39**	IBM Thomas J. Watson Research Center	BlueGene/Q, Power BQC 16C 1.60GHz, Custom	41.09
9	**2,101.12**	IBM Thomas J. Watson Research Center	BlueGene/Q, Power BQC 16C 1.60 GHz, Custom	82.19
10	**2,101.12**	Ecole Polytechnique Federale de Lausanne	CADMOS BG/Q - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect	82.19

Parallelization with distributed memory

Many compute nodes connected through a fast network.

Programming usually through **Message Passing Library (MPI)**

"Beowulf cluster"

However, some higher level languages allow different programming models.

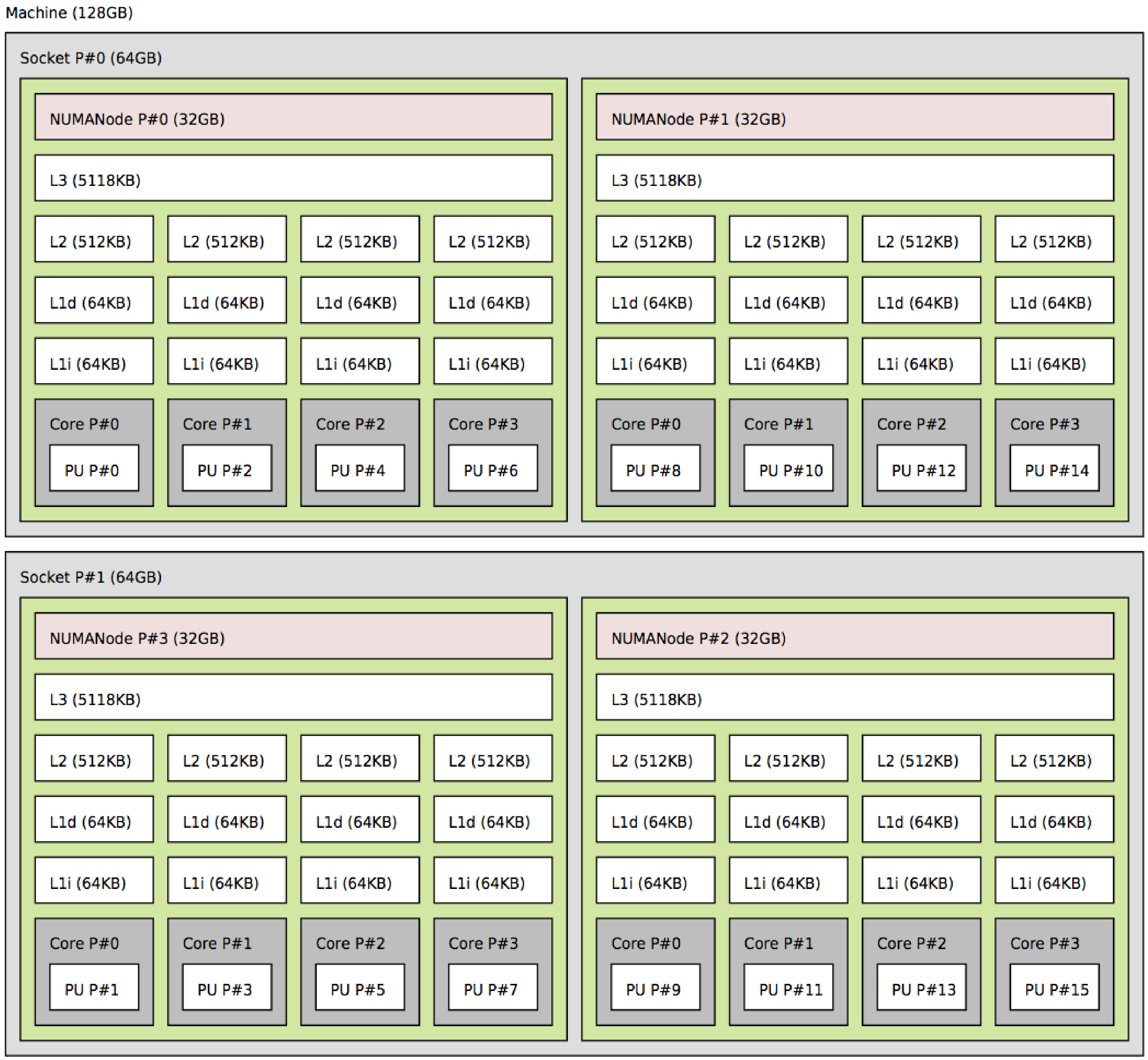
For example PGAS languages:

- Unified Parallel C
- Coarray Fortran
- Chapel

Or languages with a parallel runtime, like Charm++

These days, the compute nodes are usually SMP or ccNUMA machines

MULTIPLE SOCKETS, MULTIPLE CORES, AND MEMORY BANKS

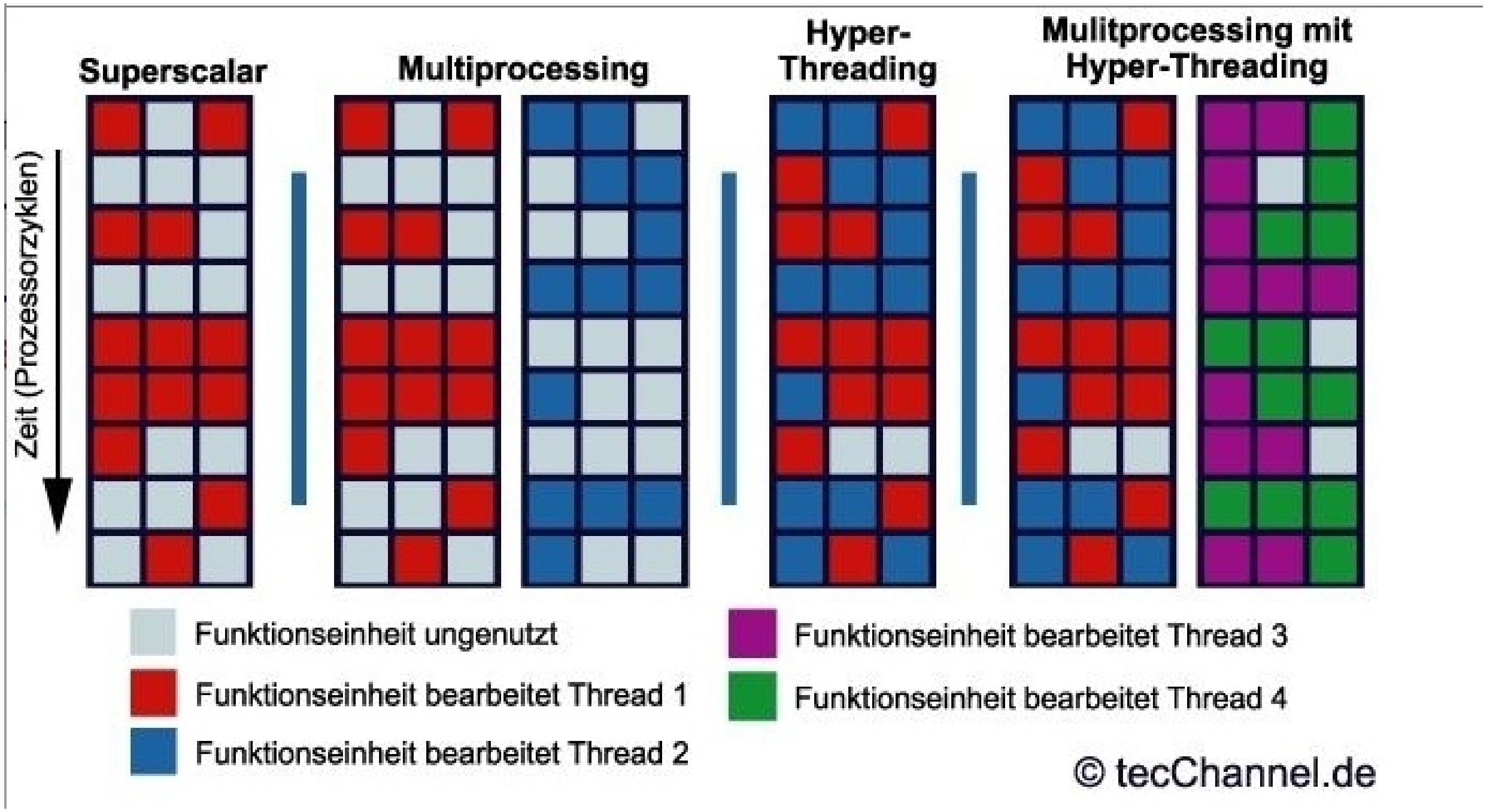

Machine (128GB)
Socket P#0 (64GB)
NUMANode P#0 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 PU P#0
Core P#1 PU P#2
Core P#2 PU P#4
Core P#3 PU P#6
NUMANode P#1 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 PU P#8
Core P#1 PU P#10
Core P#2 PU P#12
Core P#3 PU P#14
Socket P#1 (64GB)
NUMANode P#3 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 PU P#1
Core P#1 PU P#3
Core P#2 PU P#5
Core P#3 PU P#7
NUMANode P#2 (32GB)
L3 (5118KB)
L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)
L1d (64KB) L1d (64KB) L1d (64KB) L1d (64KB)
L1i (64KB) L1i (64KB) L1i (64KB) L1i (64KB)
Core P#0 PU P#9
Core P#1 PU P#11
Core P#2 PU P#13
Core P#3 PU P#15

A node with 4 sockets and 4 cores, in total 16 physical cores

Symmetric multi-threading (SMT) and hyperthreading (HT) of some modern CPUs add "virtual cores"

INCREASING THE USE OF CORE RESOURCES THROUGH HARDWARE ASSISTED THREADS

Intel and IBM produce CPUs whose cores support **hardware threads.**

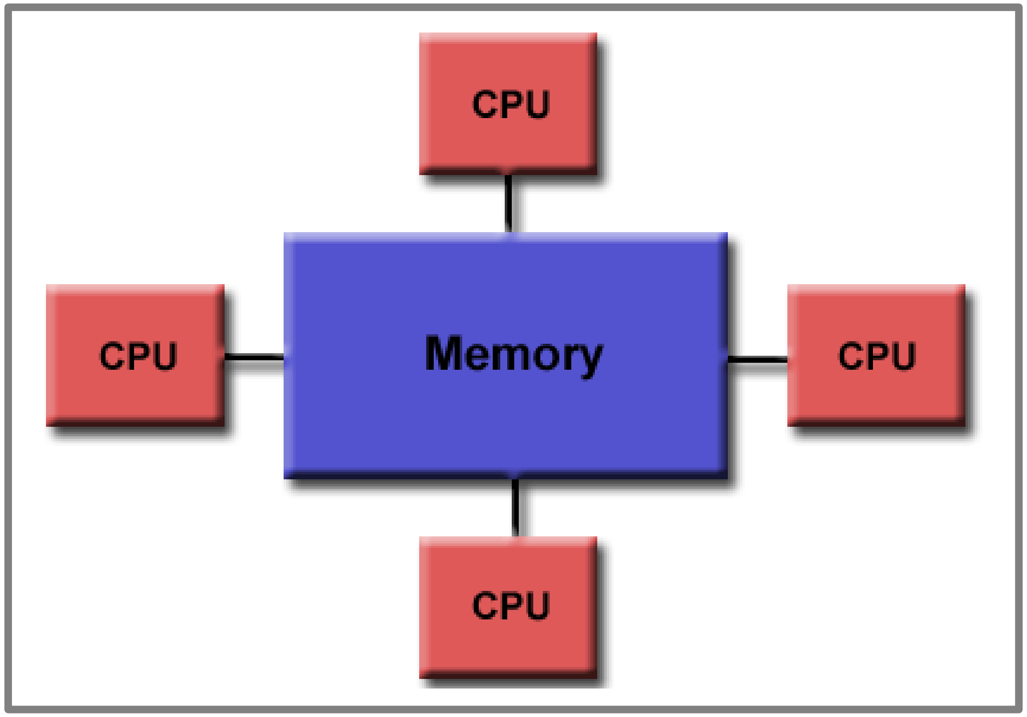

Superscalar
Multiprocessing
Hyper-Threading
Mulitprocessing mit Hyper-Threading
Zeit (Prozessorzyklen)
Funktionseinheit ungenutzt
Funktionseinheit bearbeitet Thread 1
Funktionseinheit bearbeitet Thread 2
Funktionseinheit bearbeitet Thread 3
Funktionseinheit bearbeitet Thread 4
© tecChannel.de

Bluegene/Q has 16 cores per node, each with 4 hardware threads, hence one should run 64 threads for full use.

Current multi-socket SMP nodes usually have a non-uniform memory access

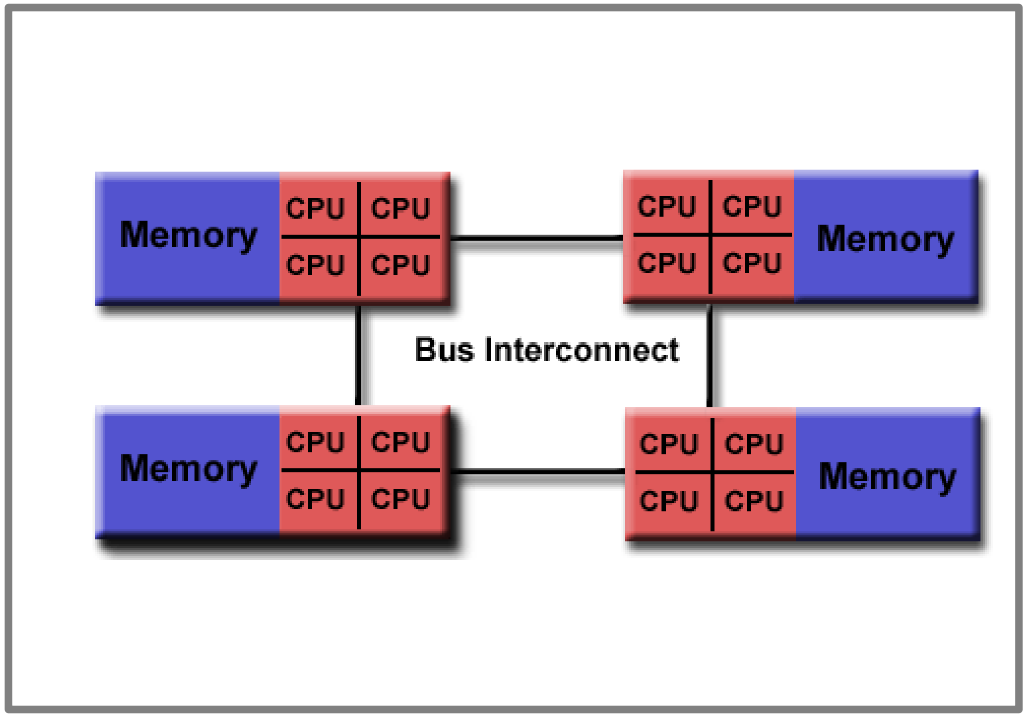
DIFFERENT MEMORY ARCHITECTURES

compute node



uniform memory access

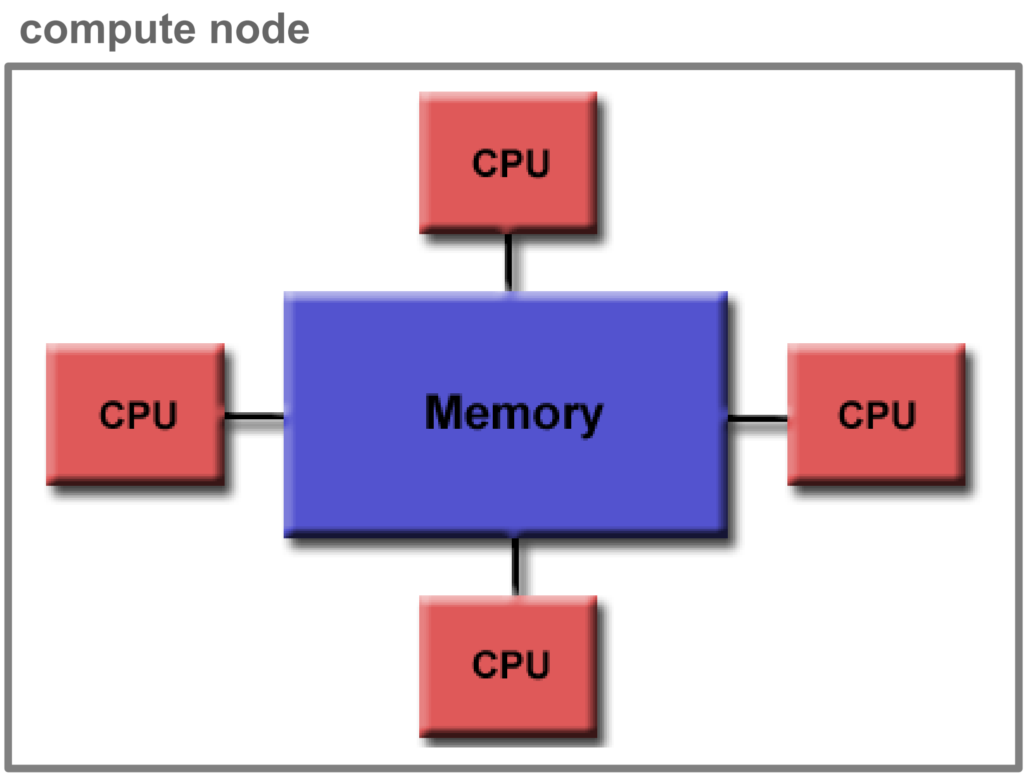
compute node



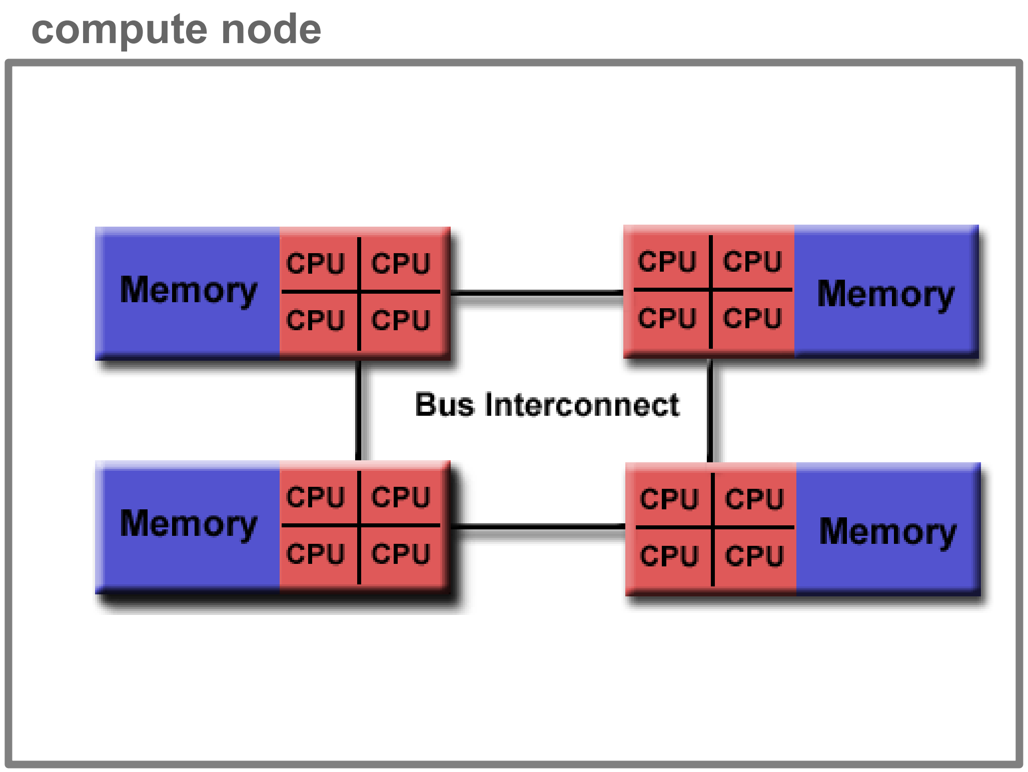
non-uniform memory access (NUMA)

Current multi-socket SMP nodes usually have a non-uniform memory access

DIFFERENT MEMORY ARCHITECTURES



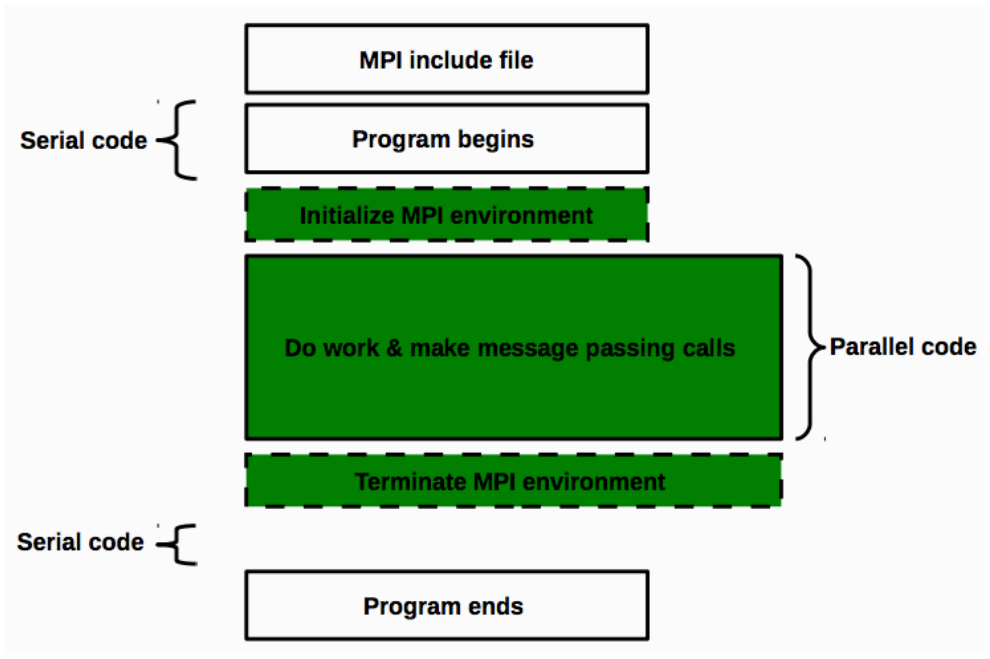
uniform memory access

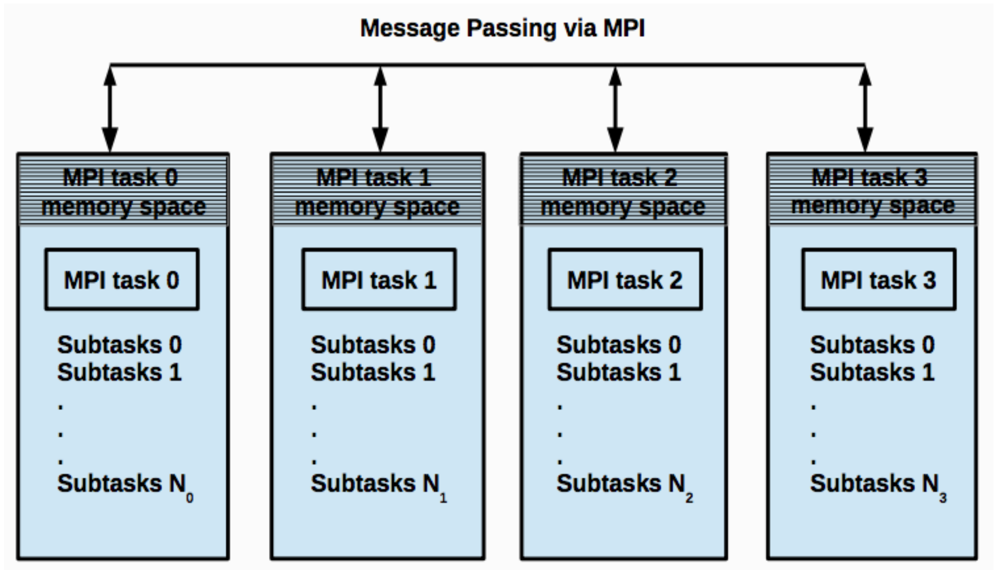


non-uniform memory access (NUMA)

Developing MPI programs often requires severe algorithmic changes

PROGRAMMING MODEL OF MPI


MPI include file
Serial code
Program begins
Initialize MPI environment
Do work & make message passing calls
Parallel code
Terminate MPI environment
Serial code
Program ends



- Data decomposition and all communication in the parallel code needs to be coded explicitly.
- Doing this well represents represents a steep learning curve.

MPI is well standardized and portable, but current MPI libraries face scaling challenges with respect to their memory need when all-to-all communication patters occur.

The ***shared memory*** of a compute nodes allow a simplified parallel programming model through multi-threading

MULTI-THREADING

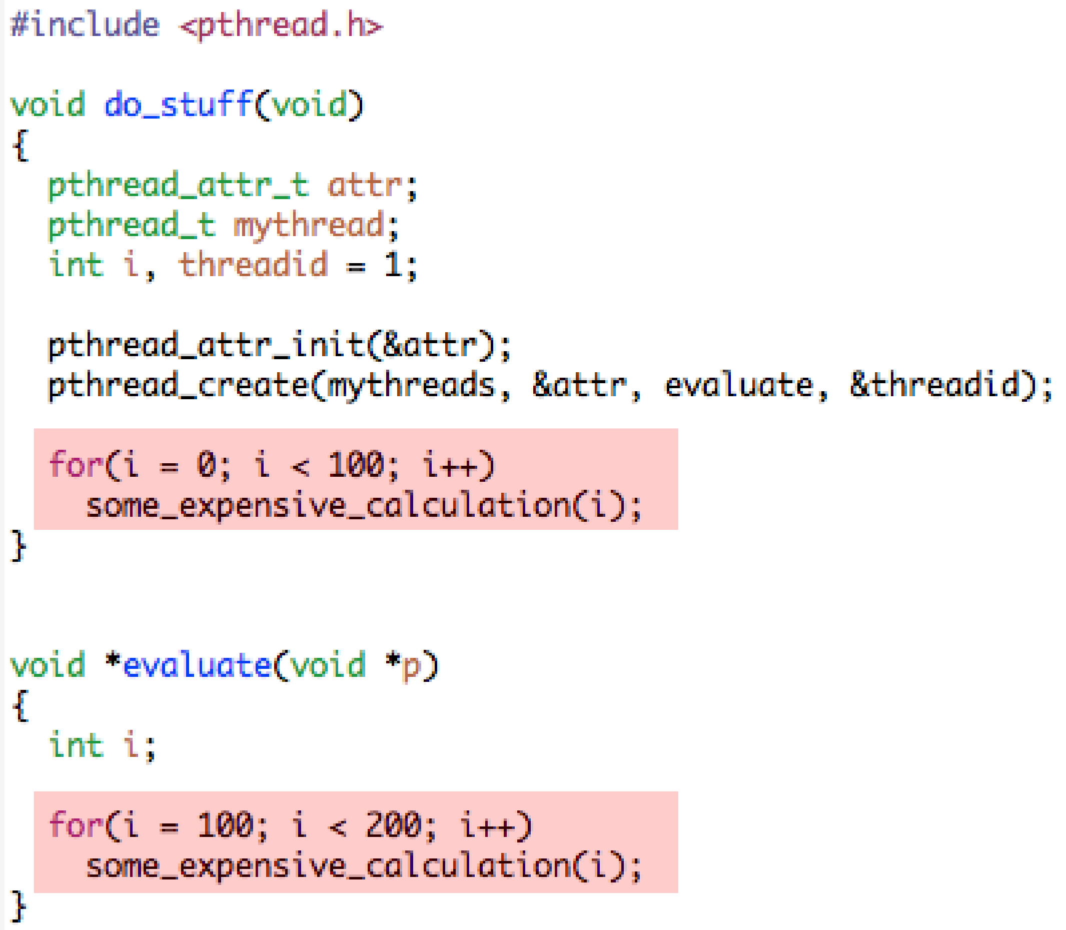
UNIX **processes are isolated** from each other - they have there own protected memory.

A process can however be split up into multiple execution paths, so-called **threads**, allowing lightweight parallelism where data-sharing is trivially achieved.

Threads can be created by the programmer explicitly through the **pthreads** system calls, or via **OpenMP**.

The two loops will be executed concurrently by two different threads on two different physical cores (if available)

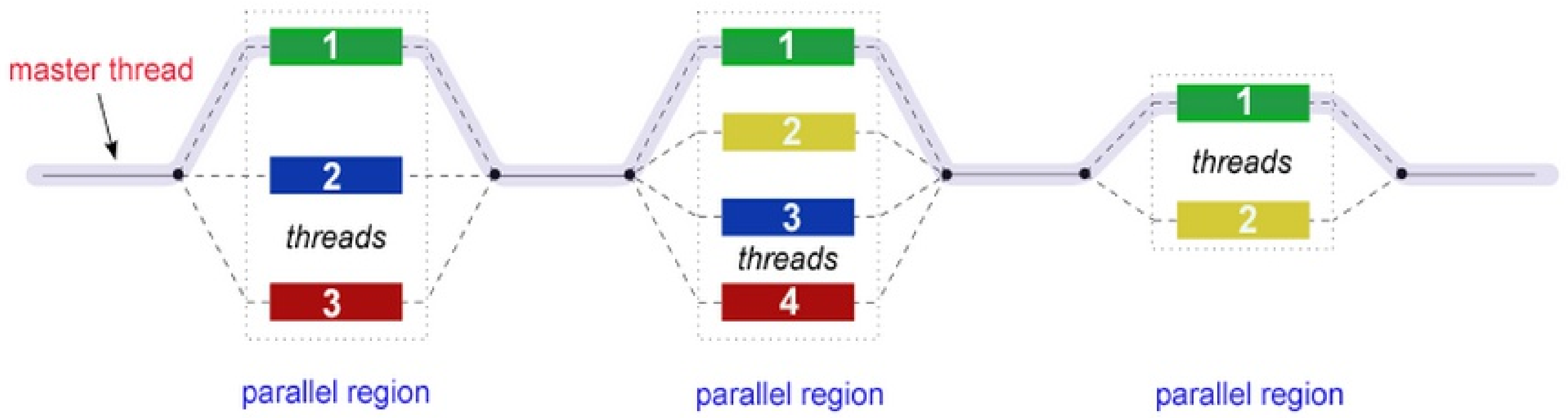
Example for pthreads



```
#include <pthread.h>

void do_stuff(void)
{
  pthread_attr_t attr;
  pthread_t mythread;
  int i, threadid = 1;

  pthread_attr_init(&attr);
  pthread_create(mythreads, &attr, evaluate, &threadid);

  for(i = 0; i < 100; i++)
    some_expensive_calculation(i);
}


void *evaluate(void *p)
{
  int i;

  for(i = 100; i < 200; i++)
    some_expensive_calculation(i);
}
```

In OpenMP, one can easily create and destroy threads through a simple language extension

THE FORK-JOIN PROGRAMMING MODEL OF OPENMP



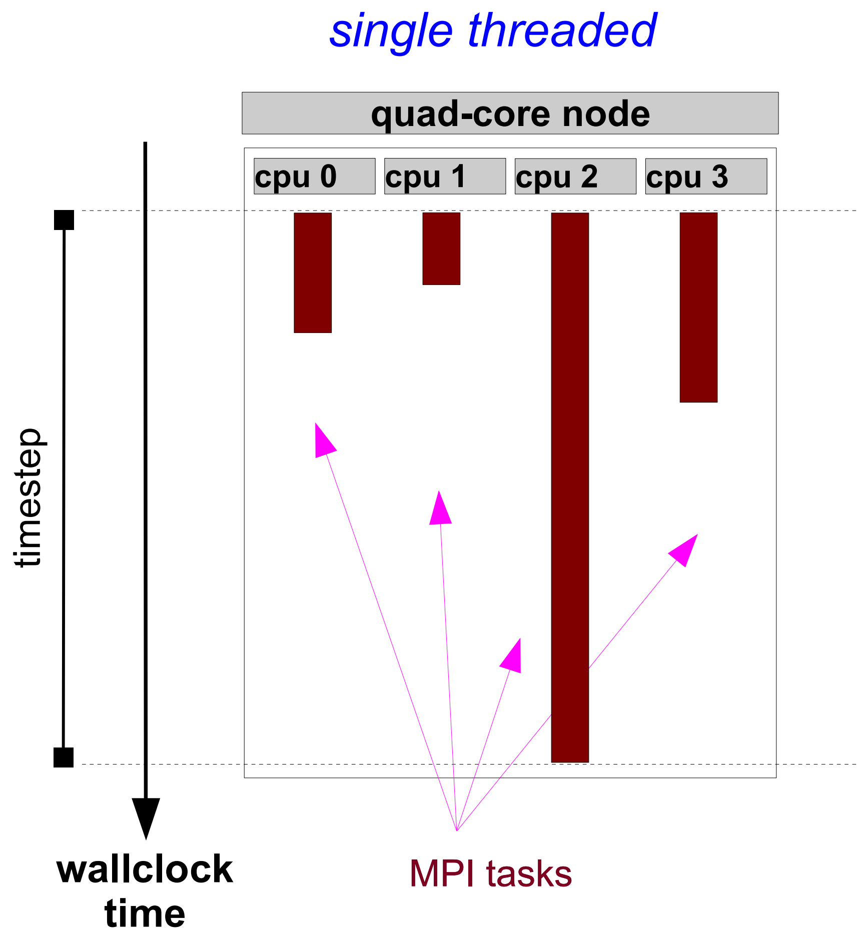
- Compilers exist for C/C++ and Fortran
- Allows easy parallelization of existing code
- Thanks to shared memory synchronization and data exchange, work-load imbalance in parallel sections can be avoided
- Reduction of memory overhead needed for ghost cells, bookkeeping data, etc.

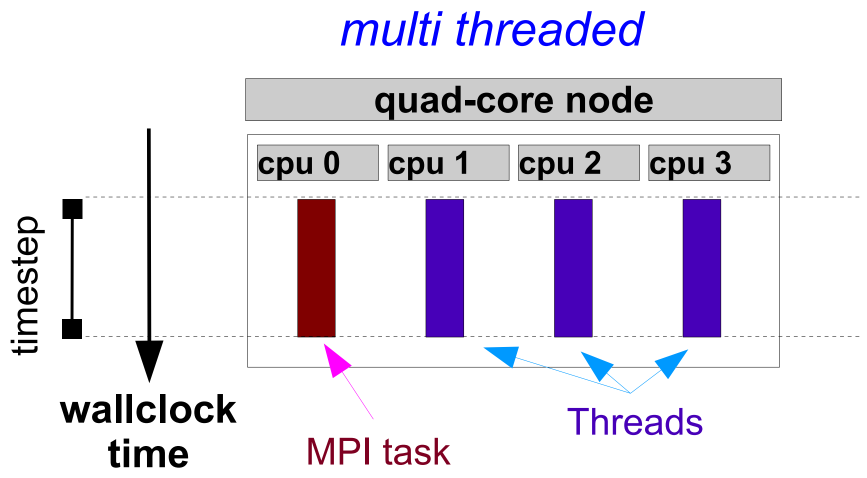
Example for OpenMP

```
#include <omp.h>

void do_stuff(void)
{
  int i;

#pragma omp parallel for
  for(i = 0; i < 200; i++)
    some_expensive_calculation(i);
}
```

Shared memory can be easily used for near perfect loop-level parallelism

USING MULTIPLE CORES WITH THREADS





- Threads are light-weight. Unlike processes, the creation/destruction takes almost no time.
- They inherit all global variables and resources (e.g. open file) from their parent process/thread.
- Mutual exclusion looks need to be used where needed to avoid race conditions.

How to get them?

- **POSIX/System-V Threads**
- **OpenMP**

Ordinary main-stream processors have acquired powerful vector extensions

THE SSE/AVX INSTRUCTIONS OF INTEL AND AMD PROCESSORS

AVX, introduced in Intel Sandy bridge, offers registers that are 256 bit wide.

With those, one can now do:

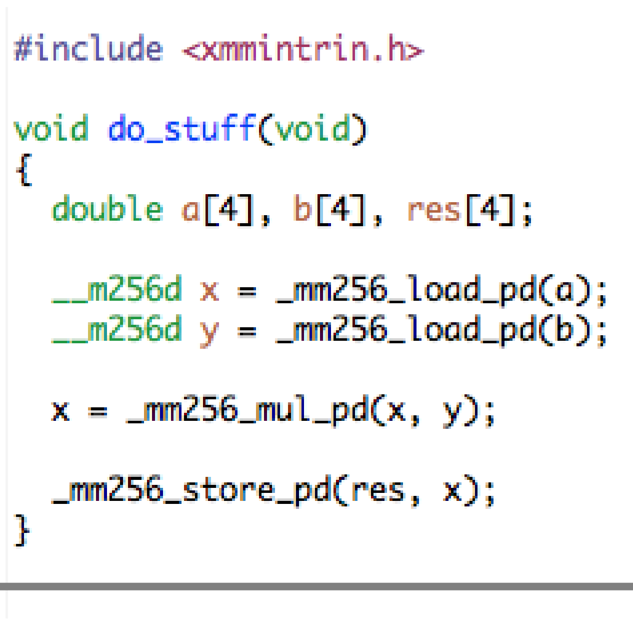
- **4 double precision calculations** in parallel, or
- **8 single precision calculations** in parallel

Essentially at the same time as the usual instructions take for a single operations.

If these instructions can be efficiently used, substantial speed ups are possible.

To use them, one either trusts that the compiler will use in an optimum way (wishful thinking), or one can program them using instrinsics....

vector intrinsics

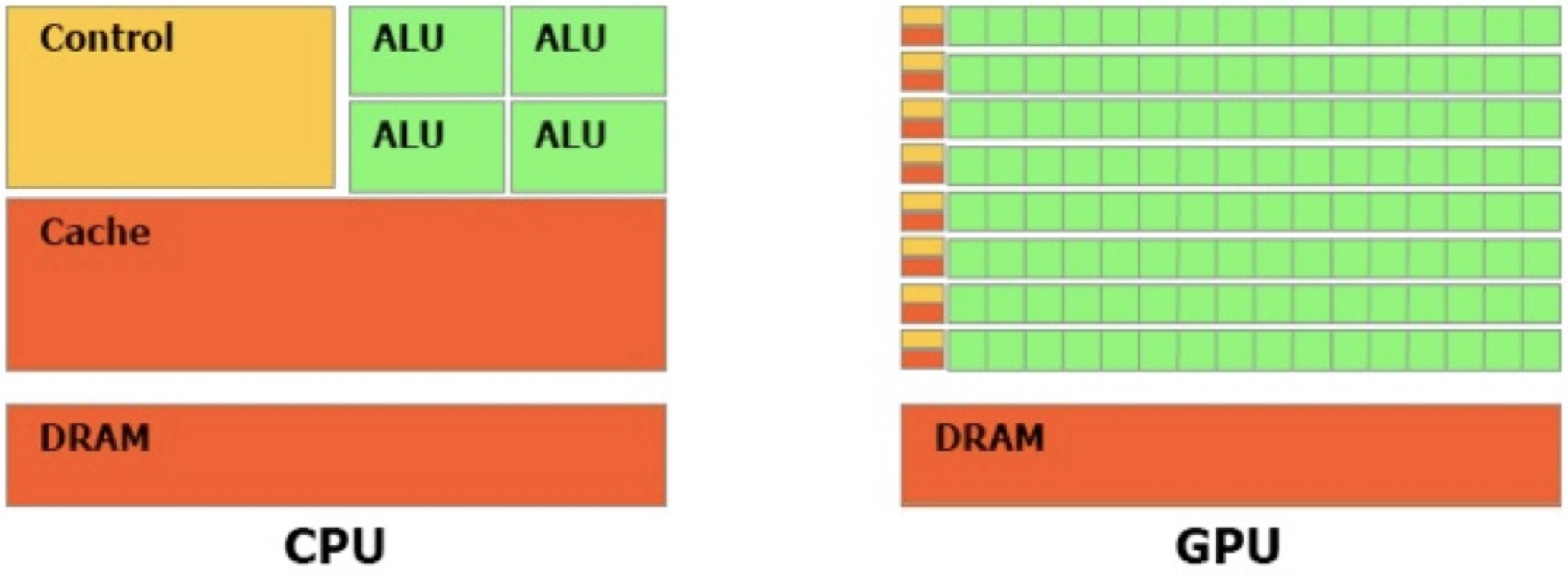


```
#include <xmmintrin.h>

void do_stuff(void)
{
  double a[4], b[4], res[4];

  __m256d x = _mm256_load_pd(a);
  __m256d y = _mm256_load_pd(b);

  x = _mm256_mul_pd(x, y);

  _mm256_store_pd(res, x);
}
```

In the Intel Xeon Phi Accelerator Cards, the width of these vector instructions has been doubled yet again to 512 bit.

Device computing is a new trend to exploit the extreme raw computational speed of GPUs or Xeon-Phi acclerator cards

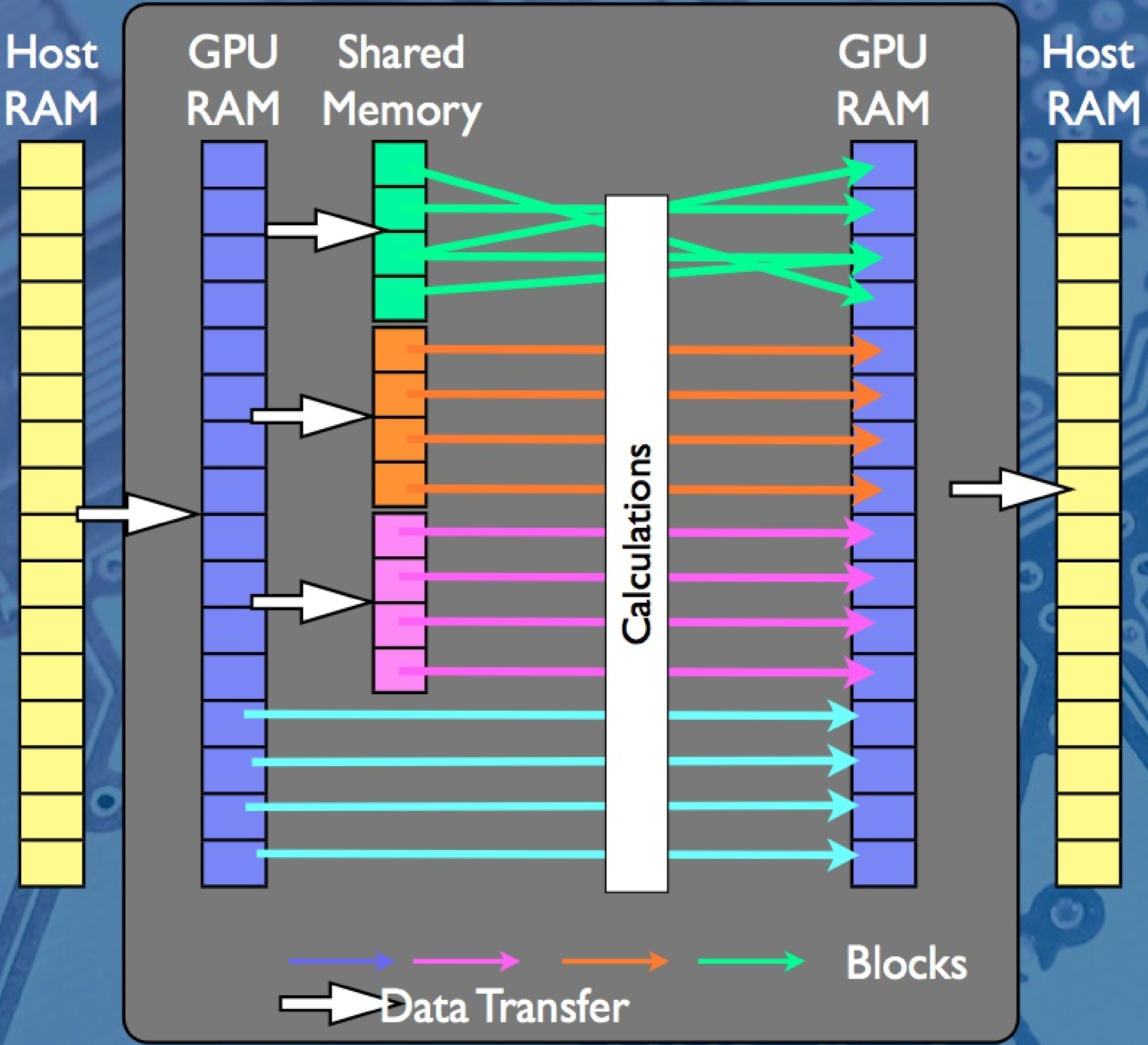
HETEROGENOUS COMPUTING



Ideally, all threads must execute the same code, otherwise full performance can't be reached. Branching needs to be minimized among the threads.

(figure by D. Aubert)

Principle of GPU programming with CUDA



<u>If possible:</u>
independent &
identical
threads
+
High arithmetic
intensity
=
acceleration

(D. Aubert)

High performance computing will offer unprecedented opportunities for astrophysics in the coming years... and a novel programming complexity

WHO ORDERED HYBRID COMPUTING?

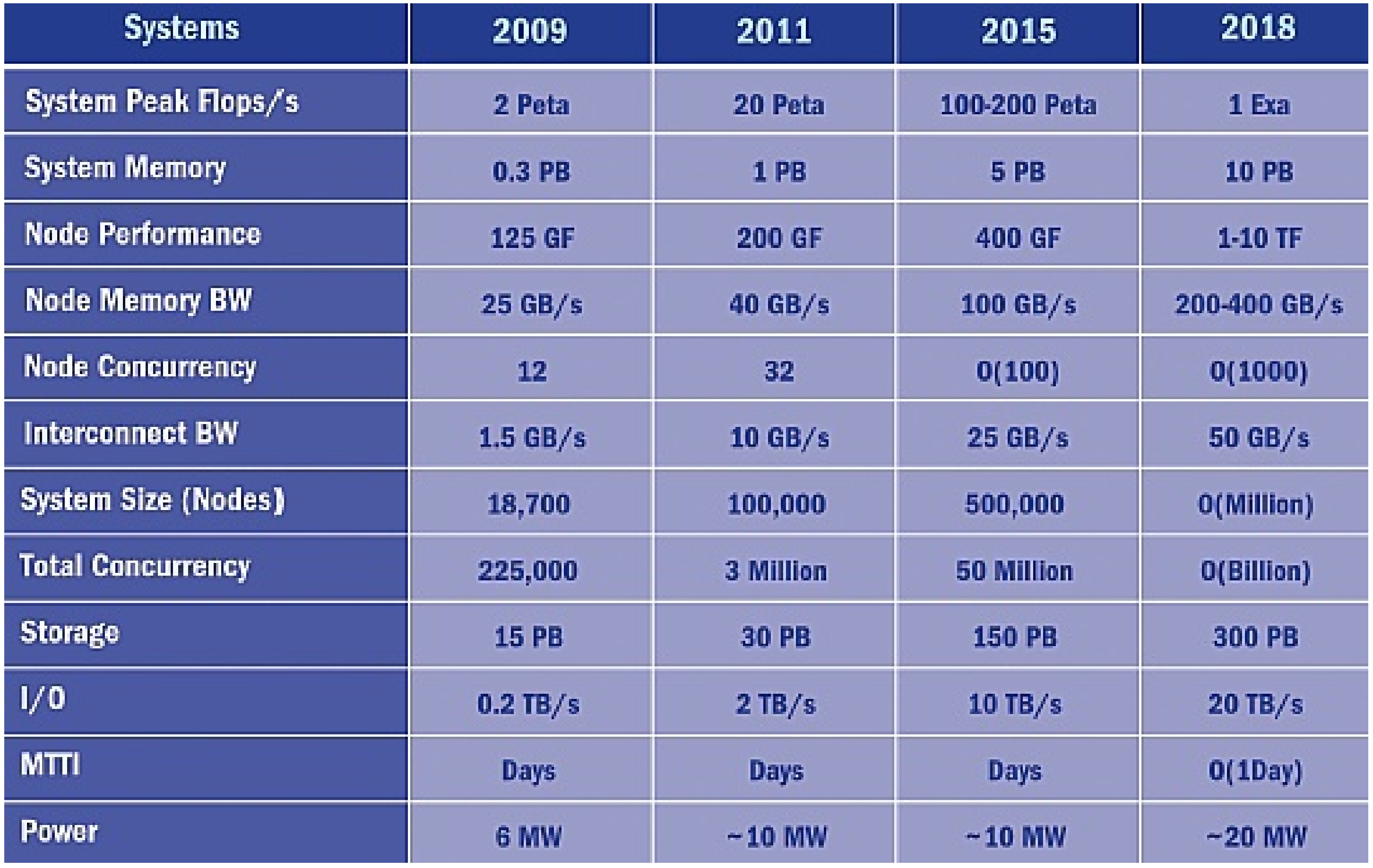
Parallelization through:

- MPI
- OpenMP
- GPU
- SSE/AVX

All of this needs to be combined!

We need MPI+OpenMP+GPU codes

Roadmap to Exascale



Systems	2009	2011	2015	2018
System Peak Flops/s	2 Peta	20 Peta	100-200 Peta	1 Exa
System Memory	0.3 PB	1 PB	5 PB	10 PB
Node Performance	125 GF	200 GF	400 GF	1-10 TF
Node Memory BW	25 GB/s	40 GB/s	100 GB/s	200-400 GB/s
Node Concurrency	12	32	O(100)	O(1000)
Interconnect BW	1.5 GB/s	10 GB/s	25 GB/s	50 GB/s
System Size (Nodes)	18,700	100,000	500,000	O(Million)
Total Concurrency	225,000	3 Million	50 Million	O(Billion)
Storage	15 PB	30 PB	150 PB	300 PB
I/O	0.2 TB/s	2 TB/s	10 TB/s	20 TB/s
MTTI	Days	Days	Days	O(1Day)
Power	6 MW	~10 MW	~10 MW	~20 MW

But do our problems really have a degree of parallelism that allows the use of a 1 billion threads?

It may be that such machines can only be used at scale by science disciplines with computationally simpler problems compared to what we have in galaxy formation....