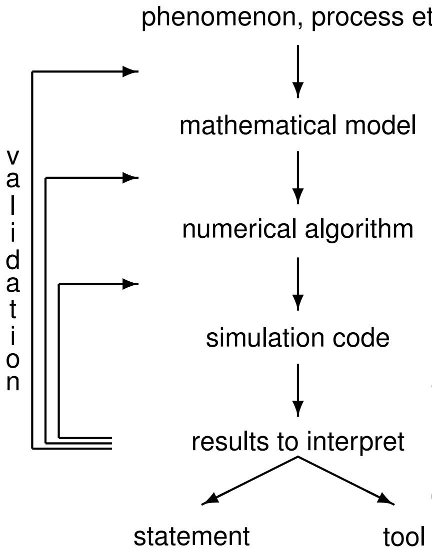
Gene Golub SIAM Summer School 2012 –
Simulation and Supercomputing in the Geosciences

Parallel and GPU Computing – Why Should I Bother?

Michael Bader, Alexander Breuer

Technische Universität München

The Simulation Pipeline



phenomenon, process etc.

mathematical model

numerical algorithm

simulation code

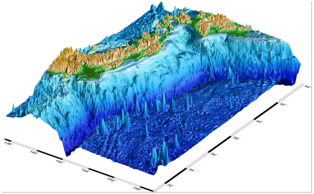
results to interpret

statement

tool

validation

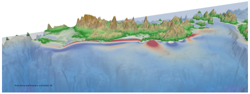
modelling



numerical treatment

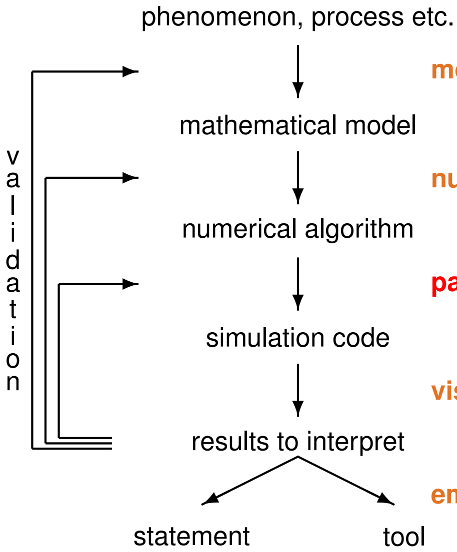
implementation

visualization



embedding

The Simulation Pipeline



phenomenon, process etc.

modelling

mathematical model

numerical treatment

numerical algorithm

parallel implementation

simulation code

visualization

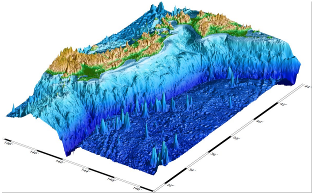
results to interpret

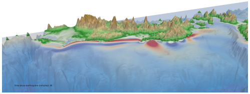
embedding

statement

tool

validation





Parallel Computing – “Faster, Bigger, More”

Why parallel high performance computing:

- **Response time:** compute a problem in $\frac{1}{p}$ time
  - speed up engineering processes
  - real-time simulations (tsunami warning?)
- **Problem size:** compute a p-times bigger problem
  - simulation of large-/multi-scale phenomena
  - maximal problem size that “fits into the machine”
  - validation of smaller, “operational” models
- **Throughput**: compute p problems at once
  - case and parameter studies, statistical risk scenarios, etc. (hazard maps, data base for tsunami warning, . . .)
  - massively distributed computing (SETI@home, e.g.)

Part I

High Performance Computing in CSE – Past(?) and Present Trends

Computational Science Demands a New Paradigm

Computational simulation must meet three challenges to become a mature partner of theory and experiment

(Post & Votta, 2005)

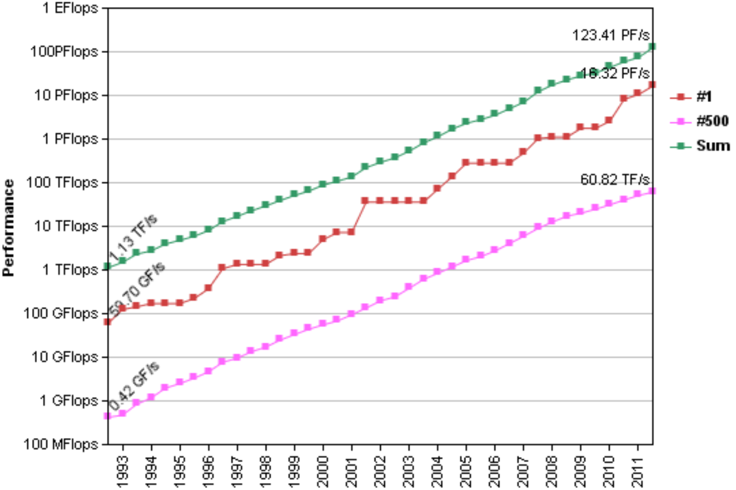
1. **performance challenge**
   → exponential growth of performance, massively parallel architectures
2. **programming challenge**
   → new (parallel) programming models
3. **prediction challenge**
   → careful verification and validation of codes; towards **reproducible** simulation experiments

Four Horizons for Enhancing the Performance . . .

. . . of Parallel Simulations Based on Partial Differential Equations
(David Keyes, 2000)

1. Expanded Number of Processors
   → in 2000: 1000 cores; in 2010: 200,000 cores
2. More Efficient Use of Faster Processors
   → PDF working-sets, cache efficiency
3. More Architecture-Friendly Algorithms
   → improve temporal/spatial locality
4. Algorithms Delivering More “Science per Flop”
   → adaptivity (in space and time), higher-order methods, fast solvers

Performance Development in Supercomputing



(source: www.top500.org)

Top 500 (www.top500.org) – June 2012

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom / 2011 IBM	1572864	16324.75	20132.66	7890.0
2	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	705024	10510.00	11280.38	12659.9
3	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom / 2012 IBM	786432	8162.38	10066.33	3945.0
4	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR / 2012 IBM	147456	2897.00	3185.05	3422.7
5	National Supercomputing Center in Tianjin China	**Tianhe-1A** - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 / 2010 NUDT	186368	2566.00	4701.00	4040.0
6	DOE/SC/Oak Ridge National Laboratory United States	**Jaguar** - Cray XK6, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA 2090 / 2009 Cray Inc.	298592	1941.00	2627.61	5142.0
7	CINECA Italy	**Fermi** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom / 2012 IBM	163840	1725.49	2097.15	821.9
8	Forschungszentrum Juelich (FZJ) Germany	**JuQUEEN** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom / 2012 IBM	131072	1380.39	1677.72	657.5

Top 500 Spotlights – Sequoia and K Computer

Sequoia → IBM BlueGene/Q (LLNL)

- 98,304 compute nodes; 1.6 mio cores(!)
- Linpack benchmark: 16.32 PFlop/s
- ≈ 8 MW power

Top 500 Spotlights – Sequoia and K Computer

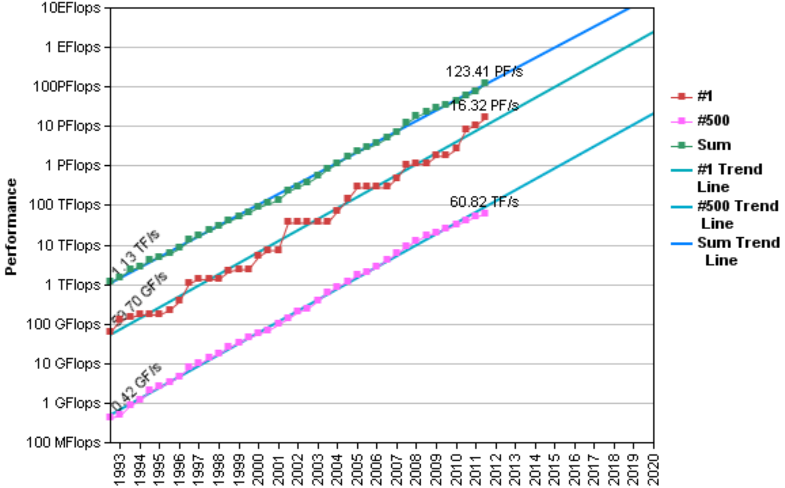
Sequoia → IBM BlueGene/Q (LLNL)

- 98,304 compute nodes; 1.6 mio cores(!)
- Linpack benchmark: 16.32 PFlop/s
- ≈ 8 MW power

K Computer → SPARC64 (RIKEN, Kobe)

- 98,304 compute nodes; 705,024 cores
- Linpack benchmark: 10.51 PFlop/s
- ≈ 12 MW power
- SPARC64 VIIIfx 2.0GHz (8-core CPU)

Performance Development in Supercomputing



(source: www.top500.org)

International Exascale Software Project Roadmap

Towards an Exa-Flop/s Platform in 2018 (www.exascale.org):

1. **technology trends**
   - → concurrency, reliability, power consumption, . . .
   - → blueprint of an exascale system: 10-billion-way concurrency, 100 million to 1 billion cores, 10-to-100-way concurrency per core, hundreds of cores per die, . . .
2. **science trends**
   - → climate, high-energy physics, nuclear physics, fusion energy sciences, materials science and chemistry, . . .
3. **X-stack** (software stack for exascale)
   - → energy, resiliency, heterogeneity, I/O and memory
4. **Polito-economic trends**
   - → exascale systems run by government labs, used by CSE scientists

Exascale Roadmap

“Aggressively Designed Strawman Architecture”

Level	What	Perform.	Power	RAM
FPU	FPU, regs,. instr.-memory	1.5 Gflops	30mW	
Core	4 FPUs, L1	6 Gflops	141mW	
Proc. Chip	**742 cores**, L2/L3, Interconn.	4.5 Tflops	214W	
Node	Proc. chip, DRAM	4.5 Tflops	230W	**16 GB**
Group	12 proc. chips, routers	54 Tflops	3.5KW	192 GB
rack	32 groups	1.7 Pflops	116KW	6.1 TB
System	**583 racks**	1 Eflops	67.7MW	3.6 PB

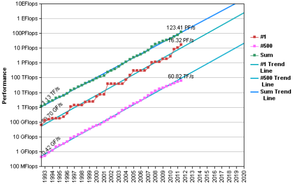
approx. 285,000 cores per rack; 166 mio cores in total

Source:

ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems

Exascale Roadmap – Should You Bother?

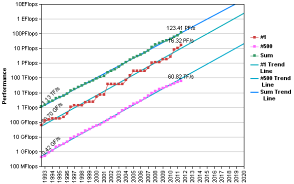
Your department's compute cluster in 5 years?



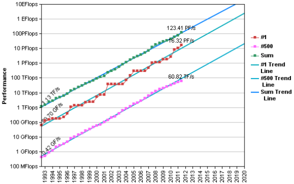
Exascale Roadmap – Should You Bother?

Your department's compute cluster in 5 years?

- a **Petaflop System**!



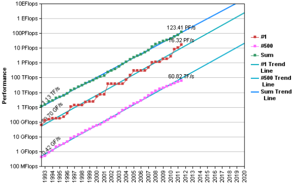
Exascale Roadmap – Should You Bother?



Your department's compute cluster in 5 years?

- a **Petaflop System**!
- "one rack of the Exaflop system"
  → using the same/similar hardware

Exascale Roadmap – Should You Bother?



Your department's compute cluster in 5 years?

- a **Petaflop System**!
- "one rack of the Exaflop system"
  → using the same/similar hardware
- extrapolated example machine:
  - peak performance: 1.7 PFlop/s
  - 6 TB RAM, 60 GB cache memory
  - "total concurrency": $1.1 \cdot 10^6$
  - number of cores: $280,000$
  - number of chips: 384

Source: ExaScale Software Study: Software Challenges in Extreme Scale Systems

Your Department's PetaFlop/s Cluster in 5 Years?

Tianhe-1A (Tianjin, China; Top500 # 5)

- # 5 in Top 500 (July 2012)
- 14,336 Xeon X5670 CPUs
- **7,168 Nvidia Tesla M2050 GPUs**
- Linpack benchmark: 2.56 PFlop/s
- ≈ 4 MW power

Your Department’s PetaFlop/s Cluster in 5 Years?

Tianhe-1A (Tianjin, China; Top500 # 5)

- # 5 in Top 500 (July 2012)
- 14,336 Xeon X5670 CPUs
- **7,168 Nvidia Tesla M2050 GPUs**
- Linpack benchmark: 2.56 PFlop/s
- $\approx$ 4 MW power

Discovery (Intel, Top500 # 150)

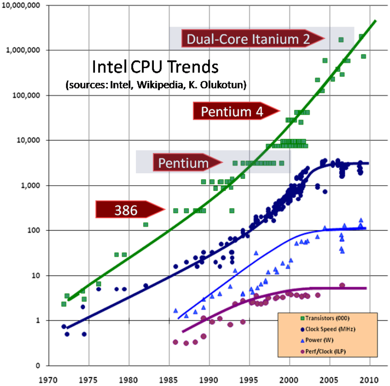
- 9800 cores (Xeon and MIC/“many integrated cores”)
- Linpack benchmark: 118 TFlop/s
- **Knights Corner** / Intel Xeon Phi / Intel MIC as accelerator
- “$>$50 cores”, roughly 1.1–1.3 GHz
- wider vector FP units: 64 bytes (i.e., 16 floats, 8 doubles)

Free Lunch is Over$^{(*)}$

. . . actually already over for quite some time!

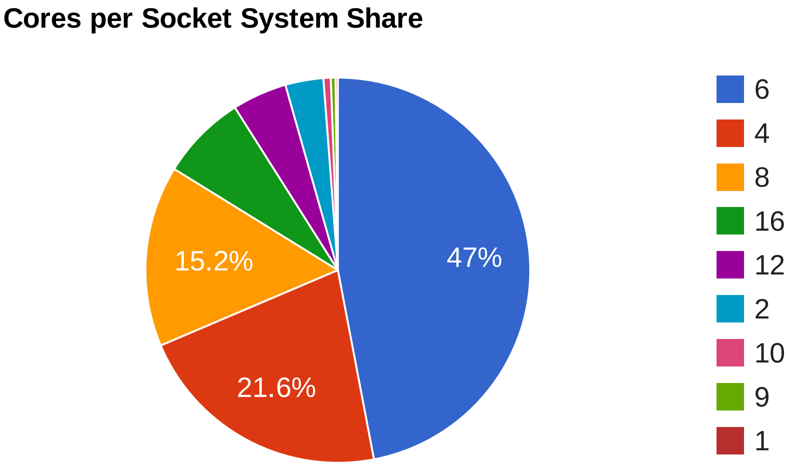
Speedup of your software can only come from parallelism:

- clock speed of CPU has stalled
- instruction-level parallelism per core has stalled
- number of cores is growing
- size of vector units is growing

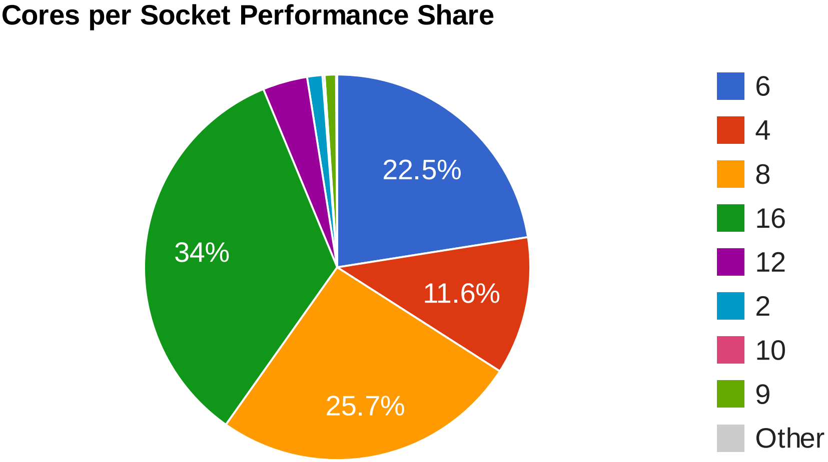


$^{(*)}$ Quote and image taken from: H. Sutter, *The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software*, Dr. Dobb's Journal 30(3), March 2005.

Top 500 (www.top500.org) – # Cores



Top 500 (www.top500.org) – # Cores



Parallel Computing Paradigms

Not exactly sure how the hardware will look like . . .
(CPU-style, GPU-style, something new?)

However: **massively parallel programming** required

- revival of vector computing
  → several/many FPUs performing the same operation
- hybrid/heterogenous achitectures
  → different kind of cores; dedicated accelerator hardware
- different access to memory
  → cache and cache coherency
  → small amount of memory per core
- our concern in this course: **novel programming paradigms**
  → look into GPU computing, as an example

The Seven Dwarfs of HPC

“dwarfs” = key algorithmic kernels in many scientific computing applications

P. Colella (LBNL), 2004:

1. dense linear algebra
2. sparse linear algebra
3. spectral methods
4. N-body methods
5. **structured grids**
6. **unstructured grids**
7. Monte Carlo

The Seven Dwarfs of HPC

“dwarfs” = key algorithmic kernels in many scientific computing applications

P. Colella (LBNL), 2004:

1. dense linear algebra
2. sparse linear algebra
3. spectral methods
4. N-body methods
5. **structured grids**
6. **unstructured grids**
7. Monte Carlo

Tsunami & storm-surge simulation:

→ usually PDE solvers on structured or unstructured meshes

The Seven Dwarfs of HPC

“dwarfs” = key algorithmic kernels in many scientific computing applications

P. Colella (LBNL), 2004:

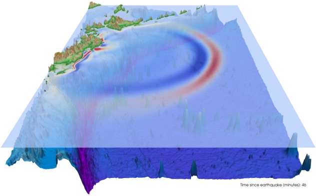
1. dense linear algebra
2. sparse linear algebra
3. spectral methods
4. N-body methods
5. **structured grids**
6. **unstructured grids**
7. Monte Carlo

Tsunami & storm-surge simulation:

→ usually PDE solvers on structured or unstructured meshes

→ **SWE**: a simple shallow water solver on Cartesian grids

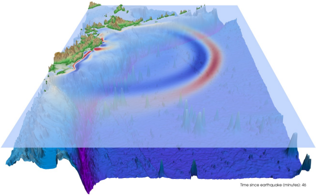
Roadmap for the Course



SWE:

1. overview on model and code
2. parallelization perspectives
3. software design for hybrid parallelism

Roadmap for the Course



SWE:

1. overview on model and code
2. parallelization perspectives
3. software design for hybrid parallelism

GPU Computing:

1. crash course on CUDA
2. first goal: “run everything on the GPU”
3. possibilities for optimization
4. SWE on a GPU cluster