Exascale Computing and Big Data

Philipp Neumann

Deutsches Klimarechenzentrum

2016-11-18

What is Exascale?

What is Exascale? Asking Google...

wikipedia:

Exascale computing refers to computing systems capable of at least one exaFLOPS, or a billion billion calculations per second. Such capacity represents a thousandfold increase over the first petascale computer that came into operation in 2008. (One exaflops is a thousand petaflops or a quintillion, 10^{18}, floating point operations per second.) At a supercomputing conference in 2009, Computerworld projected exascale implementation by 2018.

deutschlandfunk.de, 18 July 2015

Stromverbrauch würde eine Milliarde kosten
Heutige Supercomputer können mehrere Billiarden Rechenoperationen gleichzeitig ausführen. Das scheint gigantisch, Forscher denken aber schon über den nächsten Schritt nach, den Exa-Flops-Rechner: der könnte eine Trillion Rechenoperationen in der Sekunde ausführen – wenn da nicht der Stromverbrauch wäre.

zdnet.de, 30 July 2015

US-Präsident ordnet Entwicklung von Exascale-Supercomputer an

What is Exascale? Asking Google...

wikipedia:

> Exascale computing refers to computing systems capable of at least one exaFLOPS, or a billion billion calculations per second. Such capacity represents a thousandfold increase over the first petascale computer that came into operation in 2008. (One exaflops is a thousand petaflops or a quintillion, 10^{18}, floating point operations per second.) At a supercomputing conference in 2009, Computerworld projected exascale implementation by 2018.

Remark: Exascale has been postponed to 2022 or so ;-)

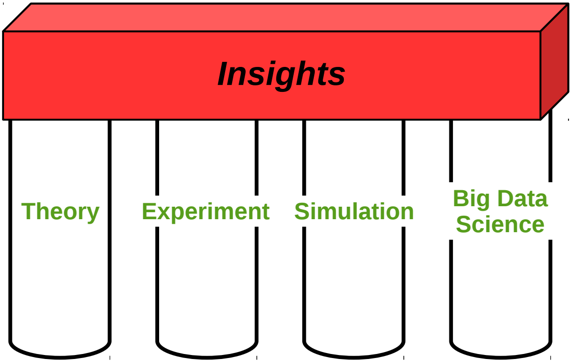
deutschlandfunk.de, 18 July 2015

> **Stromverbrauch würde eine Milliarde kosten**
> Heutige Supercomputer können mehrere Billiarden Rechenoperationen gleichzeitig ausführen. Das scheint gigantisch, Forscher denken aber schon über den nächsten Schritt nach, den Exa-Flops-Rechner: der könnte eine Trillion Rechenoperationen in der Sekunde ausführen – wenn da nicht der Stromverbrauch wäre.

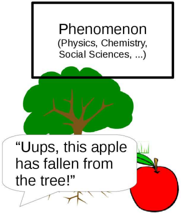
zdnet.de, 30 July 2015

> **US-Präsident ordnet Entwicklung von Exascale-Supercomputer an**

The ~~3~~ 4 Pillars of Science



The Simulation Pipeline



The Simulation Pipeline

Phenomenon
(Physics, Chemistry, Social Sciences, ...)

→

Mathematical Model

$$\frac{dx(t)}{dt} = v(t), x(0) = 0$$

$$\frac{dv(t)}{dt} = -g, v(0) = 0$$

The Simulation Pipeline

Phenomenon (Physics, Chemistry, Social Sciences, ...) → Mathematical Model → Numerical Algorithm

$$\frac{dx(t)}{dt} = v(t),\ x(0) = 0$$
$$\frac{dv(t)}{dt} = -g,\ v(0) = 0$$

$$x(t+\Delta t) = x(t) + \Delta t\, v(t) - \frac{1}{2}\Delta t^2 g$$
$$v(t+\Delta t) = v(t) - \Delta t\, g$$

The Simulation Pipeline

Phenomenon
(Physics, Chemistry, Social Sciences, ...)

→ Mathematical Model

$$\frac{dx(t)}{dt} = v(t),\ x(0) = 0$$

$$\frac{dv(t)}{dt} = -g,\ v(0) = 0$$

→ Numerical Algorithm

$$x(t+\Delta t) = x(t) + \Delta t\, v(t) - \frac{1}{2}\Delta t^2 g$$

$$v(t+\Delta t) = v(t) - \Delta t\, g$$

↓ Implementation

```
double x = 0.0;
double v = 0.0;
const double g = -9.81;
const double dt = 0.01;
const double endtime = 0.5;
double t = 0.0;

while (t<endtime){
  x = x + dt*v-0.5*dt*dt*g;
  v = v - dt*g;
  t = t + dt;
}
```

The Simulation Pipeline

Phenomenon
(Physics, Chemistry, Social Sciences, ...)

→ Mathematical Model

$$\frac{dx(t)}{dt} = v(t),\ x(0) = 0$$
$$\frac{dv(t)}{dt} = -g,\ v(0) = 0$$

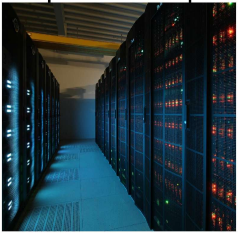
→ Numerical Algorithm

$$x(t+\Delta t) = x(t) + \Delta t\, v(t) - \frac{1}{2}\Delta t^2 g$$
$$v(t+\Delta t) = v(t) - \Delta t\, g$$

↓ Implementation

```
double x = 0.0;
double v = 0.0;
const double g = -9.81;
const double dt = 0.01;
const double endtime = 0.5;
double t = 0.0;

while (t<endtime){
  x = x + dt*v-0.5*dt*dt*g;
  v = v - dt*g;
  t = t + dt;
}
```

← Execution



Picture Execution: Mistral, dkrz.de

The Simulation Pipeline

Phenomenon
(Physics, Chemistry, Social Sciences, ...)

Mathematical Model

$$\frac{dx(t)}{dt} = v(t),\ x(0) = 0$$
$$\frac{dv(t)}{dt} = -g,\ v(0) = 0$$

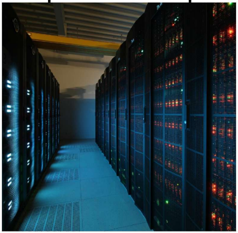
Numerical Algorithm

$$x(t+\Delta t) = x(t) + \Delta t\, v(t) - \frac{1}{2}\Delta t^2 g$$
$$v(t+\Delta t) = v(t) - \Delta t\, g$$

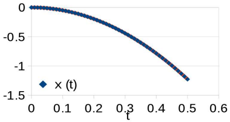
Implementation

```
double x = 0.0;
double v = 0.0;
const double g = -9.81;
const double dt = 0.01;
const double endtime = 0.5;
double t = 0.0;

while (t<endtime){
  x = x + dt*v-0.5*dt*dt*g;
  v = v - dt*g;
  t = t + dt;
}
```

Execution



Interpretation/ Visualization



Picture Execution: Mistral, dkrz.de

The Simulation ~~Pipeline~~Circle

Phenomenon
(Physics, Chemistry, Social Sciences, ...)

→ Mathematical Model

$$\frac{dx(t)}{dt}=v(t),\ x(0)=0$$
$$\frac{dv(t)}{dt}=-g,\ v(0)=0$$

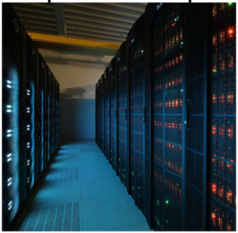
→ Numerical Algorithm

$$x(t+\Delta t)=x(t)+\Delta t\, v(t)-\frac{1}{2}\Delta t^2 g$$
$$v(t+\Delta t)=v(t)-\Delta t\, g$$

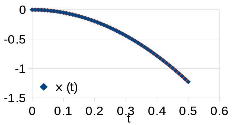
↓ Implementation

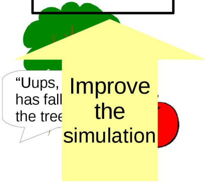
```
double x = 0.0;
double v = 0.0;
const double g = -9.81;
const double dt = 0.01;
const double endtime = 0.5;
double t = 0.0;

while (t<endtime){
  x = x + dt*v-0.5*dt*dt*g;
  v = v - dt*g;
  t = t + dt;
}
```

← Execution



← Interpretation/ Visualization

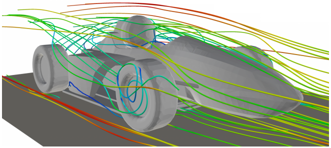




Picture Execution: Mistral, dkrz.de

Simulation, Supercomputing and Big Data (1)

Why do we want to have fast code?



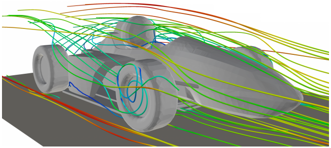
openlb.net

Consider the flow around a car

- Size of virtual wind tunnel: 20x10x10m
- Resolution of car: 1mm, resolution in time: $1 \cdot 10^{-5}$s
- per resolution cell: compute pressure, flow velocity (4 unknowns)
- Assumption: 1 floating point operation (FLOP) per unknown

Simulation, Supercomputing and Big Data (1)

Why do we want to have fast code?



openlb.net

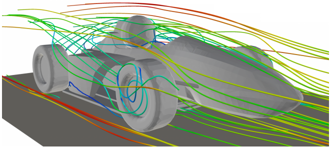
Consider the flow around a car

- Size of virtual wind tunnel: 20x10x10m
- Resolution of car: 1mm, resolution in time: $1 \cdot 10^{-5}$s
- per resolution cell: compute pressure, flow velocity (4 unknowns)
- Assumption: 1 floating point operation (FLOP) per unknown

$\rightarrow$ even for a perfect (=of linear complexity) solver, we require $8.0 \cdot 10^{12}$ operations per time step and $8.0 \cdot 10^{17}$ operations for one real-time second!

Simulation, Supercomputing and Big Data (2)

Why do we want to have fast code?



openlb.net

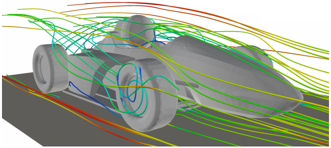
What does this mean for our simulation time?

Assumptions:

- No bottlenecks (except for limited clock speed) in our code (i.e. perfect memory access/prefetching, no memory latencies etc.)
- Using an Intel i7-3537U@2.0GHz
- 1 compute cycle per FLOP

Simulation, Supercomputing and Big Data (2)

Why do we want to have fast code?



openlb.net

What does this mean for our simulation time?

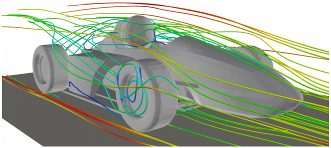
Assumptions:

- No bottlenecks (except for limited clock speed) in our code (i.e. perfect memory access/prefetching, no memory latencies etc.)
- Using an Intel i7-3537U@2.0GHz
- 1 compute cycle per FLOP

$\rightarrow$ we would require $\approx$ 12 years to solve this problem on a single core...

Simulation, Supercomputing and Big Data (2)

Why do we want to have fast code?



openlb.net

What does this mean for our simulation time?

Assumptions:

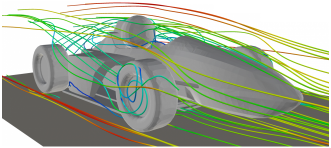
- No bottlenecks (except for limited clock speed) in our code (i.e. perfect memory access/prefetching, no memory latencies etc.)
- Using an Intel i7-3537U@2.0GHz
- 1 compute cycle per FLOP

$\rightarrow$ we would require $\approx$ 12 years to solve this problem on a single core...

...if we had a huge main memory to fit our 64TB of data in it!

Simulation, Supercomputing and Big Data (2)

Why do we want to have fast code?



openlb.net

Reality:

- Most codes far away from peak performance
- Complex physics/application, yielding non-trivial algorithms

What does this mean for our simulation time?

Assumptions:

- No bottlenecks (except for limited clock speed) in our code (i.e. perfect memory access/prefetching, no memory latencies etc.)
- Using an Intel i7-3537U@2.0GHz
- 1 compute cycle per FLOP

$\rightarrow$ we would require $\approx$ 12 years to solve this problem on a single core...

...if we had a huge main memory to fit our 64TB of data in it!

Computational Perspective: What We Need...

- ...are efficient algorithms (e.g., low complexity $O(N)$, $O(N \log N)$)
  $\rightarrow$ multigrid solvers, fast multipole methods,
  adaptive and multiscale methods, ...
- ...are algorithms which can be implemented efficiently
  $\rightarrow$ node-level optimization, shared-/distributed-memory
  parallelization, ...
- ...are efficient data structures
  $\rightarrow$ structure-of-arrays vs. array-of-structures,
  cache-efficient storage, ...
- ...is a good understanding of code, instructions and bottlenecks
  $\rightarrow$ vector instructions, memory vs. compute bound code, ...
- ...is a measure for expected performance
  $\rightarrow$ performance models (e.g., roofline)
- ...is knowledge of our hardware and its development
  $\rightarrow$ CPU, GPU, Xeon Phi, ...

Towards Exascale Hardware

Excerpt from Top 500 (November 2016)

rank	Name	Country	Cores	Perf. (Peta-FLOPS)	Power (MW)	Type
1	Sunway TaihuLight	China	10,649,600	93	15	Sunway26010
2	Tianhe-2	China	3,120,000	34	18	Intel Xeon/Xeon Phi
3	Titan	USA	560,000	18	8	Opteron/NVidia Kepler
4	Sequoia	USA	1,572,864	17	8	IBM Power BQC
5	Cori	USA	622,336	14	4	Intel Xeon Phi

$\rightarrow$ increase in core counts

$\rightarrow$ energy consumption is an issue
Extrapolating machines to exascale:
Sunway TaihuLight (2016): 161 MW $\leftrightarrow$ 12% of nuclear power plant*
Tianhe-2 (2013): 529 MW $\leftrightarrow$ 34% of nuclear power plant

$\rightarrow$ trend towards (hybrid) manycore architectures

* baseline for nuclear power plant: 1400 MW

Exascale challenge #1: Energy Consumption

Consequences:

- Manycore architectures to reduce energy consumption

> Seymour Cray:
> *"If you were plowing a field, which would you rather use: Two strong oxen or 1024 chickens? "*

$\rightarrow$ changes in programming/software design, e.g.
hybrid shared/distributed memory programming

Exascale challenge #1: Energy Consumption

Consequences:

- Manycore architectures to reduce energy consumption

> Seymour Cray:
> *"If you were plowing a field, which would you rather use:*
> *Two strong oxen or 1024 chickens? "*

$\rightarrow$ changes in programming/software design, e.g.
hybrid shared/distributed memory programming

- Enhanced programming to achieve energy efficient simulations
  $\rightarrow$ FLOPS are for free!
  Counting operations of an algorithm doesn't help anymore...
  $\rightarrow$ avoidance of memory transfer at all levels

Exascale challenge #1: Energy Consumption

Consequences:

- Manycore architectures to reduce energy consumption

> Seymour Cray:
> *"If you were plowing a field, which would you rather use: Two strong oxen or 1024 chickens? "*

$\rightarrow$ changes in programming/software design, e.g. hybrid shared/distributed memory programming

- Enhanced programming to achieve energy efficient simulations

$\rightarrow$ FLOPS are for free!
Counting operations of an algorithm doesn't help anymore...
$\rightarrow$ avoidance of memory transfer at all levels

Question: (pair work)

> Do we have to re-consider the complexity argument for fast algorithms?

Exascale challenge #1: Energy Consumption

Question:

Do we have to re-consider the complexity argument for fast algorithms?

- Only algorithms of $O(N)$ and $O(N \log N)$ suited at large scale
- However: local/sub-algorithmic parts may need to be revised
  $\rightarrow$ algorithm 1: compute bound, $O(N^2)$
  $\rightarrow$ algorithm 2: memory bound, $O(N)$
  $\rightarrow$ Which of them wins for $N = \ldots$?

The Simulation Pipeline Revisited

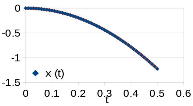
Phenomenon (Physics, Chemistry, Social Sciences, ...) → **Mathematical Model** → **Numerical Algorithm**

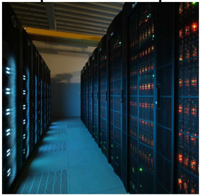
$$\frac{dx(t)}{dt} = v(t),\; x(0) = 0$$
$$\frac{dv(t)}{dt} = -g,\; v(0) = 0$$

$$x(t+\Delta t) = x(t) + \Delta t\, v(t) - \frac{1}{2}\Delta t^2 g$$
$$v(t+\Delta t) = v(t) - \Delta t\, g$$

↓

Interpretation/ Visualization ← **Execution** ← **Implementation**





```
double x = 0.0;
double v = 0.0;
const double g = -9.81;
const double dt = 0.01;
const double endtime = 0.5;
double t = 0.0;

while (t<endtime){
  x = x + dt*v-0.5*dt*dt*g;
  v = v - dt*g;
  t = t + dt;
}
```

Question:

Memory transfers are expensive. In which step would you expect a particular bottleneck?

The Simulation Pipeline Revisited

Phenomenon
(Physics, Chemistry, Social Sciences, ...)

→ **Mathematical Model**

$$\frac{dx(t)}{dt}=v(t), x(0)=0$$
$$\frac{dv(t)}{dt}=-g, v(0)=0$$

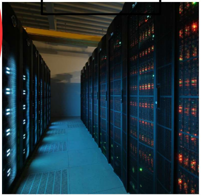
→ **Numerical Algorithm**

$$x(t+\Delta t)=x(t)+\Delta t\, v(t)-\frac{1}{2}\Delta t^2 g$$
$$v(t+\Delta t)=v(t)-\Delta t\, g$$

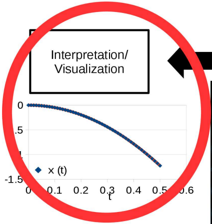
↓ **Implementation**

```
double x = 0.0;
double v = 0.0;
const double g = -9.81;
const double dt = 0.01;
const double endtime = 0.5;
double t = 0.0;

while (t<endtime){
  x = x + dt*v-0.5*dt*dt*g;
  v = v - dt*g;
  t = t + dt;
}
```

← **Execution**



← **Interpretation/ Visualization**



Question:

Memory transfers are expensive. In which step would you expect a particular bottleneck?

Simulation Data at Extreme Scale: Examples (1)

Example: Cosmic Structure Formation

Alimi et al., First-Ever Full Observable Universe Simulation, 2012:

- Particle simulation

$$\frac{d\vec{r}_i}{dt} = \vec{v}_i,$$

$$\frac{d\vec{v}_i}{dt} = -\nabla_r \phi \text{ with } \Delta_r \phi = 4\pi G\rho$$

- 550 000 000 000 particles (550 billion)
- 4752 compute nodes of supercomputer CURIE (76k cores)
- Data generated: 50 PBytes
- Data stored after reduction workflow: 500 TBytes

Simulation Data at Extreme Scale: Examples (2)

Example: Stokes Flow Simulations

Gmeiner et al., A quantitative performance study for Stokes solvers at the extreme scale, 2016:

- Stokes flow (applications: creeping flow, earth mantle convection, ...):

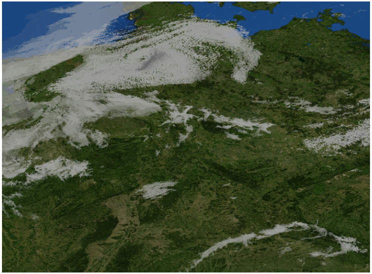
$$\begin{aligned} \nu \Delta \vec{u} + \nabla p &= \vec{f} \\ \nabla \cdot \vec{u} &= 0 \end{aligned}$$

- Solved on unstructured mesh (but with block structure) of tetrahedral elements using multigrid
- Max. number of degrees of freedom: 11 000 000 000 000 (1.1 trillion)
- 20 480 nodes of supercomputer JUQUEEN (327k threads)
- Memory requirement for solution vector etc.: 200 TBytes

Note: Ghattas et. al.: Gordon Bell Prize 2015 for earth mantle flow studies

Simulation Data at Extreme Scale: Examples (3)

Example: Climate and Weather Simulations



project HD(CP)2, dkrz.de

- Current research: Towards global 1km cloud-resolving weather simulation
- Example code ICON: solves compressible nonhydrostatic atmospheric equations of motion and tracers for different phases (water vapor, ice, ...)

- Surface of globe is discretized by icosahedral mesh and successive refinement; atmosphere is resolved by vertical cell layering
  $\rightarrow$ results in ca. 100 000 000 000 grid cells
- Size of a 1km-resolving surface mesh: ca. 1.1 TByte

Simulation Data at Extreme Scale: Examples (3)

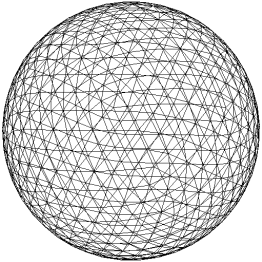
Example: Climate and Weather Simulations

$$
\begin{aligned}
\frac{\partial \tilde{v}_1}{\partial t} + \frac{\tilde{v}_h \cdot \tilde{v}_h / 2}{\partial x_1} - (\xi + f)\tilde{v}_2 + \tilde{v}_3 \frac{\partial \tilde{v}_1}{\partial x_3} &= -c_{pd}\tilde{\theta}_\rho \frac{\partial \bar{\pi}}{\partial x_1} + Q_{v_1} \\
\frac{\partial \tilde{v}_3}{\partial t} + \tilde{v}_h \cdot \nabla_h \tilde{v}_3 + \tilde{v}_3 \frac{\partial \tilde{v}_3}{\partial x_3} &= -c_{pd}\tilde{\theta}_\rho \frac{\partial \bar{\pi}}{\partial x_3} - g + Q_{v_3} \\
\frac{\partial \bar{\rho}}{\partial t} + \nabla \cdot (\tilde{v}\bar{\rho}) &= 0
\end{aligned}
$$

- Current research: Towards global 1km cloud-resolving weather simulation
- Example code ICON: solves compressible nonhydrostatic atmospheric equations of motion and tracers for different phases (water vapor, ice, ...)
- Surface of globe is discretized by icosahedral mesh and successive refinement; atmosphere is resolved by vertical cell layering
  $\rightarrow$ results in ca. 100 000 000 000 grid cells
- Size of a 1km-resolving surface mesh: ca. 1.1 TByte

Simulation Data at Extreme Scale: Examples (3)

Example: Climate and Weather Simulations

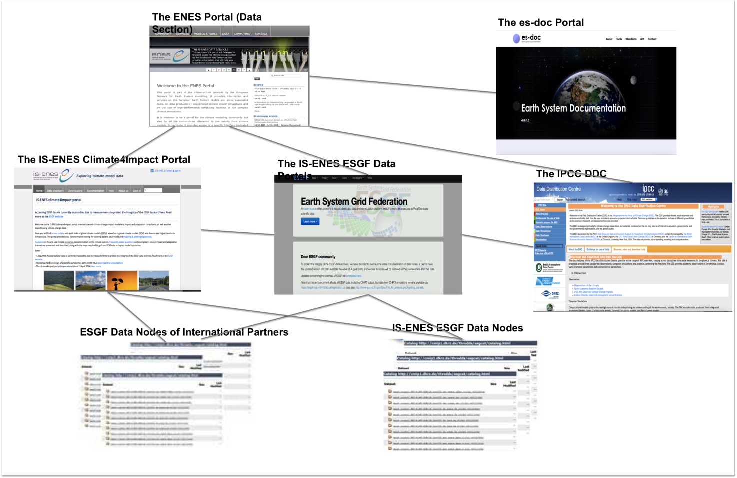


- Current research: Towards global 1km cloud-resolving weather simulation
- Example code ICON: solves compressible nonhydrostatic atmospheric equations of motion and tracers for different phases (water vapor, ice, ...)

- Surface of globe is discretized by icosahedral mesh and successive refinement; atmosphere is resolved by vertical cell layering
  $\rightarrow$ results in ca. 100 000 000 000 grid cells
- Size of a 1km-resolving surface mesh: ca. 1.1 TByte

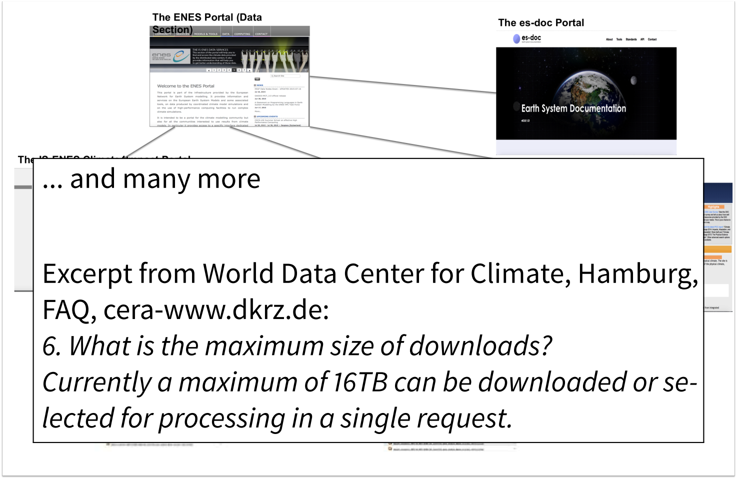
Data Management for Climate

Overview of Data Infrastructure of the European Network for Earth System Modelling (ENES)



Data Management for Climate

Overview of Data Infrastructure of the European Network for Earth System Modelling (ENES)



... and many more

Excerpt from World Data Center for Climate, Hamburg, FAQ, cera-www.dkrz.de:

6. What is the maximum size of downloads?

Currently a maximum of 16TB can be downloaded or selected for processing in a single request.

Another Remedy: In-Situ Analysis and Visualization

- interweave calculation and analysis/visualization
- advantage: all simulation data are potentially available at all times
- dedicated compute nodes for analysis

Fault Tolerance (1)

Question:

How often does your Notebook/PC crash, due to hardware defects or OS errors?

Fault Tolerance (1)

Question:

How often does your Notebook/PC crash, due to hardware defects or OS errors?

Measure: Mean time between failure (MTBF)
$\rightarrow$ example MTBFs: Blue Waters (6-8h), Blue Gene/L ($>$ 10h), Beowulf-style cluster (6h), ...
$\rightarrow$ this will be a (even bigger) issue at exascale!

Failure types:

- Hardware failures: failures that affect groups of nodes, switch, power supply, individual node failure, processor, mother board, disk
- Software failures: scheduler, file system, cluster management software, OS, client daemon

See: A. Gainaru, F. Cappello. Errors and Faults, In Fault-Tolerance Techniques for High-Performance Computing, 2015

Fault Tolerance (2)

Category	Blue Waters (%)	Blue Gene/P (%)	LANL systems (%)
Hardware	43.12	52.38	61.58
Software	26.67	30.66	23.02
Network	11.84	14.28	1.8
Facility/Environment	3.34	2.66	1.55
Unknown	2.98	-	11.38
Heartbeat	12.02	-	-

(Hardware/OS) Error detection:

- Constant hardware health monitoring (e.g., Cray Node Health Checker)
- Performance comparison of different nodes at equal loads
- Similar approach: indexing the logs (how often does an event occur per time)

Silent errors:

- Silent data corruption (SDC), e.g. single bit flip in memory
- Performance variations
- How to resolve: redundant computing, checksum encodings, checkpoint/restart, other kinds of algorithm-level recovery

See: A. Gainaru, F. Cappello. Errors and Faults, In Fault-Tolerance Techniques for High-Performance Computing, 2015

Repetition (1)

A	Clap your hands	B	Stamp your feet
C	Juchhu!	D	Wave your hands!

Repetition (1)

A	Clap your hands	B	Stamp your feet
C	Juchhu!	D	Wave your hands!

Question:

$\boxed{\text{Exa=}\quad}$

A	10^{18}	B	10^{21}
C	10^{15}	D	10^{12}

Repetition (1)

A	Clap your hands	B	Stamp your feet
C	Juchhu!	D	Wave your hands!

Question:

$\boxed{\text{Exa=}}$

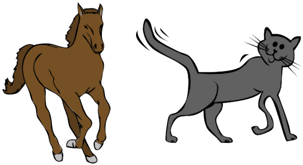
A	10^{18}	B	10^{21}
C	10^{15}	D	10^{12}

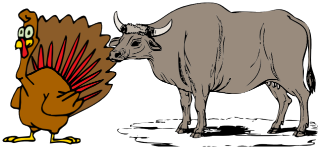
Repetition (2)

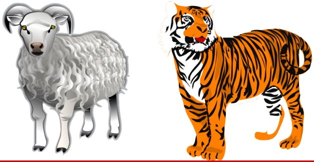
A	Clap your hands	B	Stamp your feet
C	Juchhu!	D	Wave your hands!

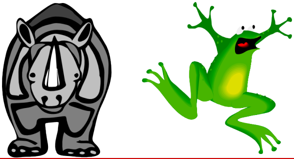
Question:

Which couple is relevant at Exascale?

A


B


C


D


Repetition (2)

A Clap your hands

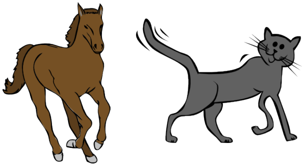
B Stamp your feet

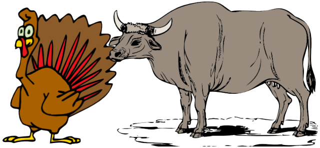
C Juchhu!

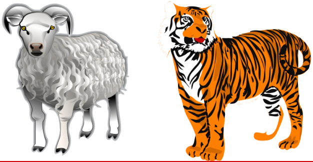
D Wave your hands!

Question:

Which couple is relevant at Exascale?

A 

B 

C 

D 

Repetition (3)

A	Clap your hands	B	Stamp your feet
C	Juchhu!	D	Wave your hands!

Question:

Order the steps of the simulation pipeline: (E) Execution, (M) Model, (I) Implementation, (N) Numerical Algorithm

A	E,M,I,N	B	M,N,I,E
C	M,E,I,N	D	M,E,N,I

Repetition (3)

A	Clap your hands	B	Stamp your feet
C	Juchhu!	D	Wave your hands!

Question:

Order the steps of the simulation pipeline: (E) Execution, (M) Model, (I) Implementation, (N) Numerical Algorithm

A	E,M,I,N	B	M,N,I,E
C	M,E,I,N	D	M,E,N,I

Repetition (4)

A Clap your hands

B Stamp your feet

C Juchhu!

D Wave your hands!

Question:

Which of the following abbreviations is a measure for faults? And what does it mean?

A SDC

B MFZB

C MTBF

D MTHD

Repetition (4)

A	Clap your hands	B	Stamp your feet
C	Juchhu!	D	Wave your hands!

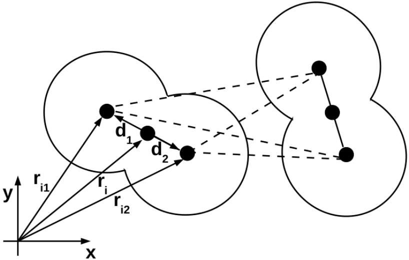
Question:

Which of the following abbreviations is a measure for faults? And what does it mean?

A	SDC	B	MFZB
C	MTBF	D	MTHD

Molecular Dynamics in a Nutshell

- Molecular model: rigid molecules
- Equations of motion

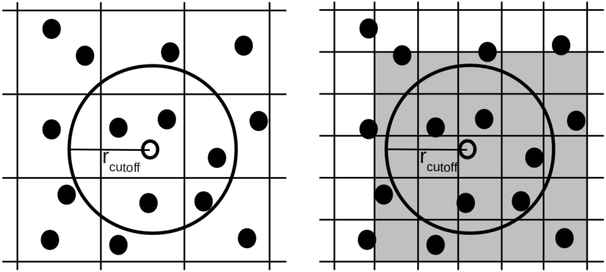


$$
\begin{aligned}
m_i \cdot \frac{d^2\vec{r}}{dt^2} &= \vec{F}_i \\
\frac{d\omega_i}{dt} &= I_i^{-1}\tau_i \\
\vec{F}_i &= \sum_{\substack{j\in particles,\\ j\neq i}} \sum_{n\in sites_i} \sum_{m\in sites_j} \vec{F}_{nm}(\vec{r}_n - \vec{r}_m) \\
\tau_i &= \sum_{n\in sites_i} \vec{d}_n \times \vec{F}_n
\end{aligned}
$$

- Discretize equations by rotational Leapfrog scheme

(Fincham. Molecular Simulation 8:165–178, 1992)

Molecular Dynamics: Short Range Interactions

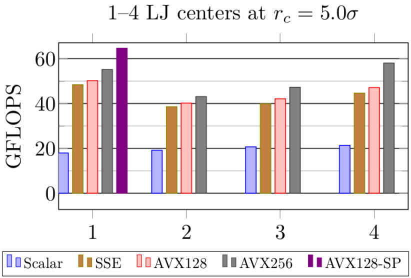


- Molecule-molecule interaction: $O(N^2)$
  $\rightarrow$ only consider local interactions within a *cut-off radius* r_{cutoff}: $O(N)$
- Linked cell algorithm:
  $\rightarrow$ sort molecules into cells of size r_{cutoff}
  $\rightarrow$ only consider molecules for interactions in same or neighboured cells
- Standard vs. generalized linked cells

Single Node Performance

Lennard-Jones interaction:

$$\vec{F}_{nm}^{LJ} = 24\epsilon \frac{1}{||\vec{r}_{nm}||^2} \left(\frac{\sigma}{||\vec{r}_{nm}||}\right)^6 \left(1 - 2\left(\frac{\sigma}{||\vec{r}_{nm}||}\right)^6\right) \vec{r}_{nm}, \quad \vec{r}_{nm} = \vec{r}_m - \vec{r}_n$$



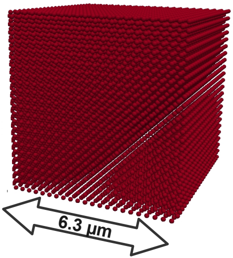
- Node type: Sandy Bridge
- 55 GFLOPS (1CLJ), 58 GFLOPS (4CLJ) $\approx$ 17% peak efficiency
- theoretical limit: $\approx$ 25%
  $\rightarrow$ many+dependent multiplications
  $\rightarrow$ cut-off branching

W. Eckhardt. Efficient HPC Implementations for Large-Scale Molecular Dynamics Simulation in Process Engineering, PhD thesis, 2013

Parallel Performance

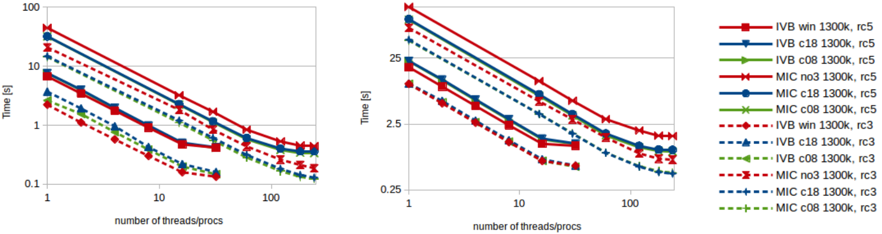
- Number density $\rho\sigma^3 = 0.78$, cut-off radius $r_c = 3.5\sigma$
- $4.52 \cdot 10^8$ particles per node
- **Largest simulation on 9 126 nodes with $4.125 \cdot 10^{12}$ particles**

- In case of liquid krypton:
  cube with edge-length $l = 6.3\ \mu$m
- Peak performance of 591 TFLOPS (9.4 %) on 146 016 cores (292 032 threads)
- **Parallel efficiency of 91.2 % on 146 016 cores** compared to 1 core (2 threads)

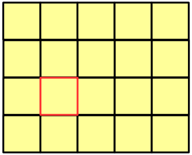


W. Eckhardt. Efficient HPC Implementations for Large-Scale Molecular Dynamics Simulation in Process Engineering, PhD thesis, 2013

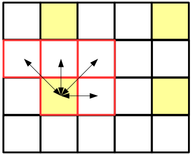
Xeon Phi 5110p vs. IvyBridge (Xeon E5-2650)



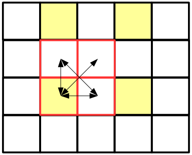
1CLJ 4CLJ



no3



c18



c08

N. Tchipev et al. Euro-Par 2015 Workshop Proceedings, p. 774-785, 2015.

Concept of Lecture: Exascale Computing and Big Data

- Prerequisites: Basics in mathematics and computer architecture
- Time: $\approx$ 90 minutes
- Content:
  - Terminology: Exascale, computing/simulation
  - Exascale development and related challenges: Energy consumption, hardware heterogeneity, fault tolerance, and implications on algorithms
  - Simulation example at extreme scale: Molecular dynamics
- Expected learning outcomes:
  - The students are able to describe the simulation circle.
  - They can define relations between compute intensive simulations, supercomputing, and big data.
  - They can discuss issues that are expected to arise at the exascale and can compare them to the current state.
  - They can give examples for potential fields of research that bring together exascale simulation and big data.
  - They can differentiate between different kinds of hard- and software faults.
  - They can describe the principles of short-range molecular dynamics.