Parallel Algorithms and Programming

Introduction

Thomas Ropars

Email: thomas.ropars@univ-grenoble-alpes.fr

Website: tropars.github.io

Organization of the course

Schedule

- 15 hours of lectures
- 6 hours of tutorials
- 12 hours of labs

Grading

- 30% graded labs (2 graded labs)
- 70% final exam

Useful links

- The slides
- The Moodle page

Teaching staff

- Thomas Ropars (thomas.ropars@univ-grenoble-alpes.fr)
- Sebastien Valat (sebastien.valat@atos.net)

Content of the course

- Programming challenges
- Performance
- Parallel architectures
- Shared-memory algorithms and programming
- Message-passing algorithms and programming
- Parallel linear algebra

Execution Platform

- The course has received a Google Cloud Platform Education Grant
- We will be able to use Google Cloud to run experiments
  - Each student will receive $50 of credits
- Details will be given later during the semester

Introduction

Outline of the lecture

- Why we need parallel systems?
- Applications of parallel computing
- Challenges of parallel computing

References

The content of this lecture is inspired by:

- The lecture notes of F. Desprez
- Introduction to Parallel Computing by B. Barney
- The lecture notes of K. Fatahalian
  - CS149: Parallel Computing @Standford
  - 15418: Parallel Computer Architecture and Programming @CMU
- Parallel Programming – For Multicore and Cluster System. T. Rauber, G. Rünger
- The lecture nodes of S. Lantz

Definition of parallel computing

```
Parallel computing uses multiple processing elements
simultaneously to solve a problem **quickly**.
```

Goal: Performance

- Solve problems faster
- Solve larger problems
- Get a better answer to a problem

Parallel vs Concurrent programming:

- Concurrency is about dealing with lots of things at once.
- Parallelism is about doing lots of things at once.

- Concurrent programming refers to multiple flows of executions executing concurrently.
- Concurrent programming does not necessarily imply parallelism

Note that there are many possible definitions of concurrent programming. Not a very interesting debate here.

Definition of parallel computing

```
Parallel computing uses multiple processing elements
simultaneously to solve a problem **quickly**.
```

Goal: Performance

- Solve problems faster
- Solve larger problems
- Get a better answer to a problem

Parallel vs Concurrent programming:

- Concurrency is about dealing with lots of things at once.
- Parallelism is about doing lots of things at once.

- Concurrent programming refers to multiple flows of executions executing concurrently.
- Concurrent programming does not necessarily imply parallelism
    - Multiple threads executing on a single processor

Note that there are many possible definitions of concurrent programming. Not a very interesting debate here.

Why we need parallel computing?

Until 2000's

- To solve problems that a single processor cannot solve

Today

- To keep increasing the performance of systems
- **End of Moore's law**

Moore's law

An observation made by Gordon Moore in 1965 that the number of transistors in processors chips is doubling every 2 years (because of transistor size reduction)

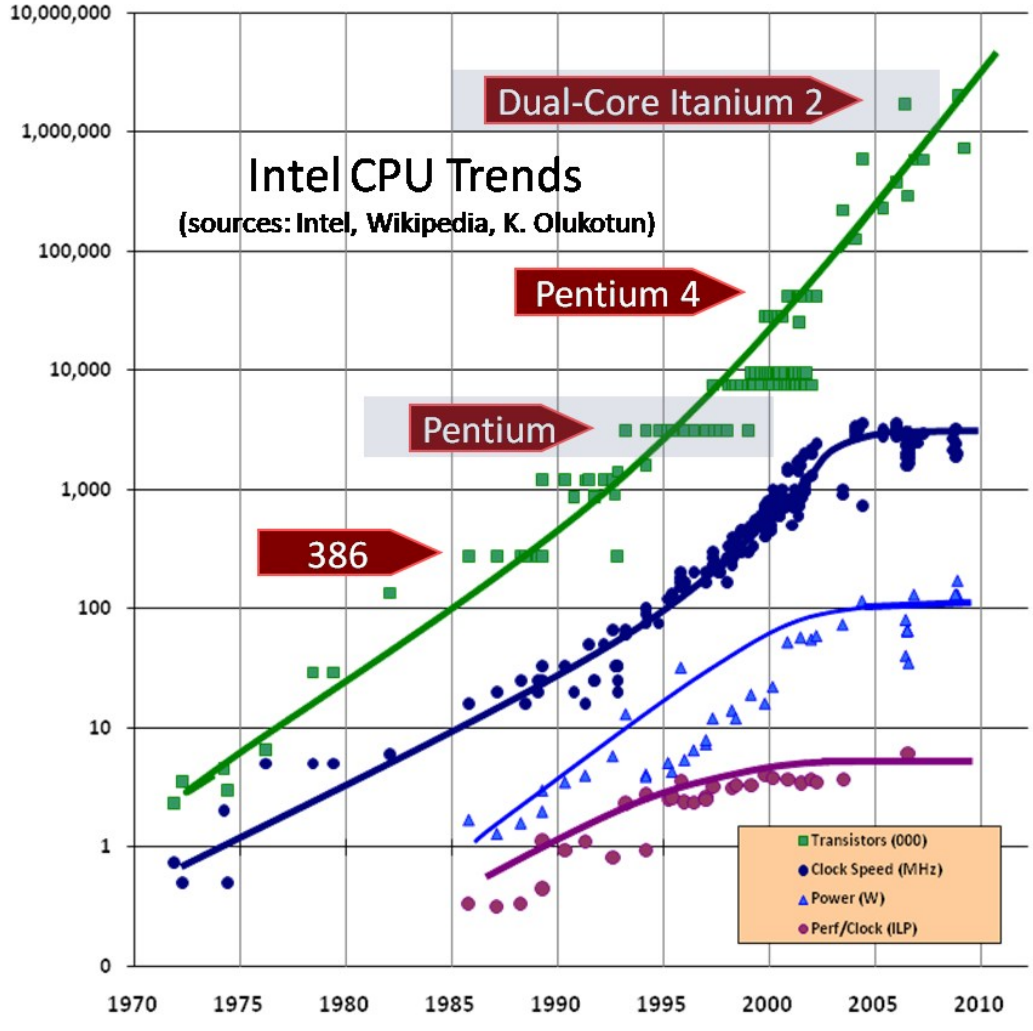
- It was later updated to *doubling every 18 months*

It was implying that:

- the clock frequency could be increased while keeping the power consumption constant (Dennard scaling).

```
To get more performance, you just had to wait for the next
generation of processors
```

The end of the free lunch (H. Sutter)



Lessons learned from previous graph

Moore's law was having two side effects:

Lessons learned from previous graph

Moore's law was having two side effects:

- Allowing increasing the clock frequency
- More transistors were allowing more complex logic to be implemented
    - Optimizing the execution = doing more work per processor cycle
    - **Instruction Level Parallelism** (ILP)

Lessons learned from previous graph

Moore's law was having two side effects:

- Allowing increasing the clock frequency
- More transistors were allowing more complex logic to be implemented
    - Optimizing the execution = doing more work per processor cycle
    - **Instruction Level Parallelism** (ILP)

In the recent years

- No frequency improvement
- No further improvement in ILP

Current state

The number of transistors per chip keeps increasing

- Increase in the number of processor cores
- Trend: large number of simpler cores

To run faster, a program has to be parallel

About ILP

- ILP is a measure of the number of instructions per cycle.
- Several techniques to increase the number of instructions per cycle:
    - Pipelining
    - Branch prediction
    - Out-of-order execution
    - Superscalar architecture
    - Vector operations

```
The expected performance improvement is limited by the
parallelism that can be found in the sequence of instructions
of the program to execute
```

About ILP

A simple C code:

```
a = x*x + y*y + z*z
```

About ILP

A simple C code:

```
a = x*x + y*y + z*z
```

- 5 operations to be executed (ignoring data movements)

What is the best performance that can be obtained with ILP?

About ILP

With ILP

time	operations
1	`o1=x*x; o2=y*y; o3=z*z`
2	`o4=o1+o2`
3	`o5=o4+o3`

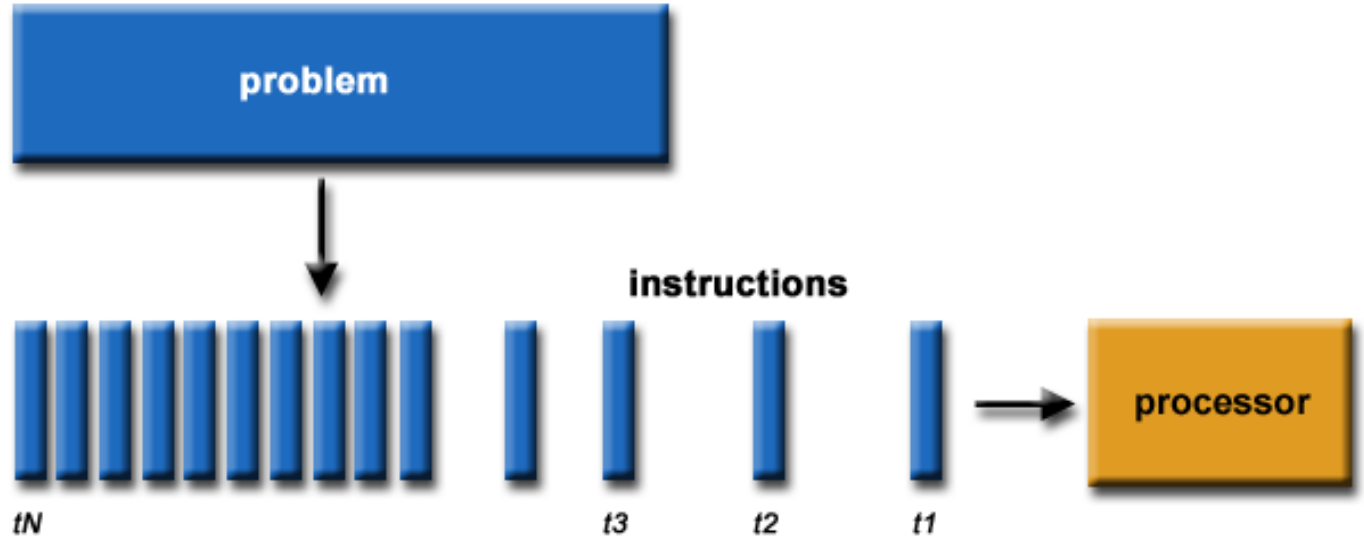
- At most 3 operations can be executed in parallel

Most available ILP is exploited by issuing 4 instructions per cycle

Exemple from K. Fatahalian

What is parallel computing?

Serial computing

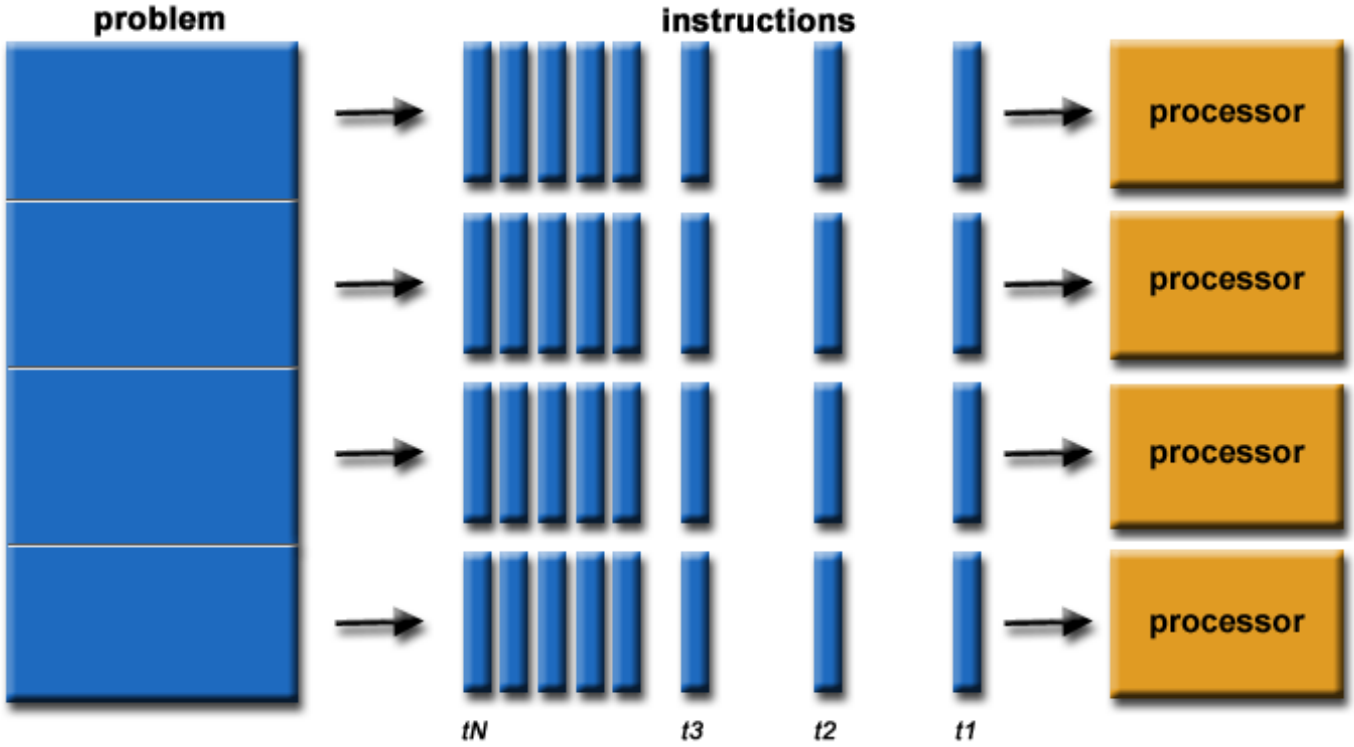


- One instruction is executed at a time (ignoring ILP)

Illustration by B. Barney

What is parallel computing?

Parallel computing



- A problem is broken into multiple parts
- Instructions from multiple parts are executed in parallel
- An overall control/coordination mechanism should be employed

Illustration by B. Barney

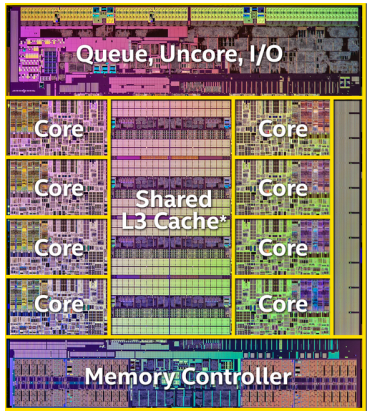
Parallel systems

Desktop computer (and mobile devices)

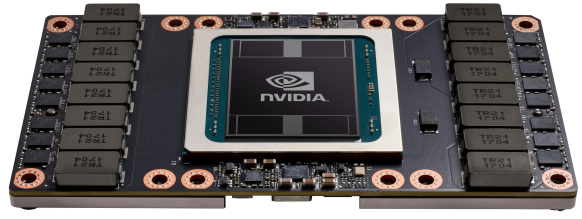
- Multicore processors
- GPUs

Distributed architectures

- Cluster of commodity nodes
- Cloud computing
- Supercomputers (High Performance Computing)



Intel Haswell



NVIDIA Tesla v100

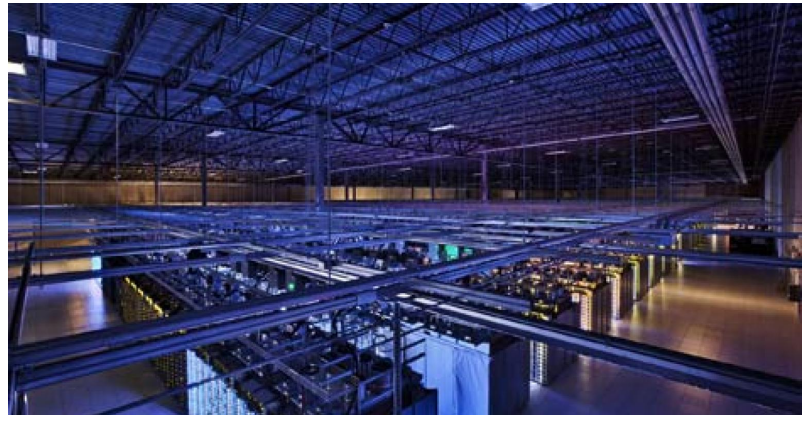
About distributed architectures

Definition

```
*Large* set of computing resources that communicate through a network
```

- Allow building very large scale systems at a reasonable price (Scale out)
- Is in general less expensive than scaling up (updating the hardware with more powerful -- and more expensive -- components)

Summit Supercomputer (Oak Ridge National Lab)



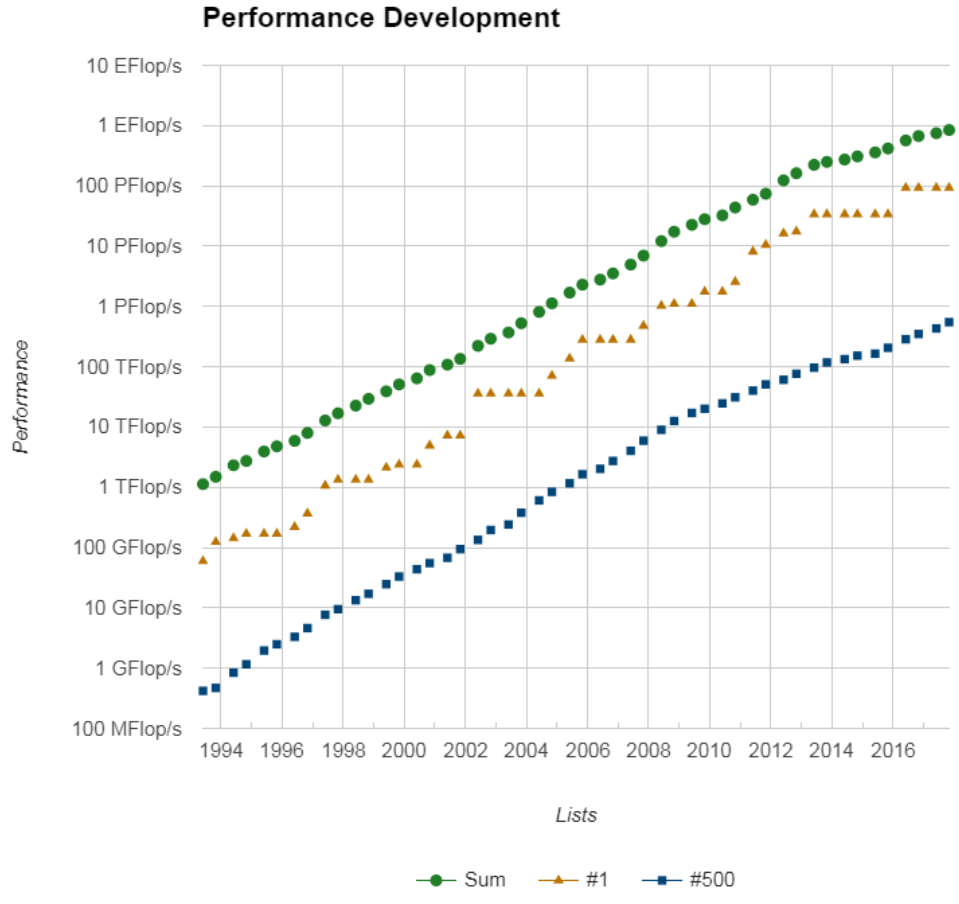
Google data center

Different levels of parallelism

- **Multicore processors:** 10's of processing cores
- **GPUs:** 1000's of processing cores
- **Supercomputers:** Millions of processing cores (CPU and GPU)
- **Cloud computing infrastructures:** Millions of servers

Statistics from TOP 500

Ranking of the 500 hundred most powerful supercomputers (see www.top500.org)



Applications of parallel computing

Use of parallel computing

```
With the evolution of processors, going parallel is the only way to
significantly improve the performance of any application
```

Type of applications requiring large amount of computation

- Numerical simulations
- Data analysis
- Artificial intelligence
- Image processing
- etc.

Numerical simulations

Description

- Simulation can be used to study complex systems/phenomenons
- Without simulation:
    - Study on paper
    - Perform real experiments

Advantages of simulation

- Reduces costs and risks
    - Plane crash
    - Drugs
- Allows simulating phenomena that are hard to study in reality
    - Tsunami
    - Climate evolution

Numerical simulations examples

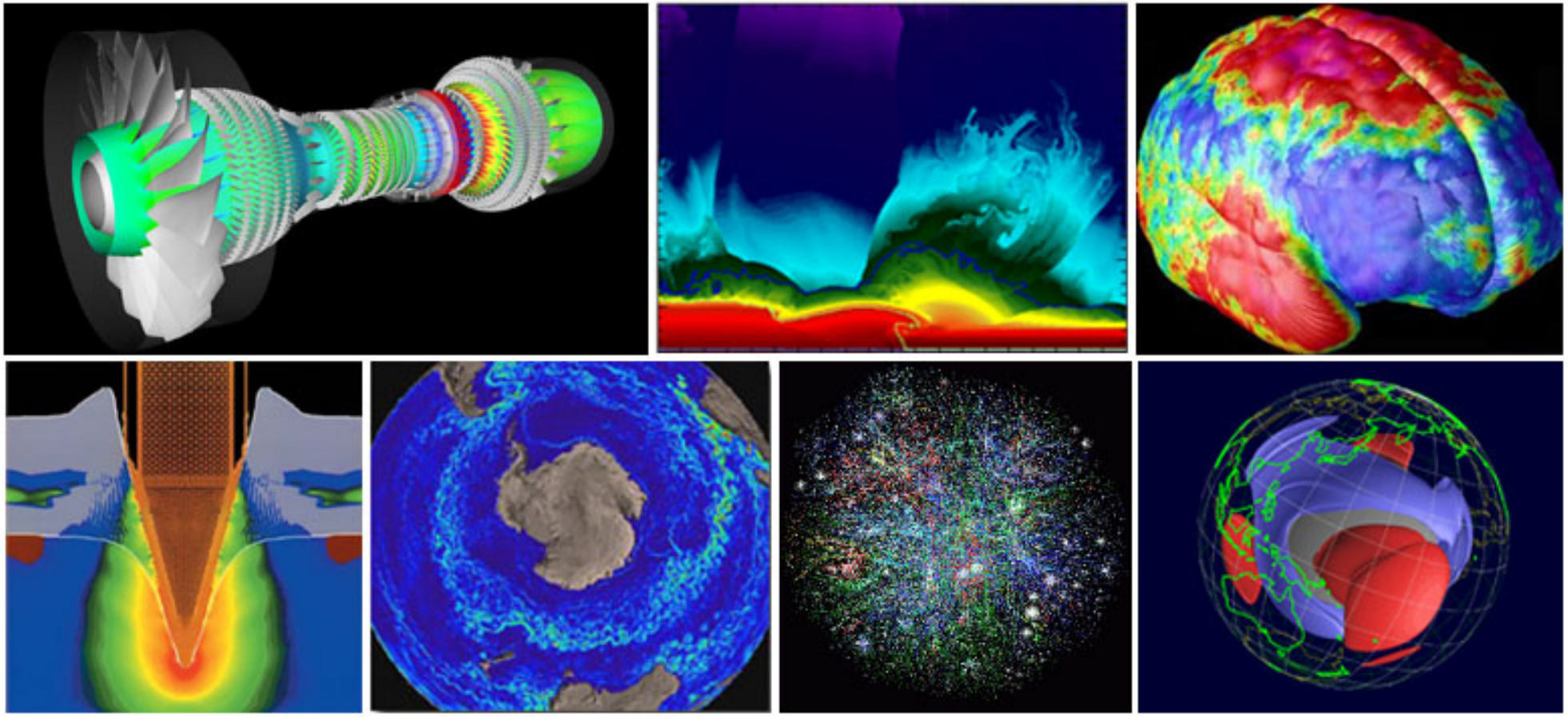
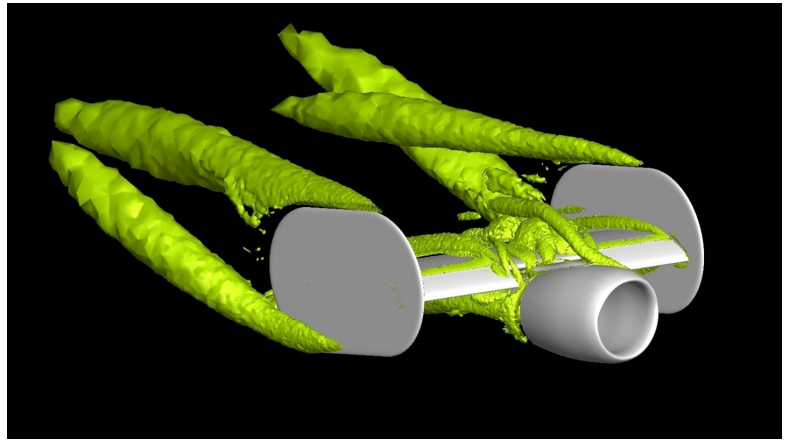


Illustration by B. Barney

Domains where numerical simulations is used

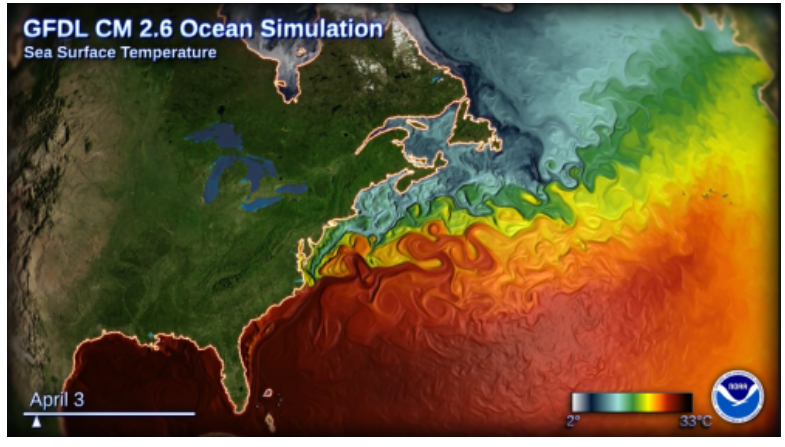
Science

- Climate prediction
- Molecular science
- Seismology
- Bio-science



Engineering

- Engine design
- Airplane design
- Structural modeling

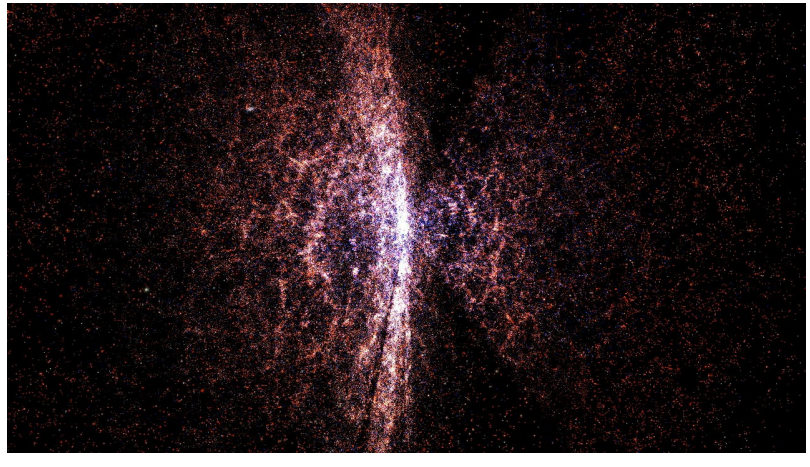


Data analysis (Big Data)

```
Parallel systems can also be used to process huge amounts of data.
```

Domains of application

- DNA analysis
- Data generated by large equipments
    - Telescopes
    - Large Hadron Collider (CERN)
- Data from the Web
    - Analysis of data from social networks
    - Search engines



Credits: NASA/University of Chicago and Adler Planetarium and Astronomy Museum

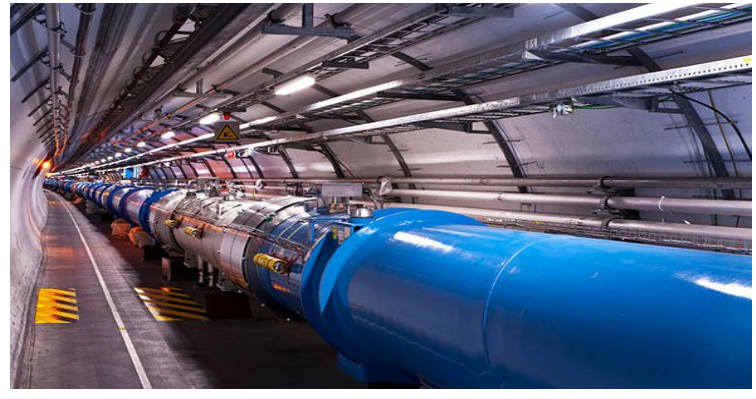
Some numbers

Big Data

- Every 2 days, we create as much information as we did since 2013
- 30M messages posted on Facebook every minute
- 570 new web sites every minute

Large Hadron Collider

- The most powerful instrument ever built to investigate elementary particles
- 40 PB of raw data per second during an experiment
- 10 PB of useful data per year

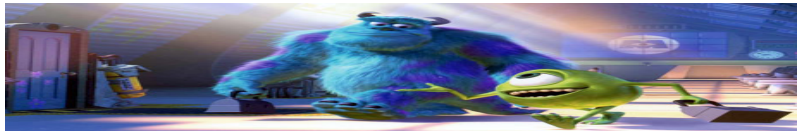


Credit: https://www.slideshare.net/BernardMarr/big-data-25-facts | Maximilien Brice (CERN)/Wikimedia Commons

Entertainment industry

Computer-generated imagery

- Advanced graphics
    - Animation movies
- Virtual reality



2001, Pixar, Monster Inc.: 250 servers with 14 processors (3500 processors)

Artificial intelligence

Machine learning

```
Build a mathematical model out of training data
```

- Classification
- Clustering
- Regression

Applications

- Recommendation systems
- Anomaly detection
- Speech recognition
- etc.

The case of deep learning

- Execution on clusters of GPUs (from 10's to 1000's of nodes)
- Training a neural network = Operations on large matrices

Challenges of parallel programming

Making a program parallel

Multiple challenges

- Find parallelism in the problem
- Divide the work into multiple tasks
  - Granularity of the tasks
- Study the dependencies between the tasks
- Study the need for communication and synchronization
- Study the need for load balancing

Taking into account the underlying hardware

```
The best way of parallelizing a computation depends on the target platform
```

Main points to take into account

- Different levels of parallelism
- Storage hierarchy
    - Size and performance of the different storage levels
- Communication model
    - Shared memory vs message passing
- Topology
    - Network
    - Network on chip

About the storage hierarchy

- Each level has different performance and capacity
- Data transfers have a huge impact on the performance of programs

Storage level	latency	bandwidth
L1 cache	ns	KB
L3 cache	10's ns	MB
DRAM	100 ns	10 GB
SSD	10 us	100 GB
Disk	10 ms	TB
Storage server	s	PB

Search for "Latency numbers every programmer should know"

Conclusion

Take-away points

Single-thread performance increases very slowly

- *End* of Moore's law
- To improve performance, the programs have to be parallel

A need for parallel computing

- Several application domains require much more computing power than what a single processor core can provide
- Parallel systems can fullfill this need
    - They can feature huge numbers of computing resources

Parallel programming requires a dedicated expertise

- Writing parallel programs is a challenging task
    - Finding parallelism and dealing with communication, dependencies, load imbalance, etc
    - Designing solutions that can make best use of the underlying hardware