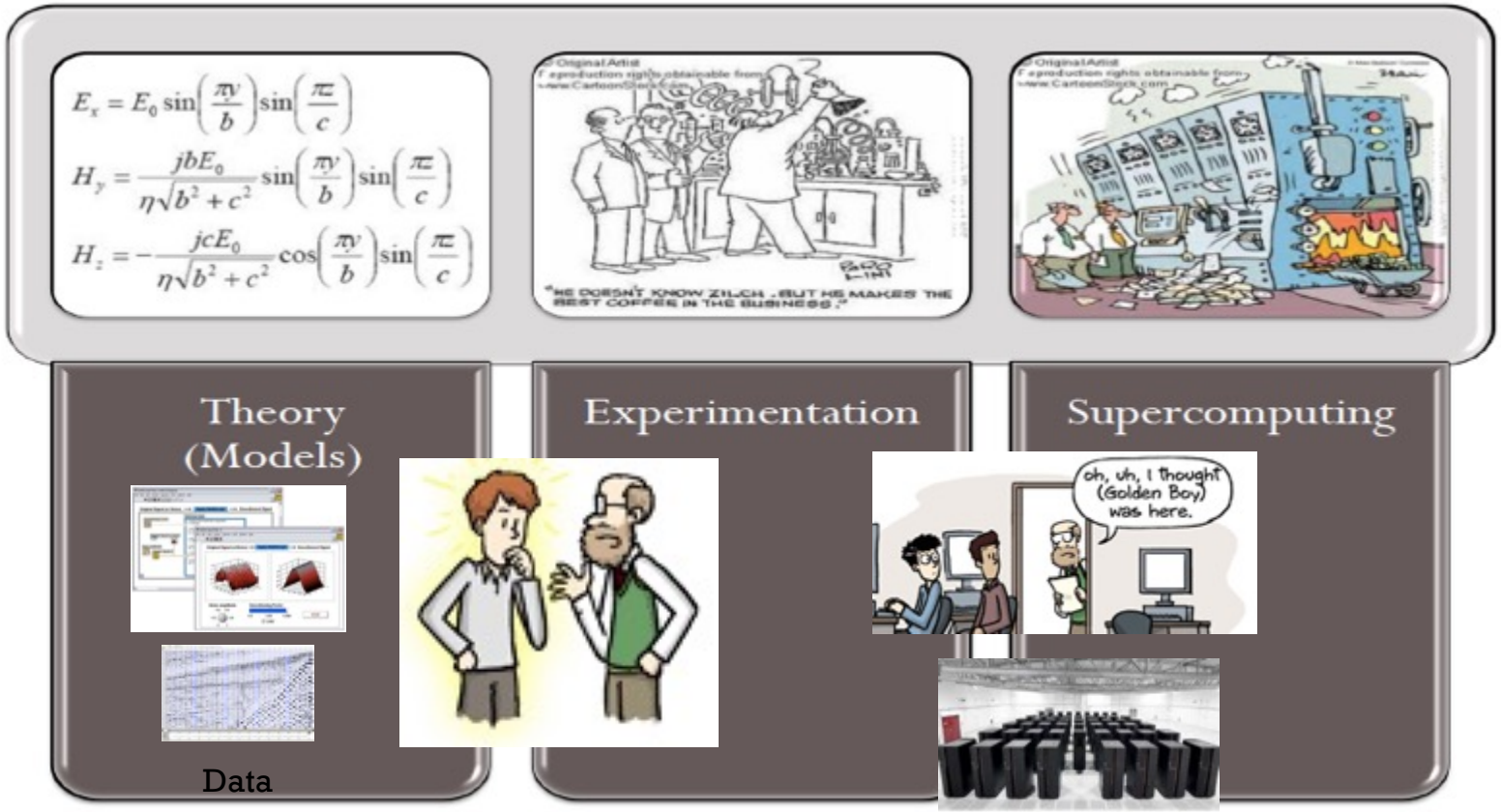
Parallel Computing Introduction

Carlos Jaime Barrios Hernández, PhD

@carlosjaimebh

Science and Advanced Technology for All



Decision Takers

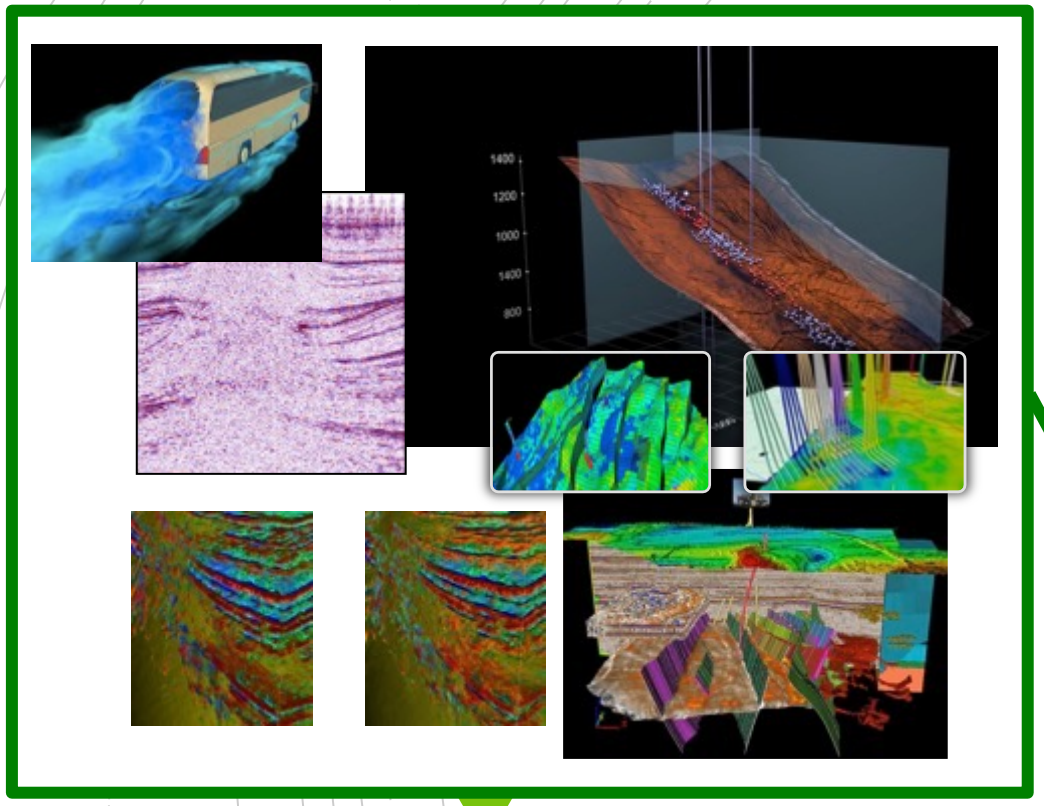
Citizens, All

Experts- Scientists (Biologists, Architects, Pysiciens)

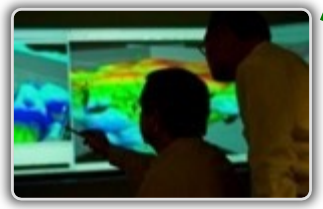
Experts Scientists – Computer and Informatics Informatic, Math Applied

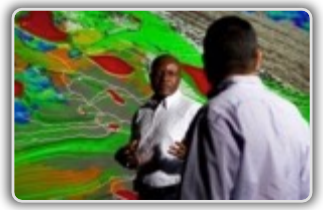
Technology - Infrastructure

Why?

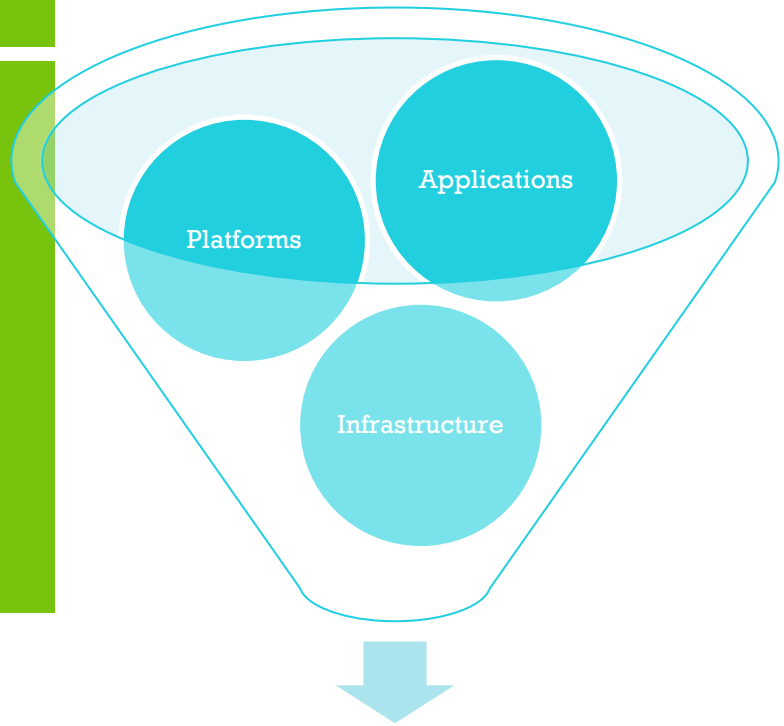


- Large Data Sets
- Complex Mathematics
- Complex Models
- Real Time
- Interaction and Confrontation
- Large Scale Visualization
- High Resolution
- High Performance and Capacity
  - VR Needs
  - Big Data and Deep Learning





Big Problems, Smart Solutions



High Performance
(Computing) Knowledge

Challenges

Infrastructure

- Post Moore Era Architectures
  - Parallel Balancing, I/O, Memory Challenges
- Dark Sillico
- Exascale
  - Computer Efficiency (Processing/Energy Consumption)
- Hybrid Platforms (CISC+RISC+Others)
  - TPUs, ARM…
- Data Management
- Advanced Networks
- Fog/Edge
- HPC@Pocket
- … Quantum Computing

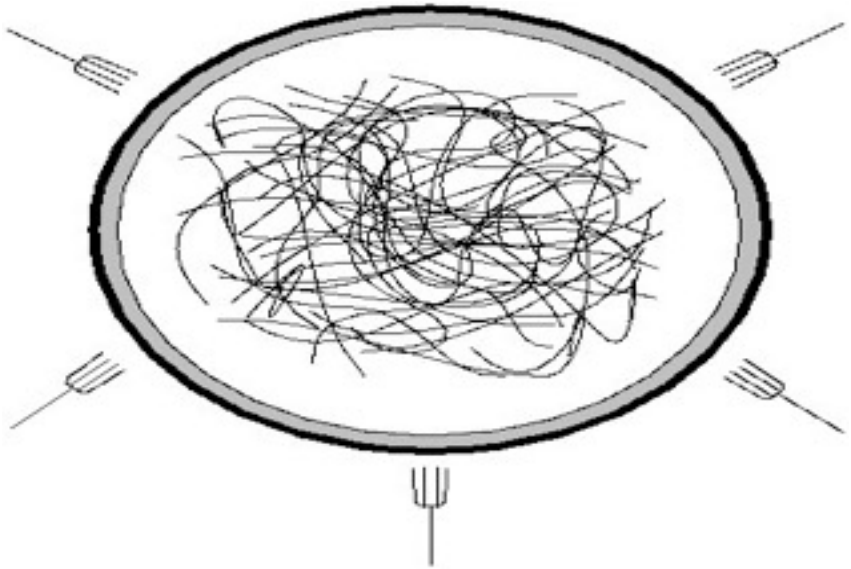
Platform

- Programmability
  - New Languages and Compilers
- Computing Efficiency
- Data Movement and Processing (In Situ, In Transit, Workflows)
- HPC as a Service
  - Science Gateways, Containers
- Viz as a Service (In Situ)
- Protocols
- IA and Deep Learning Frameworks
- Quantum Computing

Applications

- IA and Deep Learning
- Algorithms Implementation
- Use of Interpretators (as Python)
- Community versions
- Open Algorithms, Open Data
- Utra Scale Applicatons
- …and more!

About Parallelism

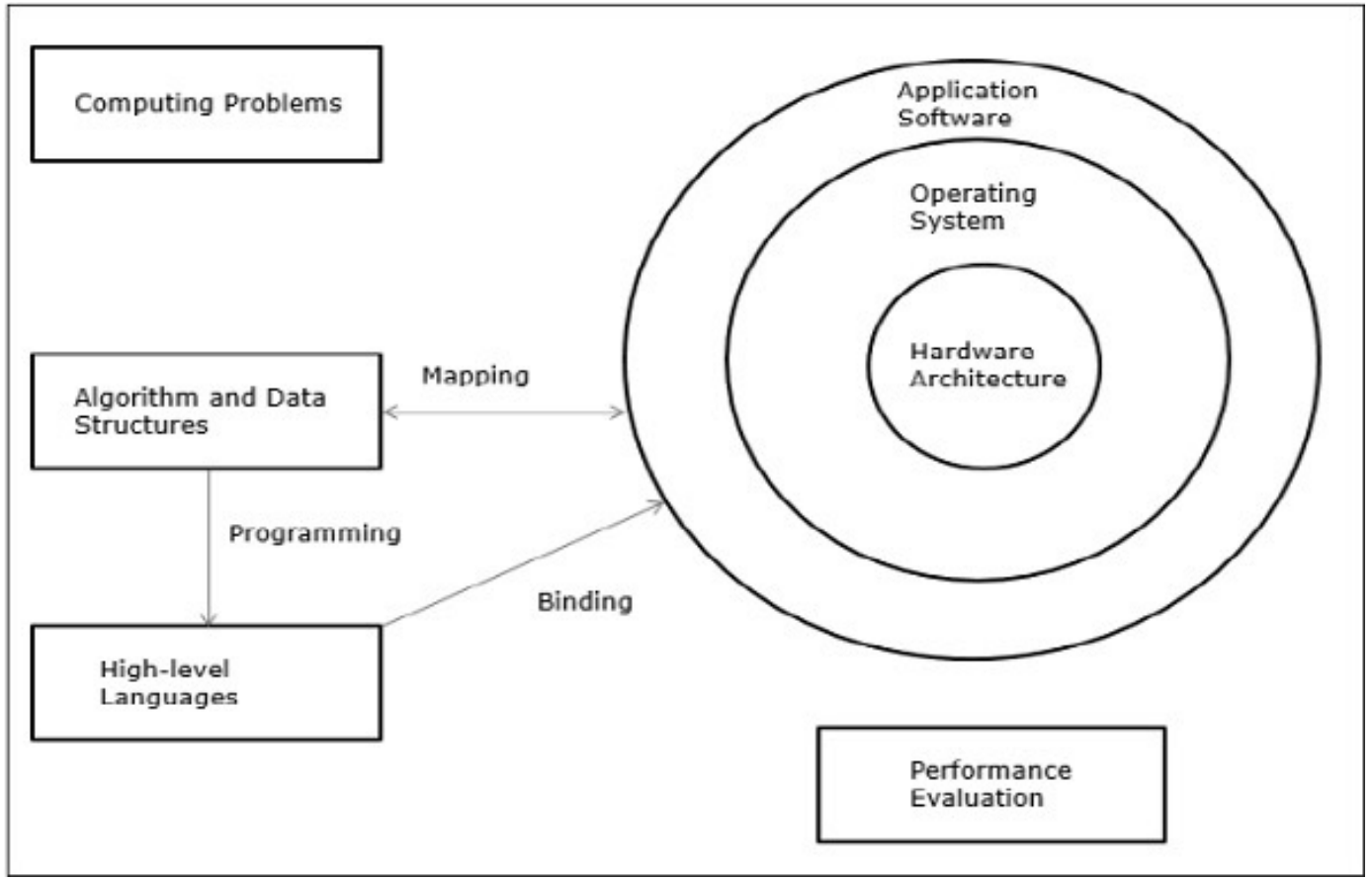


- **Concurrency** is a property of systems in which several computations are executing simultaneously, and potentially interacting with each other.

- **Implicit parallelism** is a characteristic of a programming language that allows a compiler or interpreter to automatically exploit the parallelism inherent to the computations expressed by some of the language's constructs.
- **Explicit parallelism** is the representation of concurrent computations by means of primitives in the form of special-purpose directives or function calls.

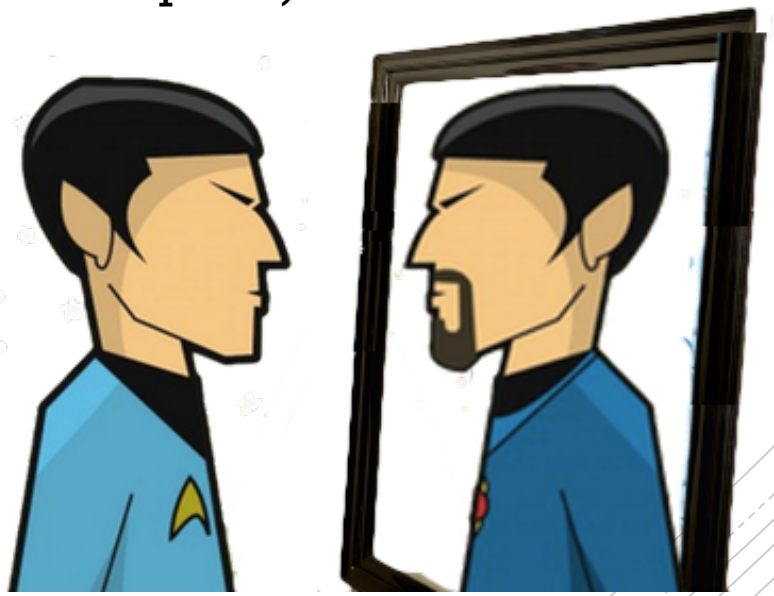
Elements of Parallelism

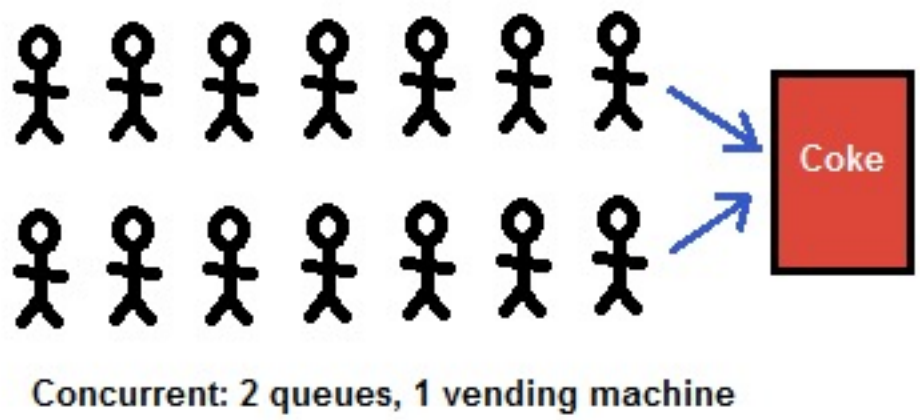
1. Computing Problems
   - Numerical (Intensive Computing, Large Data Sets)
   - Logical (AI Problems)
2. Parallel Algorithms and Data Structures
   - Special Algorithms (Numerical, Symbolic)
   - Data Structures (Dependency Analysis)
   - Interdisciplinary Action (Due to the Computing Problems)
3. System Software Support
   - High Level Languages (HLL)
   - Assemblers, Linkers, Loaders
   - Models Programming
   - Portable Parallel Programming Directives and Libraries
   - User Interfaces and Tools
4. Compiler Support
   - Implicit Parallelism Approach
     - Parallelizing Compiler
     - Source Codes
   - Explicit parallelism Approach
     - Programmer Explicitly
       - Sequential Compilers, Low Level Libraries
       - Concurrent Compilers (HLL)
     - Concurrency Preserving Compiler
5. Parallel Hardware Architecture
   - Processors



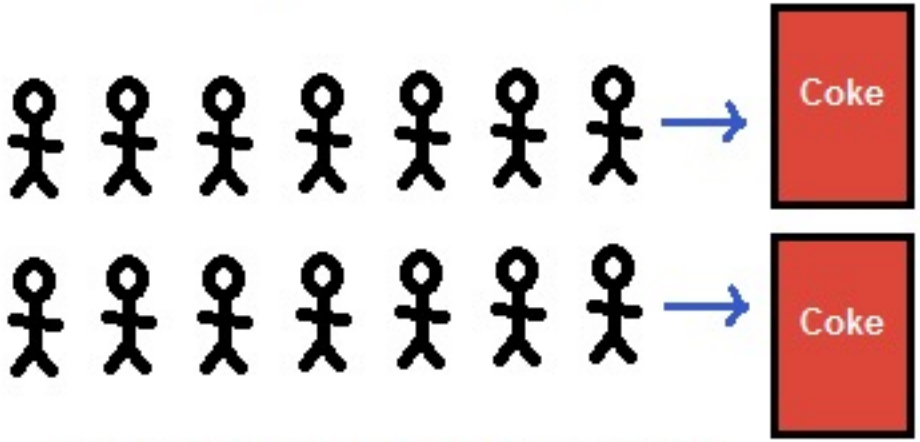
Pervasive and Thinking Parallelism

- It is not a question of « Parallel Universes » (Almost)
- Data Sources
- Processing and Treatment
- Resources (Available and Desire)
- Energy Consumption
- Natural "thinking" (Natural Compute?)



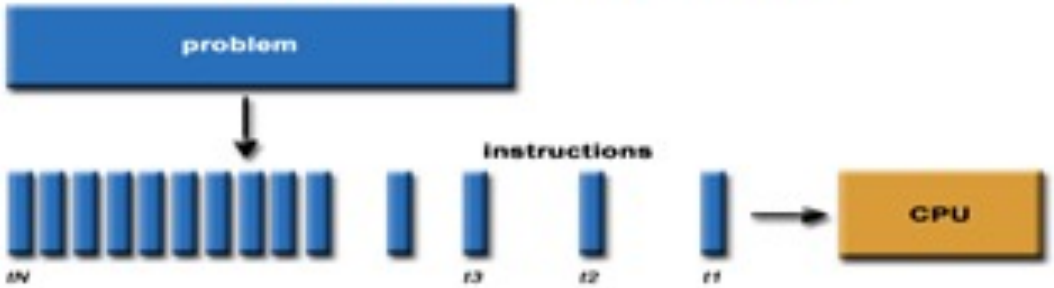


Concurrent: 2 queues, 1 vending machine

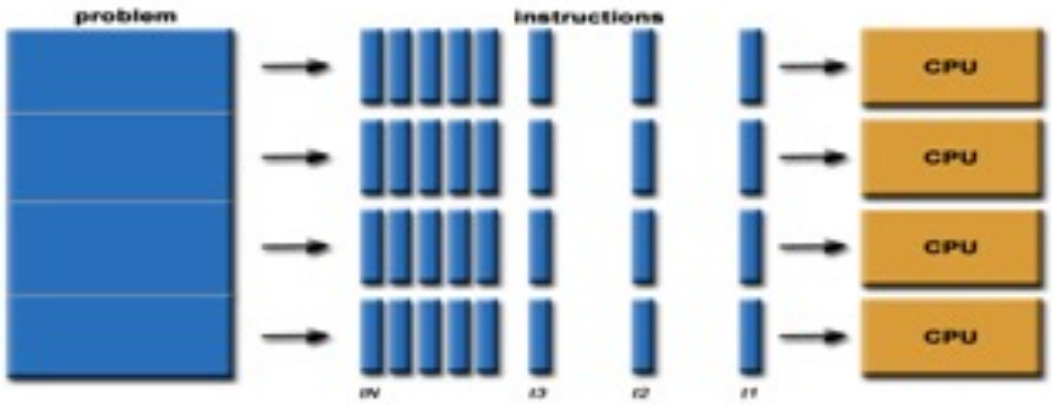


Parallel: 2 queues, 2 vending machines

Traditional Sequential Processing

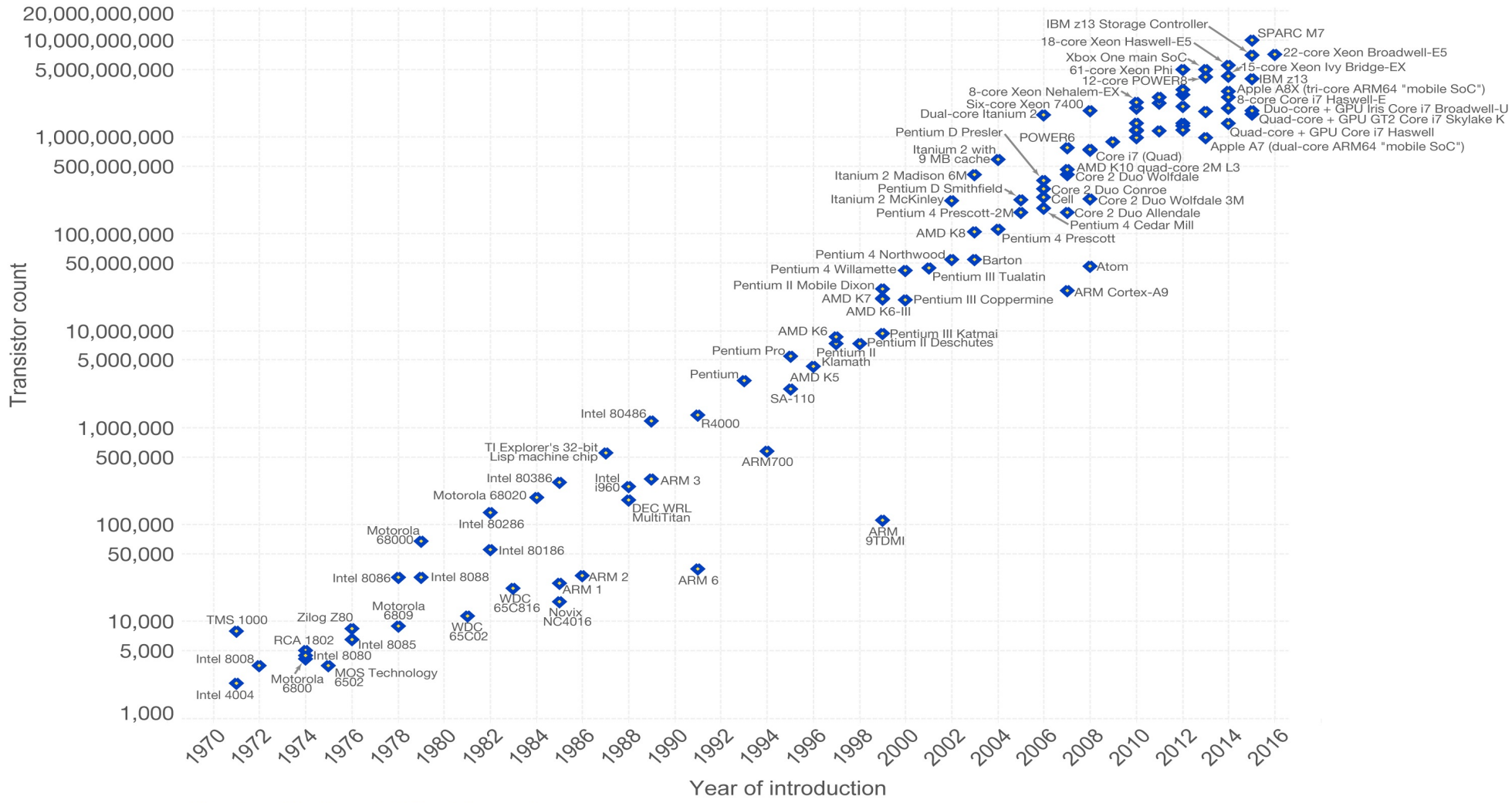


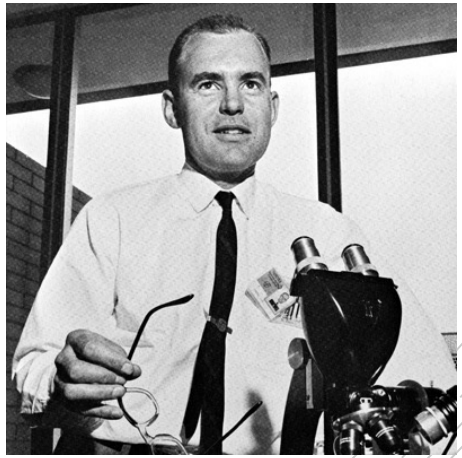
Parallel Processing



Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

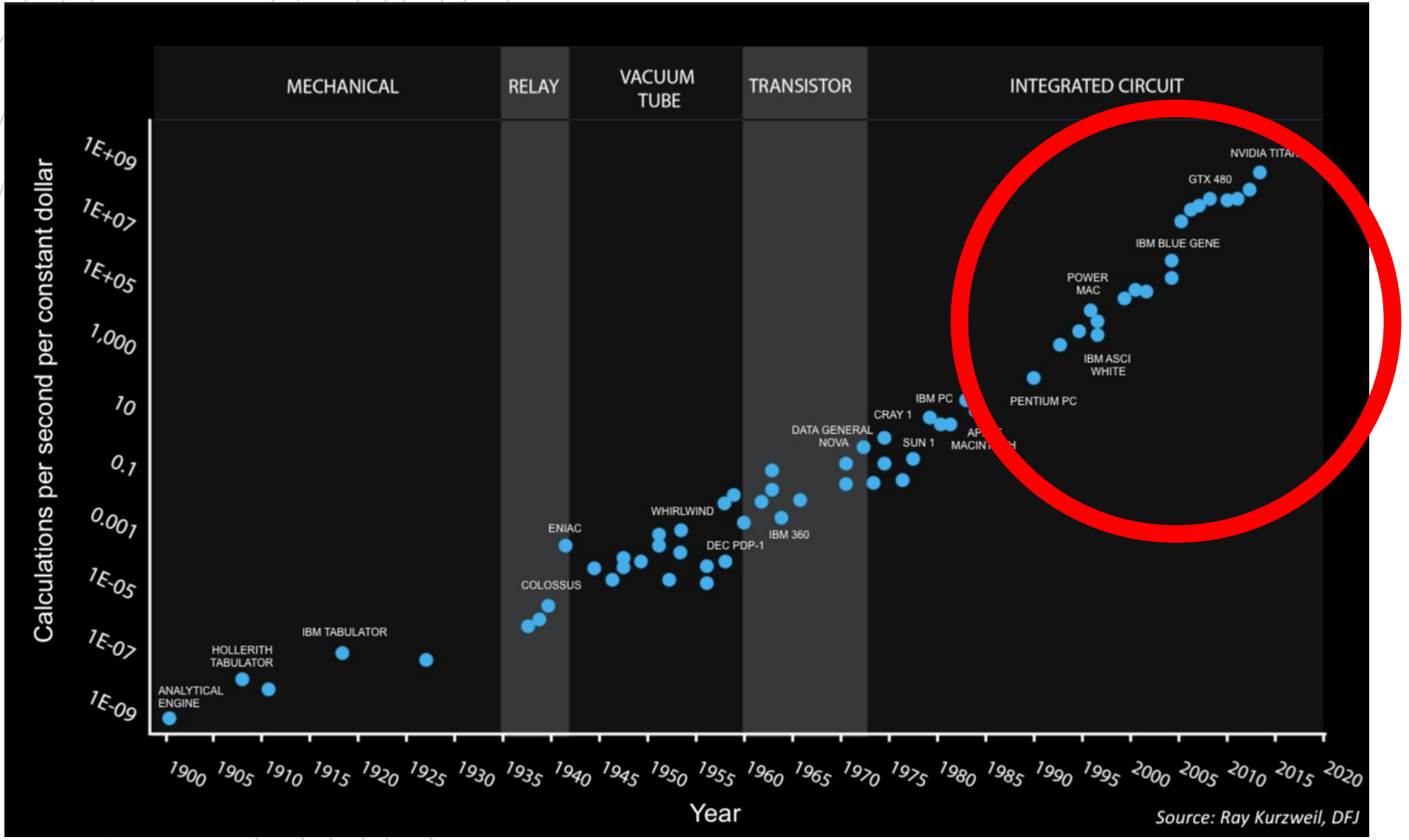
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.





Gordon Moore (In the 60's)

Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

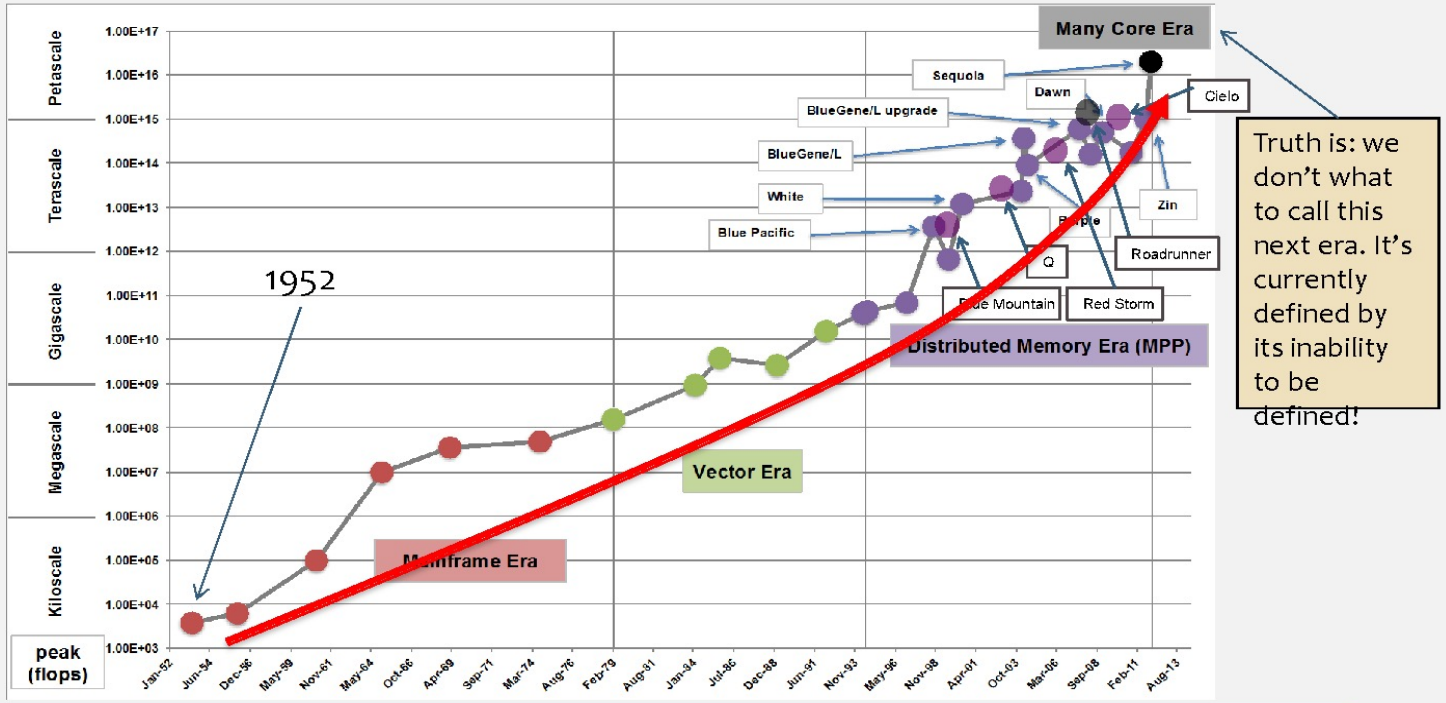


After 120 years… The Moore's Law is Dead

Jack Dongarra

It is more than a publicity!

Advancements in (High Performance) Computing Have Occurred in Several Distinct "Eras"



Each of these eras define not so much a common hardware architecture, but a common programming model

Rob Neely

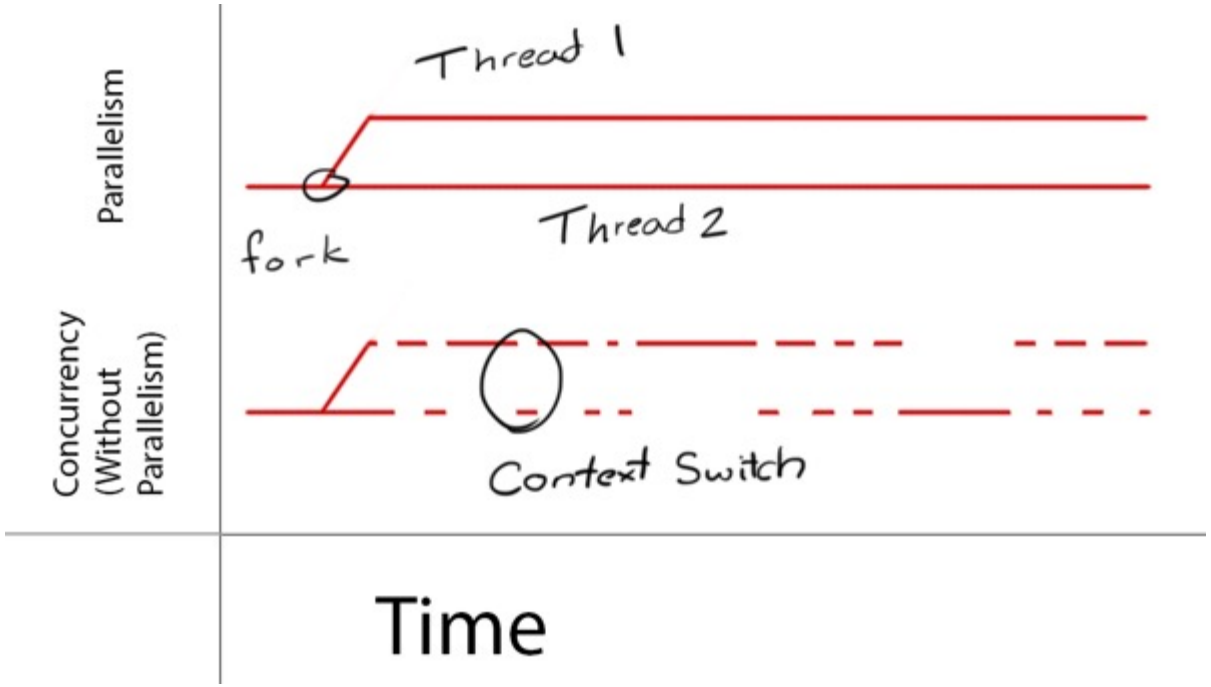
About High Performance Computing

- HPC is useful to being faster, more precise overall, to solve large problems and to treat, intrinsically, parallelism in essence.
- However allows
  - Technological Advantage
  - Technological Independency
  - Competitively
  - Energy Savings
- But, HPC is expensive

What & Why

- What is high performance computing (HPC)?
  - The use of the most efficient algorithms on computers capable of the highest performance to solve the most demanding problems.
- Why HPC?
  - Large problems – spatially/temporally
    - 10,000 x 10,000 x 10,000 grid □ 10^12 grid points □ 4x10^12 double variables □ 32x10^12 bytes = 32 Tera-Bytes.
    - Usually need to simulate tens of millions of time steps.
    - On-demand/urgent computing; real-time computing;
  - Weather forecasting; protein folding; turbulence simulations/CFD; aerospace structures; Full-body simulation/ Digital human …

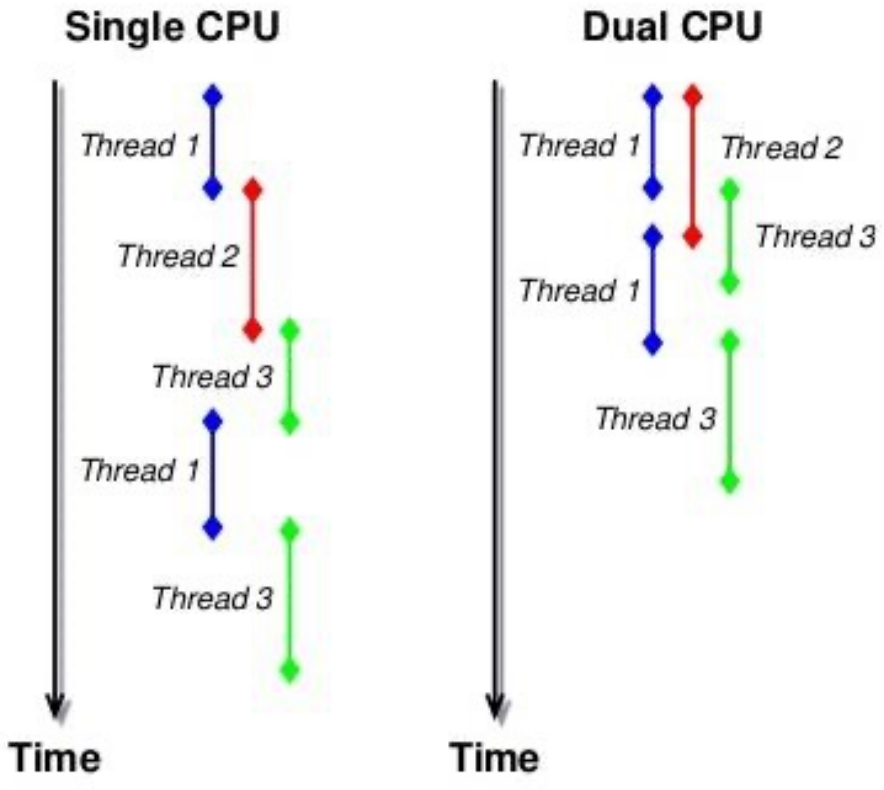
Concurrency vs Concurreny/Parallelism Behavior



Non Shared Processing Ressources (However the Memory...)
Switching
Parallel Threards (Multitasking, Multithreading)

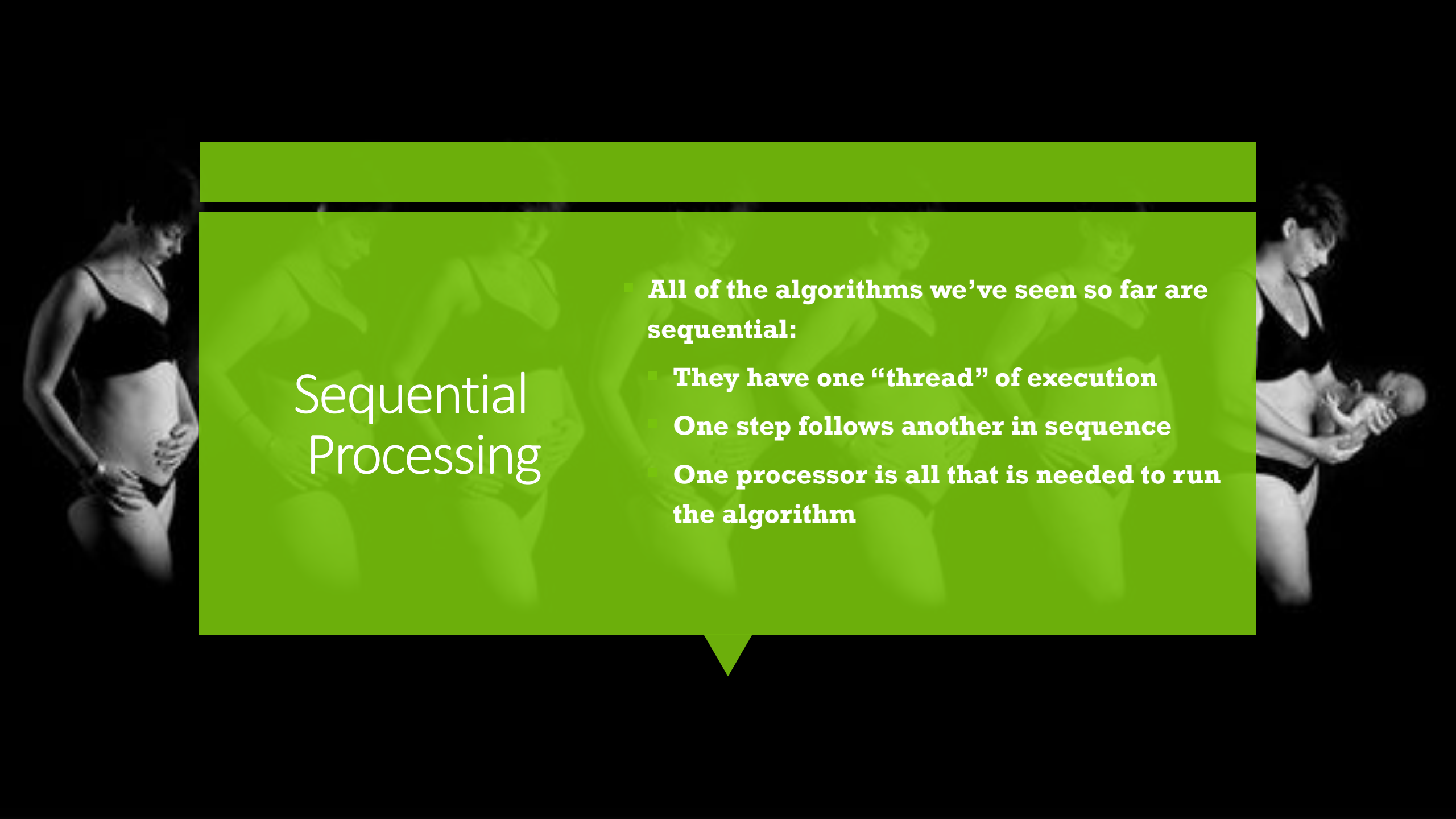
Shared Processing Ressources
Switching
Non Parallel Threards (Non Multitasking, Yes Multithreading)

Concurrency vs Concurreny/Parallelism Example



Single System
- Multiple Threads in Runtime
- Almost Synchronization Strategies
- Memory Allocation

Dual System
- Multiple Paralle Threads in Runtime
- Strategies to Paralellism following models (PRAM, LogP, etc) addressed to exploit memory and overhead reduction



Sequential Processing

- **All of the algorithms we've seen so far are sequential:**
  - **They have one "thread" of execution**
  - **One step follows another in sequence**
  - **One processor is all that is needed to run the algorithm**

Concurrent Systems

- **A system in which:**
  - **Multiple tasks can be executed at the same time**
  - **The tasks may be duplicates of each other, or distinct tasks**
  - **The overall time to perform the series of tasks is reduced**

Advantages of Concurrency

- **Concurrent processes can reduce duplication in code.**
- **The overall runtime of the algorithm can be significantly reduced.**
- **More real-world problems can be solved than with sequential algorithms alone.**
- **Redundancy can make systems more reliable.**

Disadvantages of Concurrency

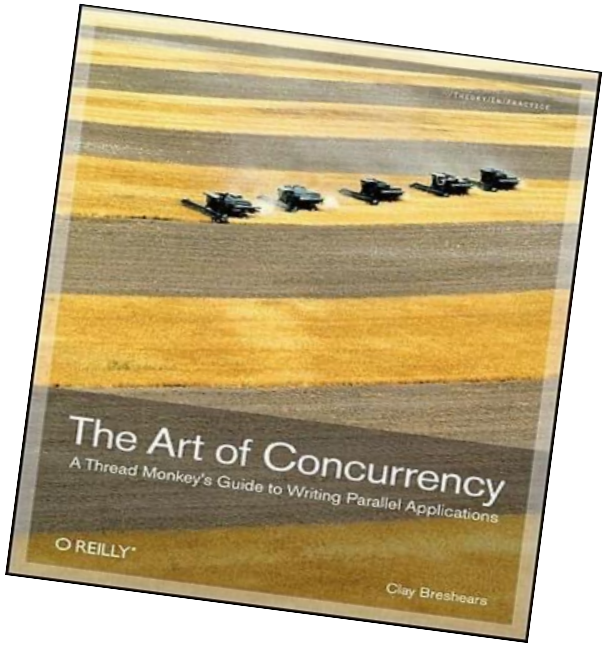
- **Runtime is not always reduced, so careful planning is required**
- **Concurrent algorithms can be more complex than sequential algorithms**
- **Shared data can be corrupted**
- **Communications between tasks is needed**

Concurrent Programming General Steps

- **Analysis**
- Identify Possible Concurrency
  - Hotspot: Any partition of the code that has a significant amount of activity
  - Time spent, Independence of the code…
- **Design and Implementation**
- Threading the algorithm
- **Tests of Correctness**
- Detecting and Fixing Threading Errors
- **Tune of Performance**
- Removing Performance Bottlenecks
  - Logical errors, contention, synchronization errors, imbalance, excessive overhead
  - Tuning Performance Problems in the code (tuning cycles)

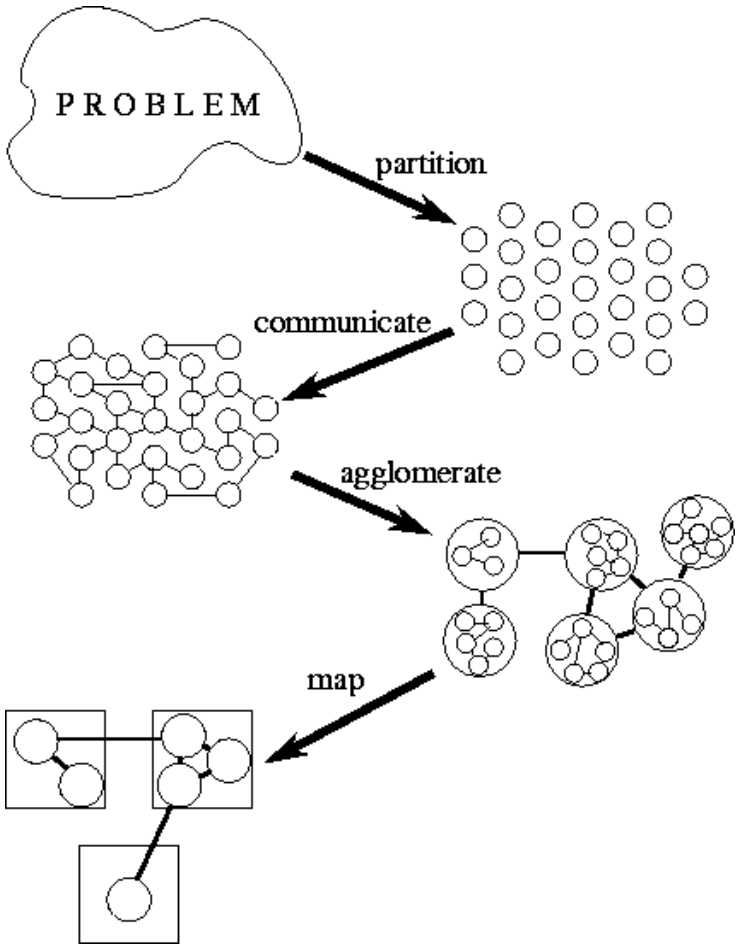
Parallel Computing

- Parallel Computing exploit Concurrency
  - In “system” terms, concurrency exists when a problem can be decomposed in sub problems that can safely executed at same time (in other words, concurrently)



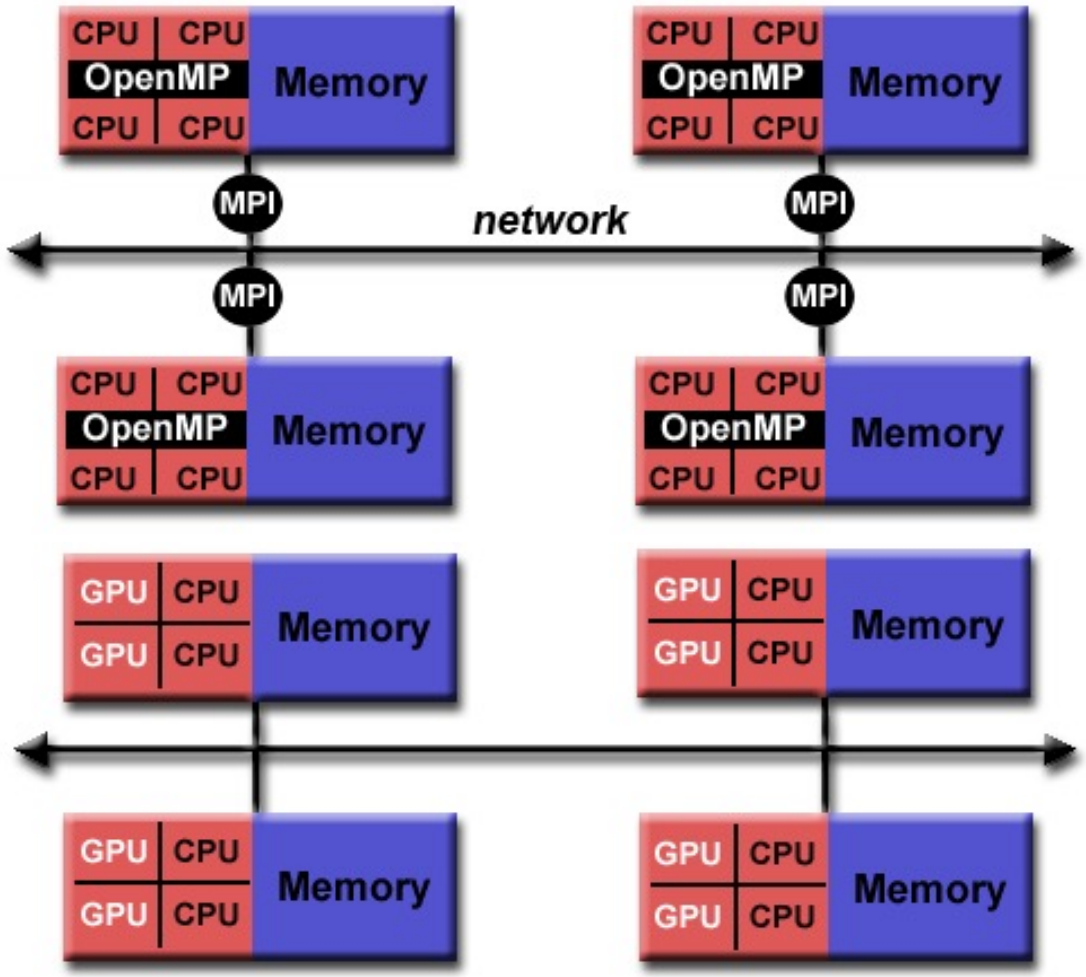
https://ignorelist.files.wordpress.com/2012/01/the-art-of-concurrency.pdf

Traditional Way



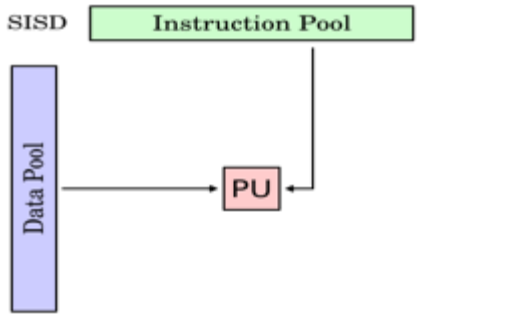
Designing and Building Parallel Programs, by Ian Foster in http://www.mcs.anl.gov/~itf/dbpp/

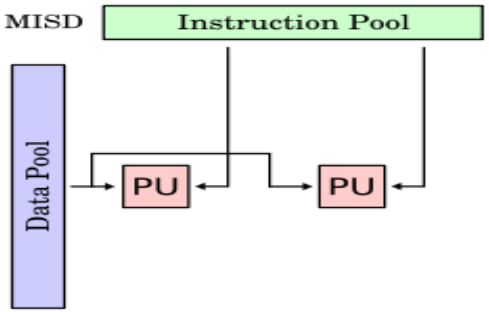
Shared, Distributed and Hybrid Memory Architectures

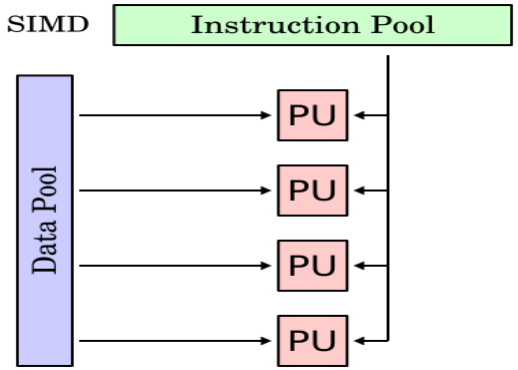


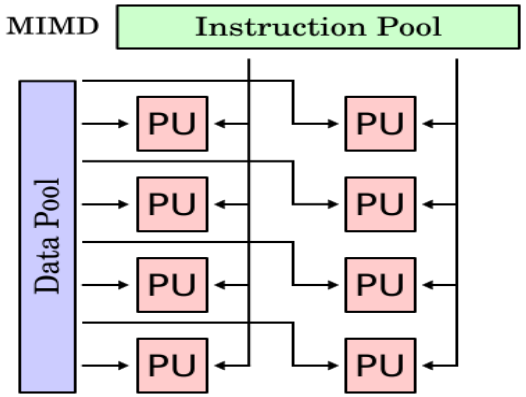
- Memory Exploitation involves Memory Hierarchy
  - Models as PRAM, BSP, etc..
- All modern architectures to HPC allows different memory models
  - Shared Memory (Inside Nodes)
  - Distributed Memory (Among Nodes)
  - Hybrid Memory
    - Using Accelerators (GPUs, MICs)
    - Interaction Nodes/Processors

Flynn's Taxonomy*



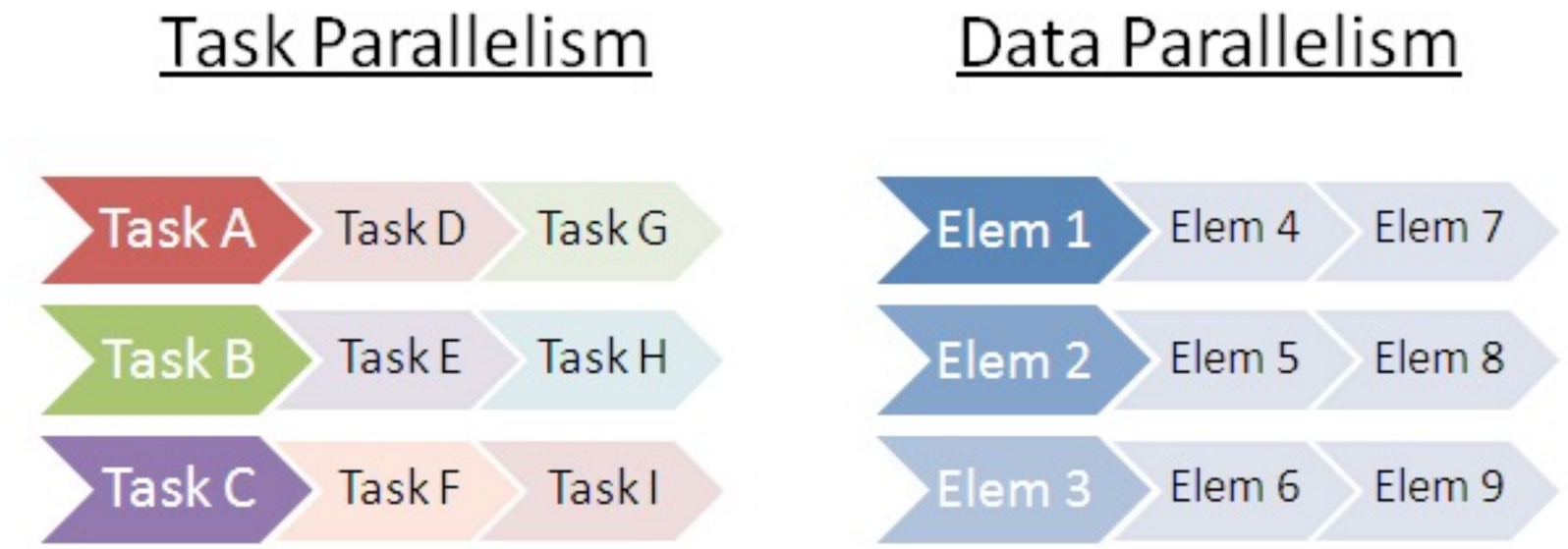






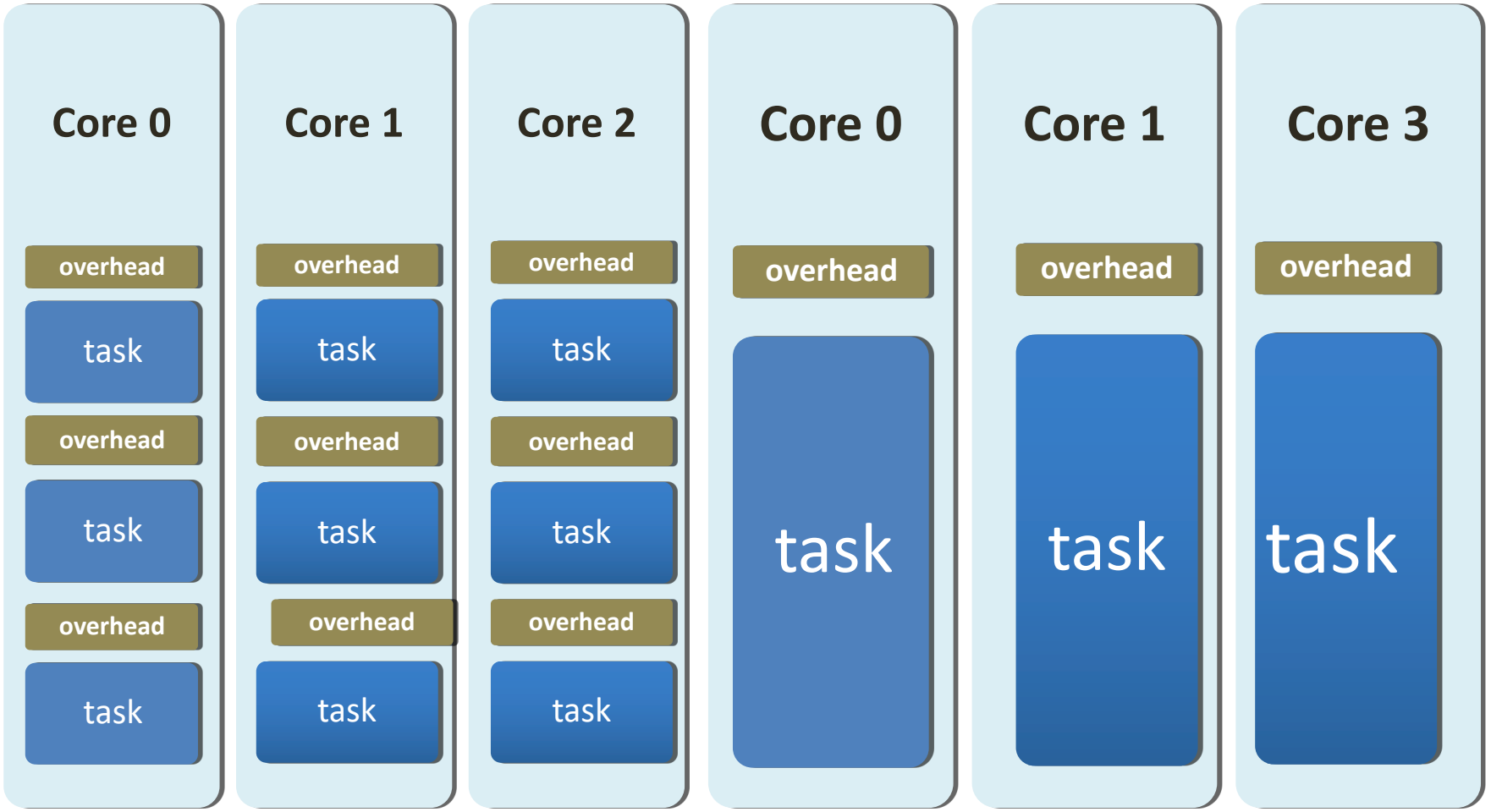
* Proposed by M. Flynn in 1966

Descomposition



Tasks Decomposition : Task Parallelism
Data Decomposition: Data Parallelism /Geometric Parallelism

Task Granularity

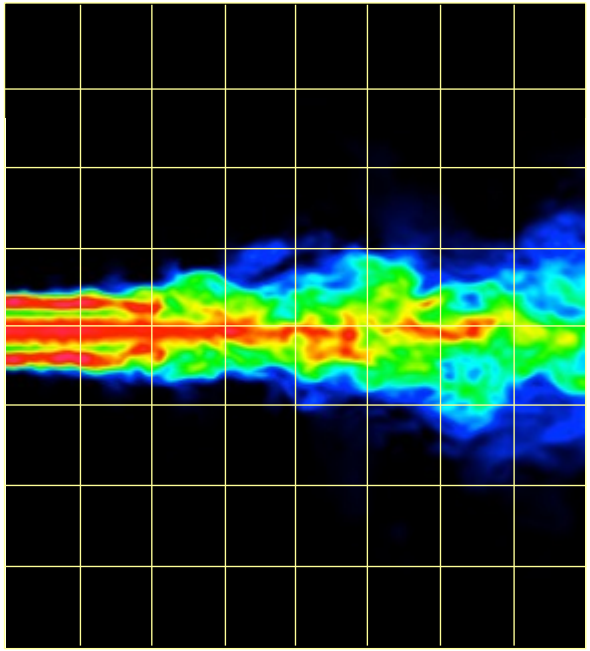


Fine-grained decomposition

Coarse-grained decomposition

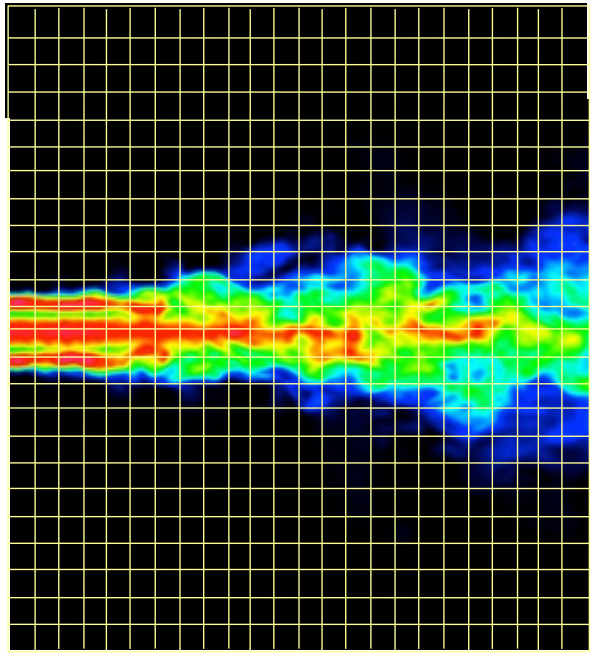
Granularity in Implementations

Coarse grid



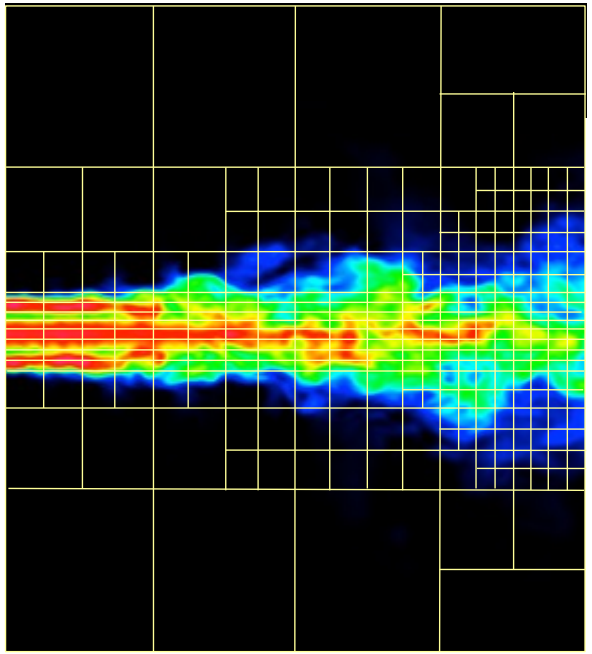
Higher Performance
Lower Accuracy
(Using Nodes)

Fine grid



Lower Performance
Higher Accuracy
(Using Processors)

Dynamic grid

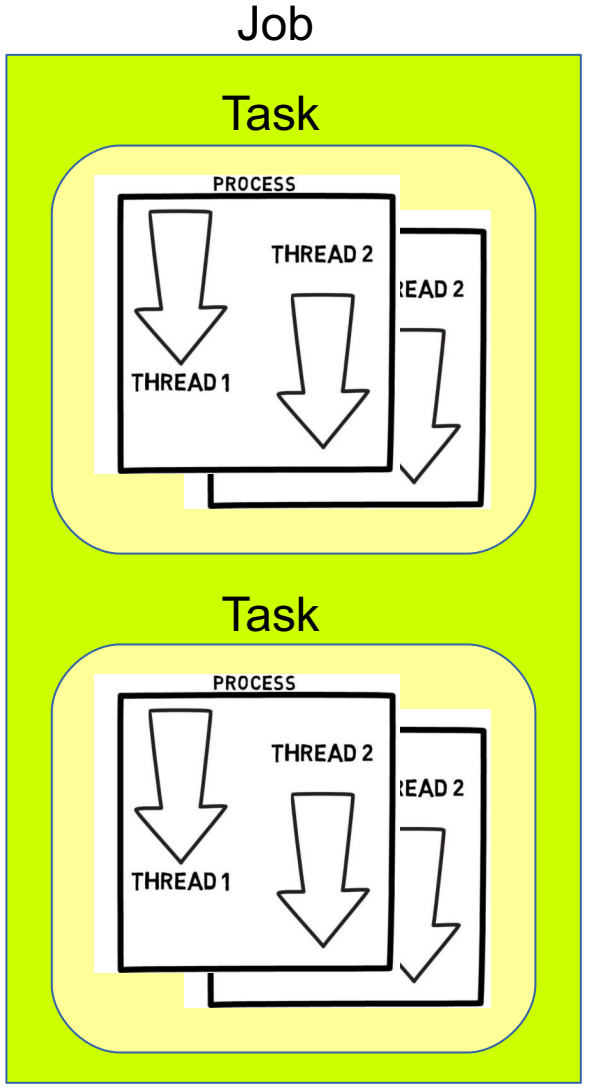


Target performance where accuracy is required (Using Processors and Nodes)

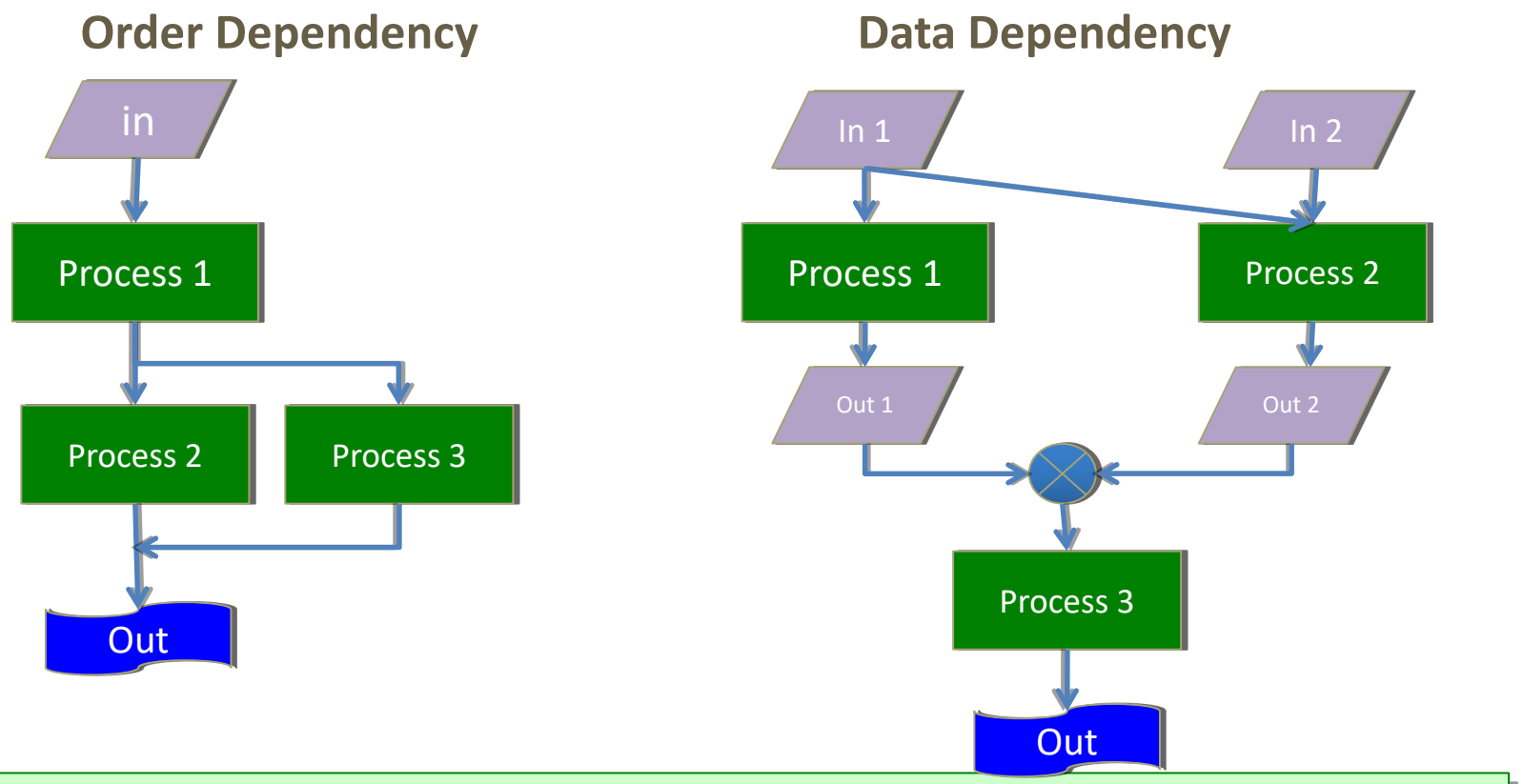
Task Decomposition Considerations

- What are the tasks and how are defined?
- What are the dependencies between task and how can they be satisfied?
- How are the task assigned to threads?

Tasks must be assigned to threads for execution



Task Dependencies

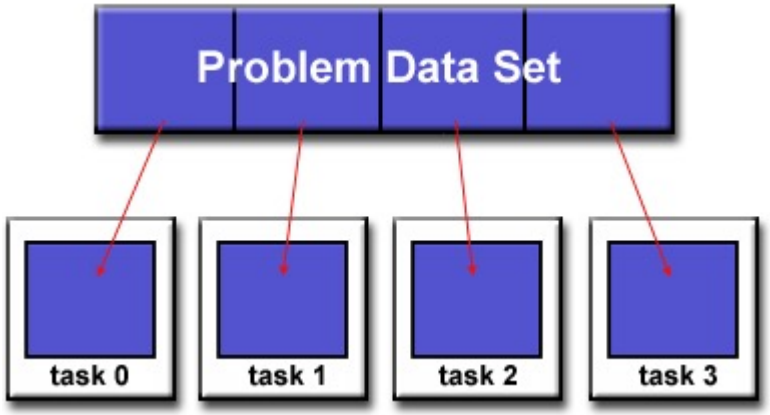


Enchantingly Parallel Code: Code without dependencies

Data Decomposition Considerations

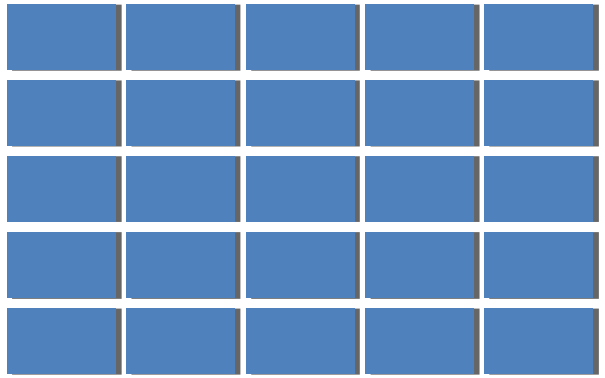
(Geometric Decomposition)

Data Structures must be (commonly) divided in arrays or logical structures.

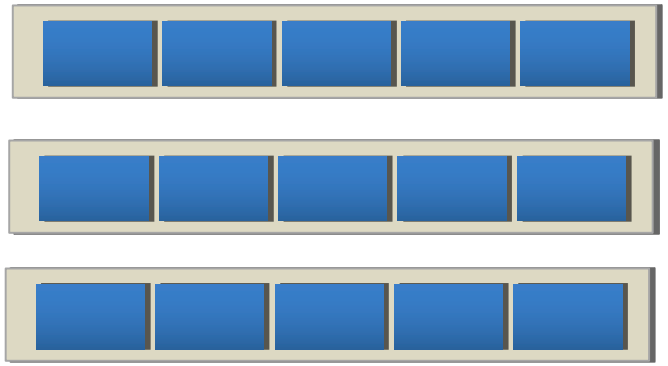


- How should you divide the data into chunks?
- How should you ensure that the tasks for each chunk have access to all data required for update?
- How are the data chunks assigned to threads?

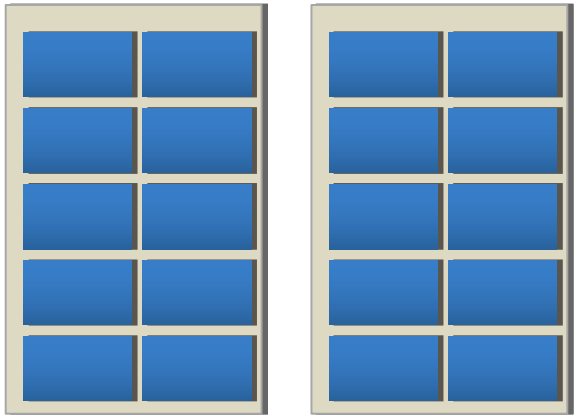
How should you divide data into chunks?



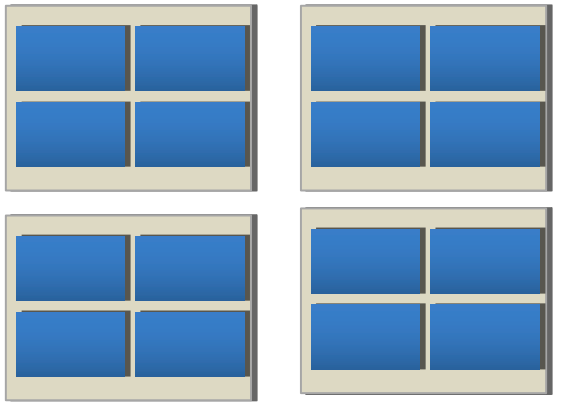
By individual elements



By rows



By groups of columns



By blocks

The Shape of the Chunk

- Data Decomposition have an additional dimension.
- It determines what the neighboring chunks are and how any exchange of data will be handled during the course of the chunk computations.

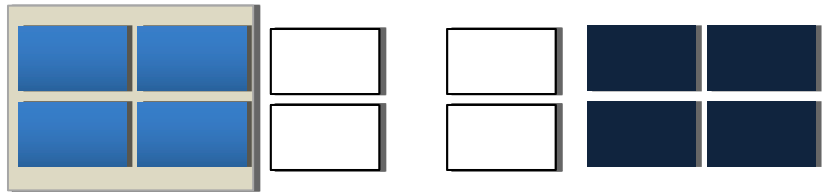
2 Shared Borders

5 Shared Borders

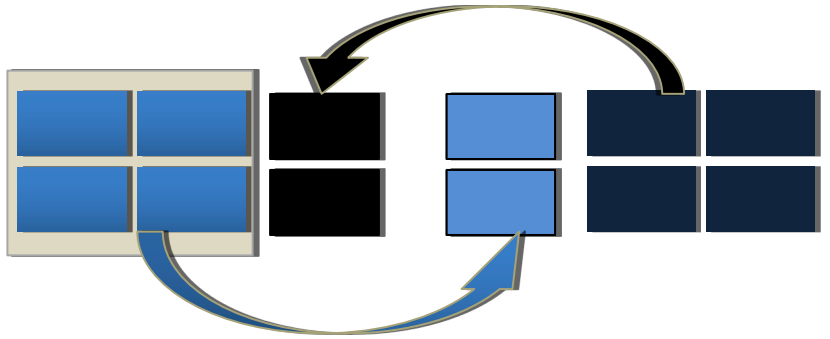
- Regular shapes : Common Regular data organizations.
- Irregular shapes: may be necessary due to the irregular organizations of the data.

How should you ensure that the tasks for each chunk have access to all data required for update?

- Using Ghost Cells
  - Using ghost cells to hold copied data from a neighboring chunk.



Original split with ghost cells



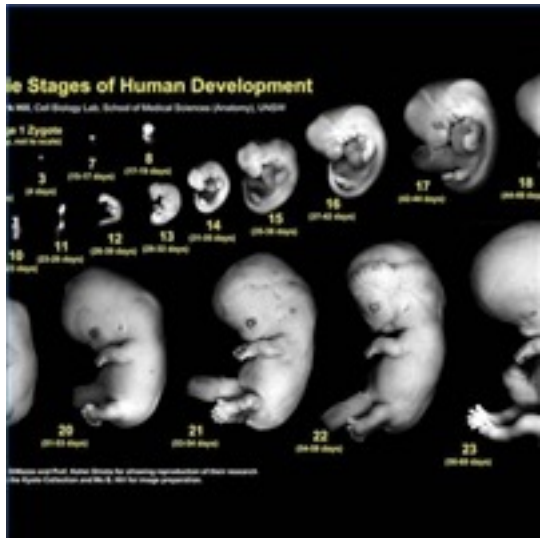
Copying data into ghost cells

Tasks and Domain Decomposition Patterns

- Task Decomposition Pattern
  - Understand the computationally intensive parts of the problem.
  - Finding Tasks (as much…)
    - Actions that are carried out to solve the problem
    - Actions are distinct and relatively independent.
- Data Decomposition Pattern
  - Data decomposition implied by tasks.
  - Finding Domains:
    - Most computationally intensive part of the problem is organized around the manipulation of large data structure.
    - Similar operators are being applied to different parts of the data structure.
  - In shared memory programming environments, data decomposition will be implied by task decomposition.

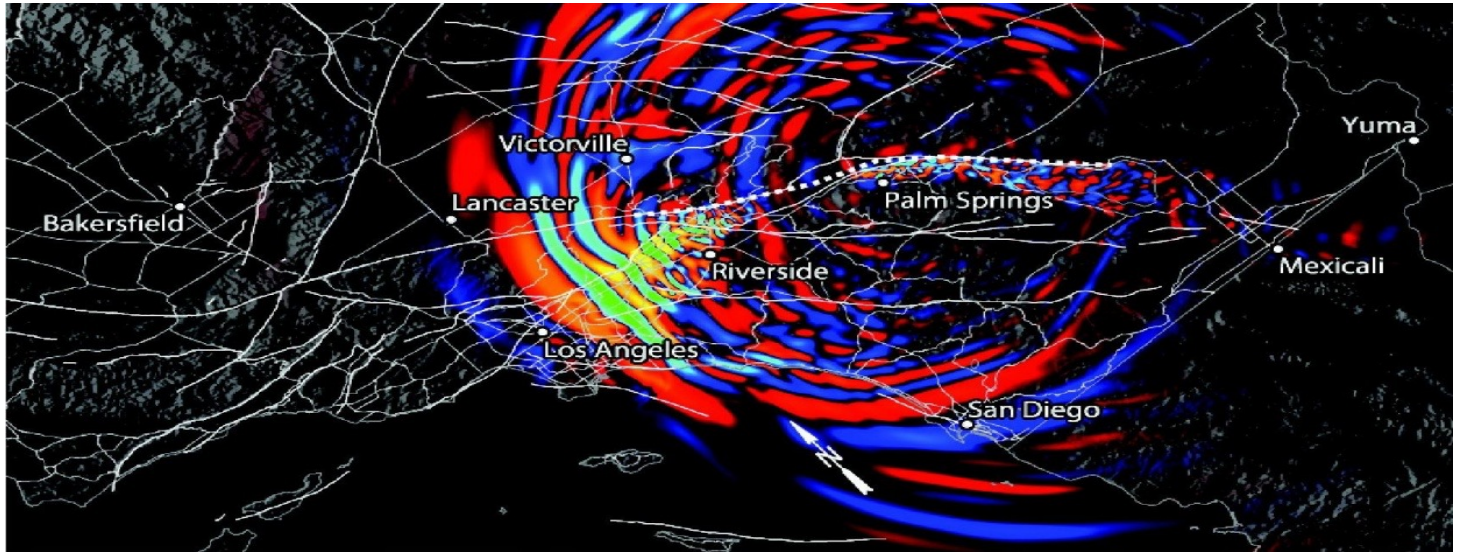
Not Parallelizable Jobs, Tasks and Algorithms

- Algorithms with state
- Recurrences
- Induction Variables
- Reductions
- Loop-carried Dependencies



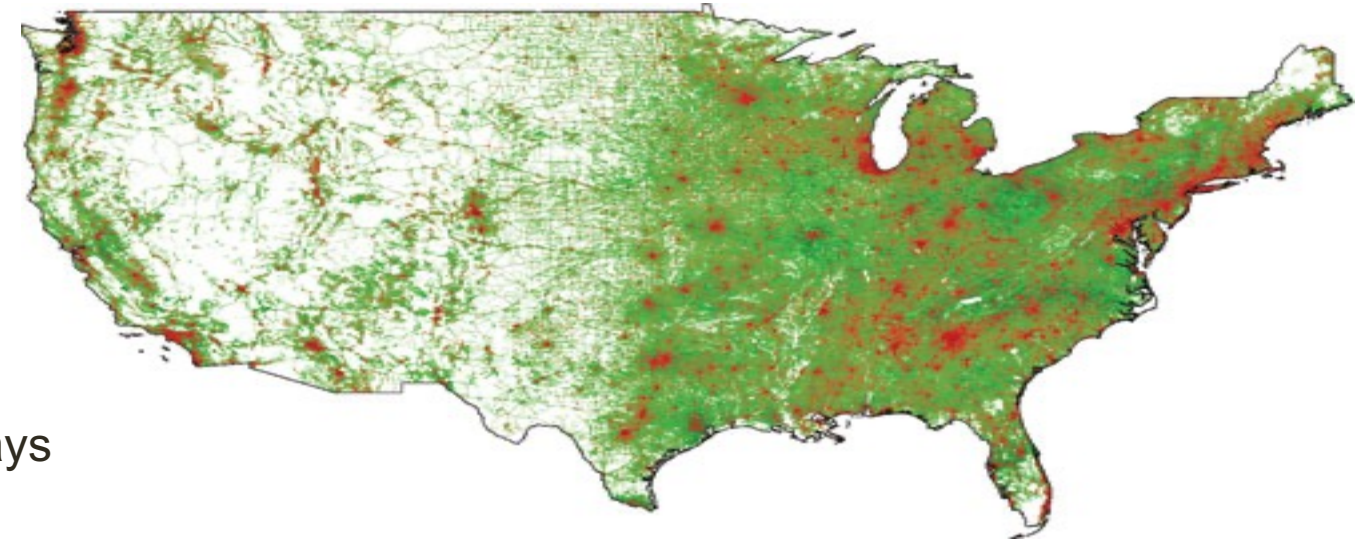
The Mythical Man-Month: Essays on Software Engineering. By Fred Brooks. Ed Addison-Wesley Professional, 1995

HPC Examples



Earthquake simulation

Surface velocity 75 sec after earthquake

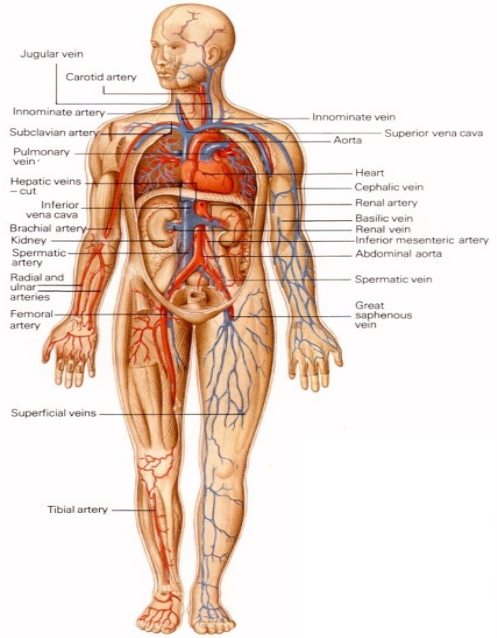


Flu pandemic simulation
300 million people tracked

Density of infected population, 45 days after breakout

HPC Examples: Blood Flow in Human Vascular Network

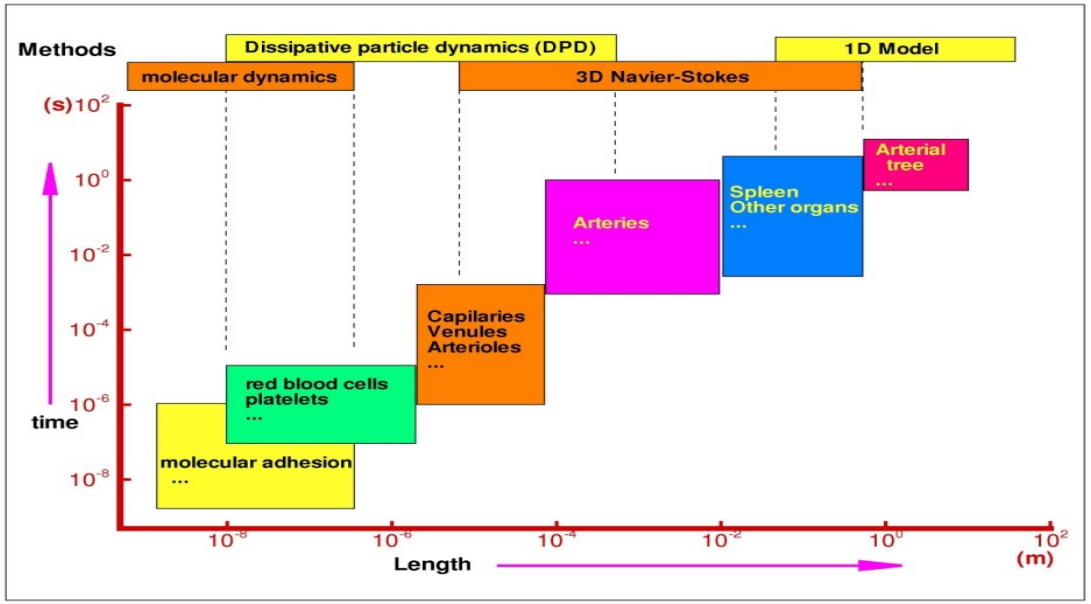
- Cardiovascular disease accounts for about 50% of deaths in western world;
- Formation of arterial disease strongly correlated to blood flow patterns;



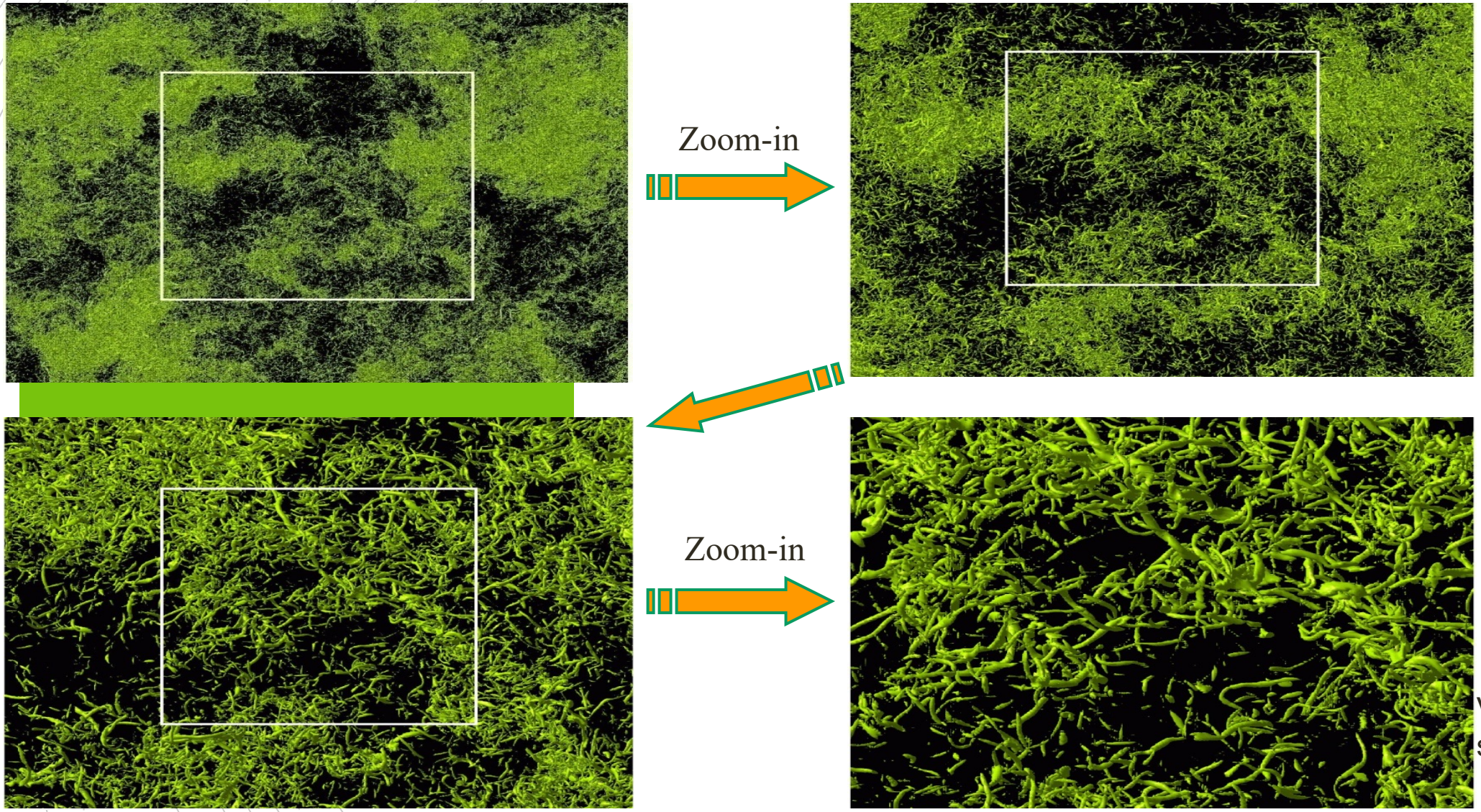
In one minute, the heart pumps the entire blood supply of 5 quarts through 60,000 miles of vessels, that is a quarter of the distance between the moon and the earth

Computational challenges: Enormous problem size

Blood flow involves multiple scales



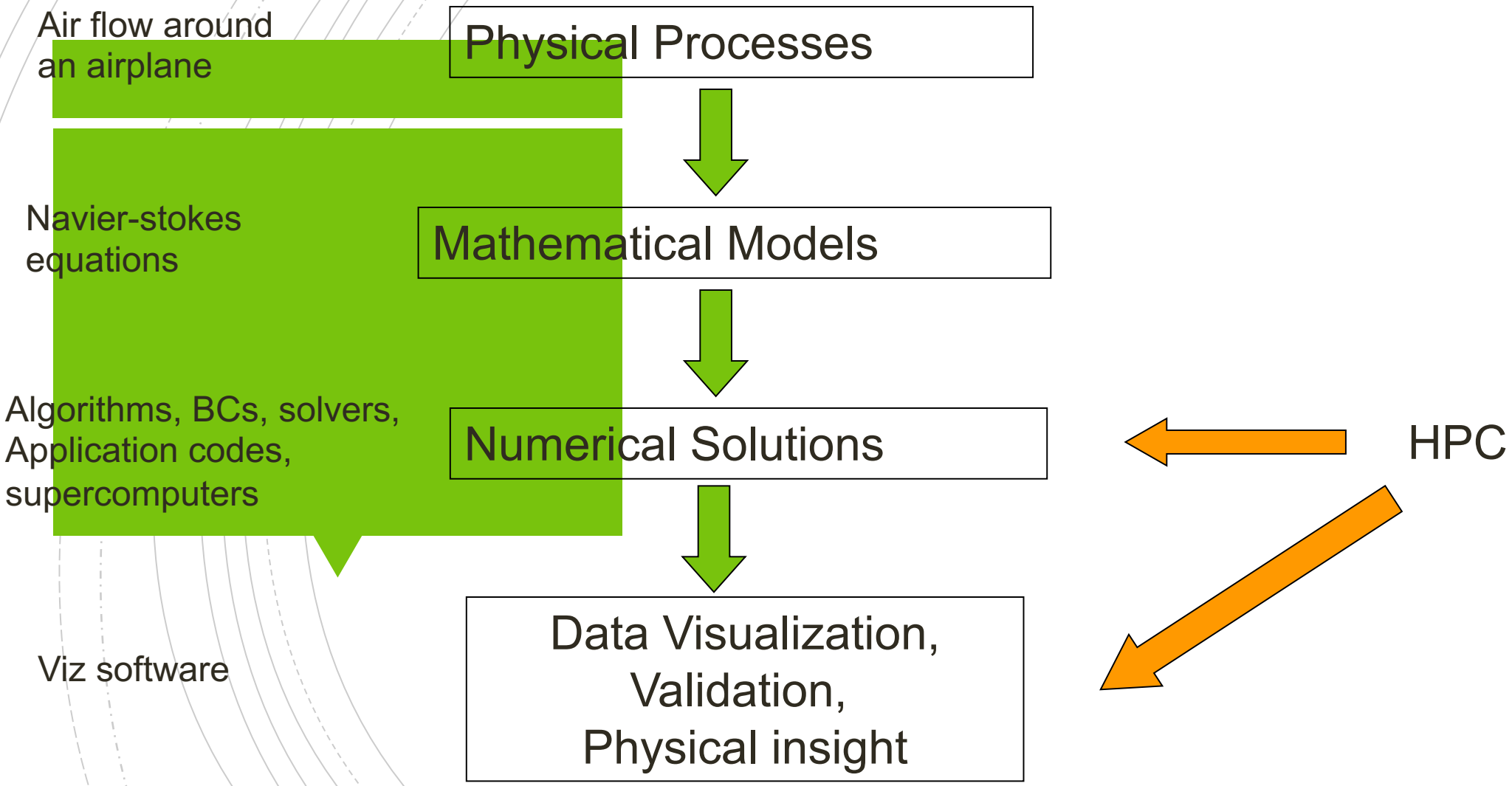
HPC Example: Homogeneous Turbulence



Vorticity iso-surface

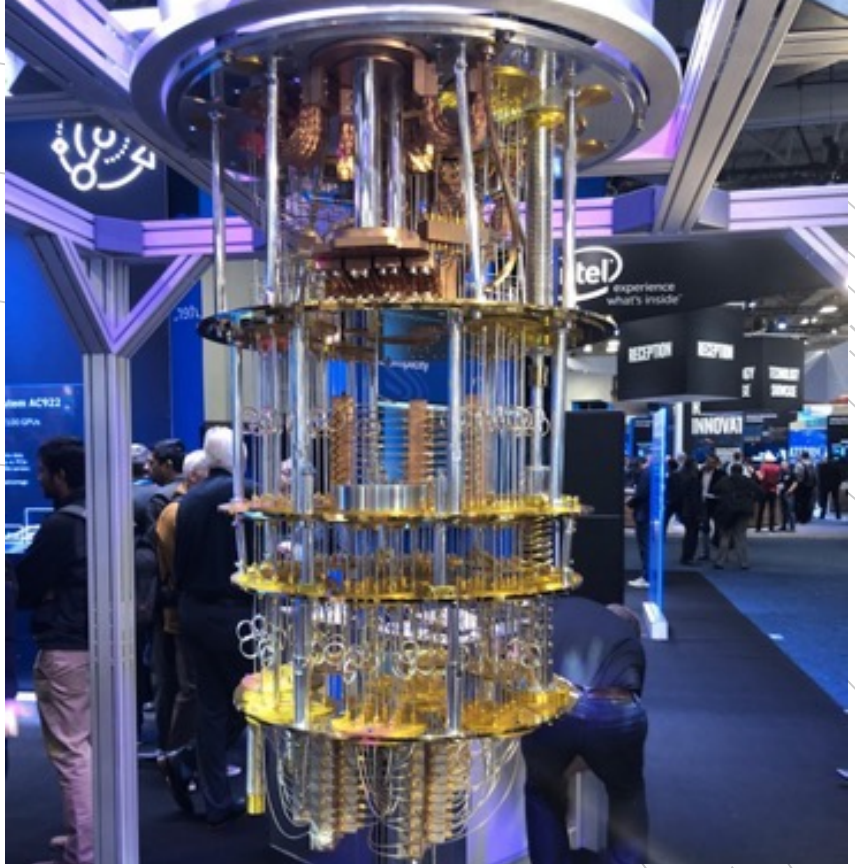
Direct Numerical Simulation of Homogeneous Turbulence: 4096^3

How HPC fits into Scientific Computing



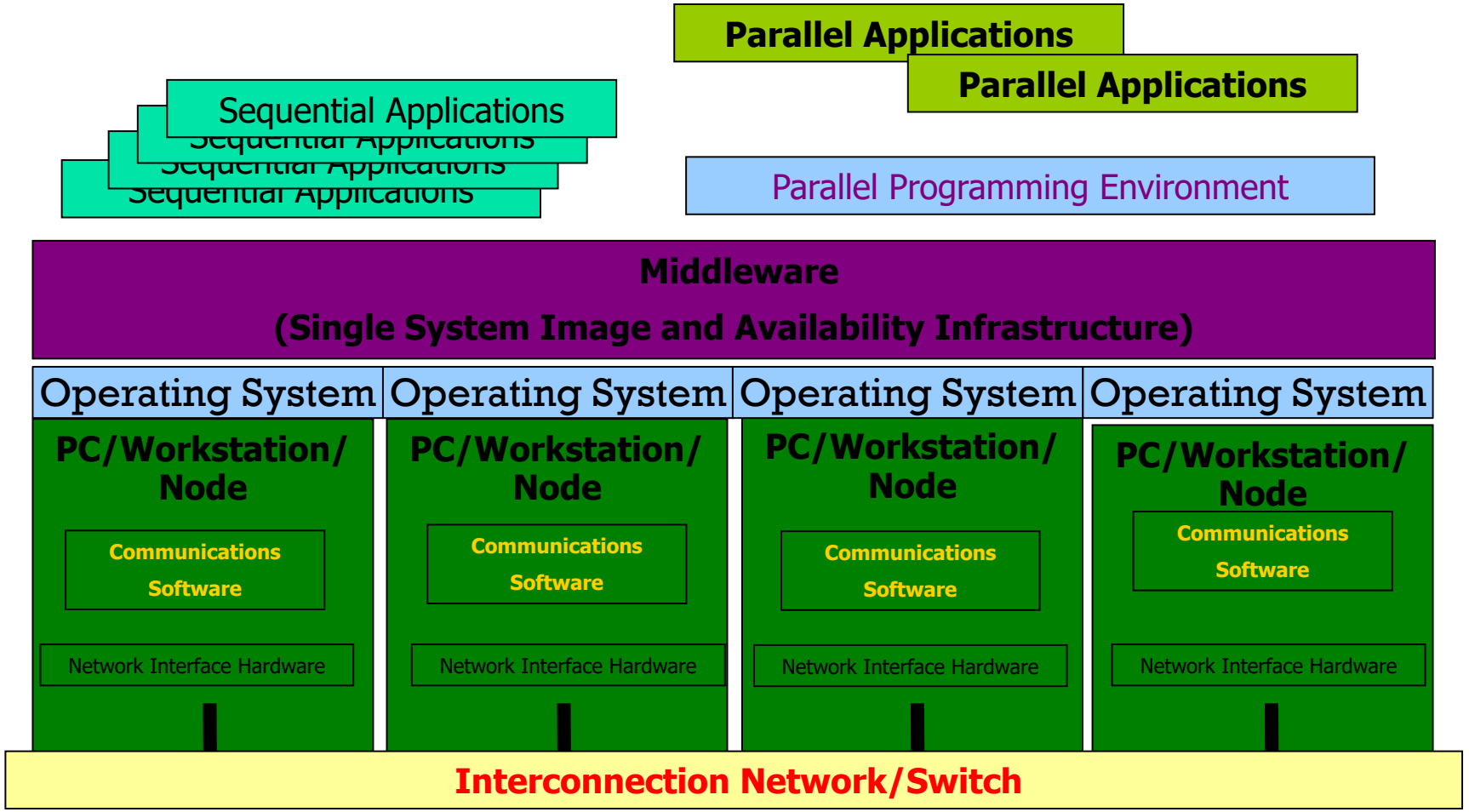
Advantages of Parallelization

- Cheaper, in terms of Price/Performance Ratio
- Faster than equivalently expensive uniprocessor machines
- Handle bigger problems
- More scalable: the performance of a particular program may be improved by execution on a large machine
- More reliable: In theory if processors fail we can simply use others



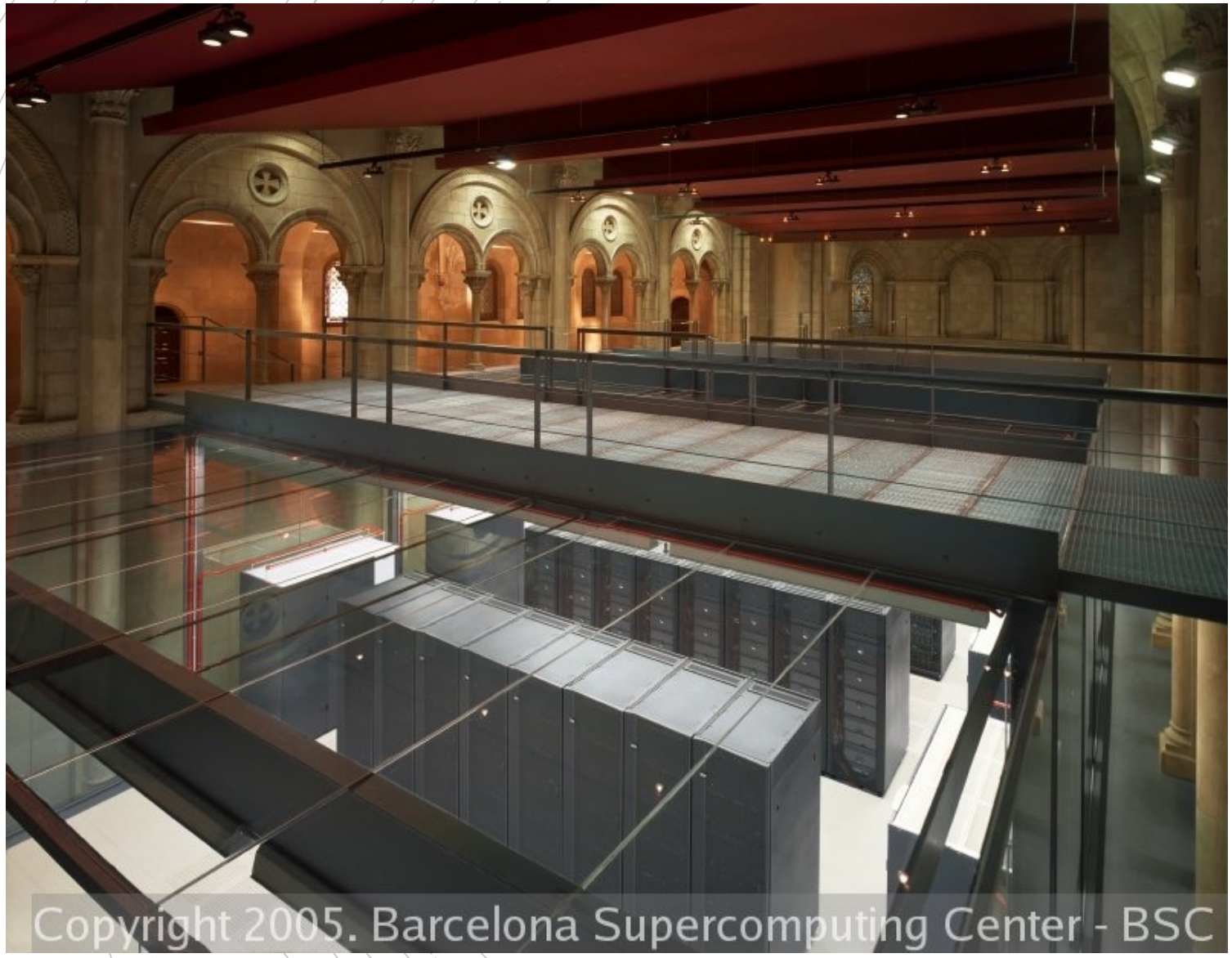
Now, a Little of Advanced Computing Architecture

Cluster Computing Architecture



The Spain Exemple: BSC-CNS Marenostrum

www.bsc.es

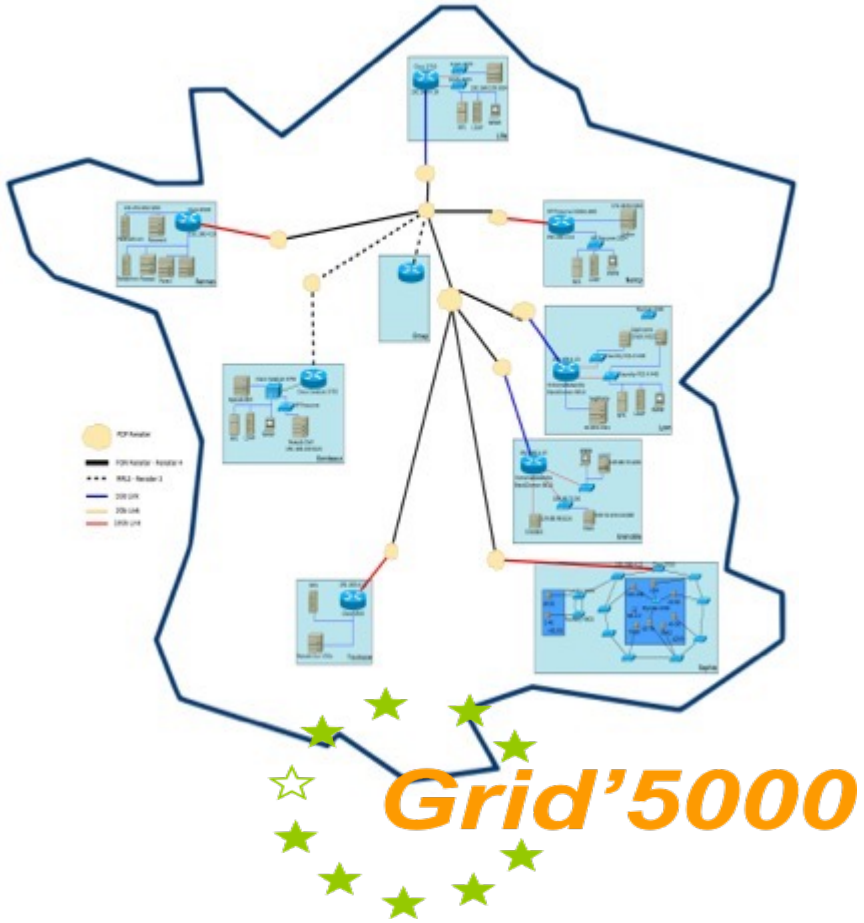


- 11.15 Petaflops
- 384.75 TB Memory
- 3.465 Computing Nodes
  - 2x Intel Xeon Plantium 8160 24C /2.1Ghz
  - 216 Nodes with 12x32GB DDR4 2667 DIMMS (8GB Cores)
  - 3240 Nodes with 12x8GB DDR4-2667 DIMMS (2GB Cores)
- Network
  - 100Gb Intel Omni-Path Full Fat Tree
  - 100Gb Ethernet
- Operating System
  - Suse Linux Enterprise Server 12SP2

Grid Computing

- Grid Computing implies technology, technics and methodology to support Parallel*/Distributed Computing.
- Grid Computing needs Grid Computing Infrastructure and dedicated and high disponibility networks or interconexion.
- Different Types or Possibilities:
  - Experimental Testbeds
  - Production Grids
  - Lightweigth Grids
  - Desktop Grid Computing (May be Lightweigth too)
- **Grid Computing is in the back of Cloud Computing Systems (from Infrastructure Point of View)**

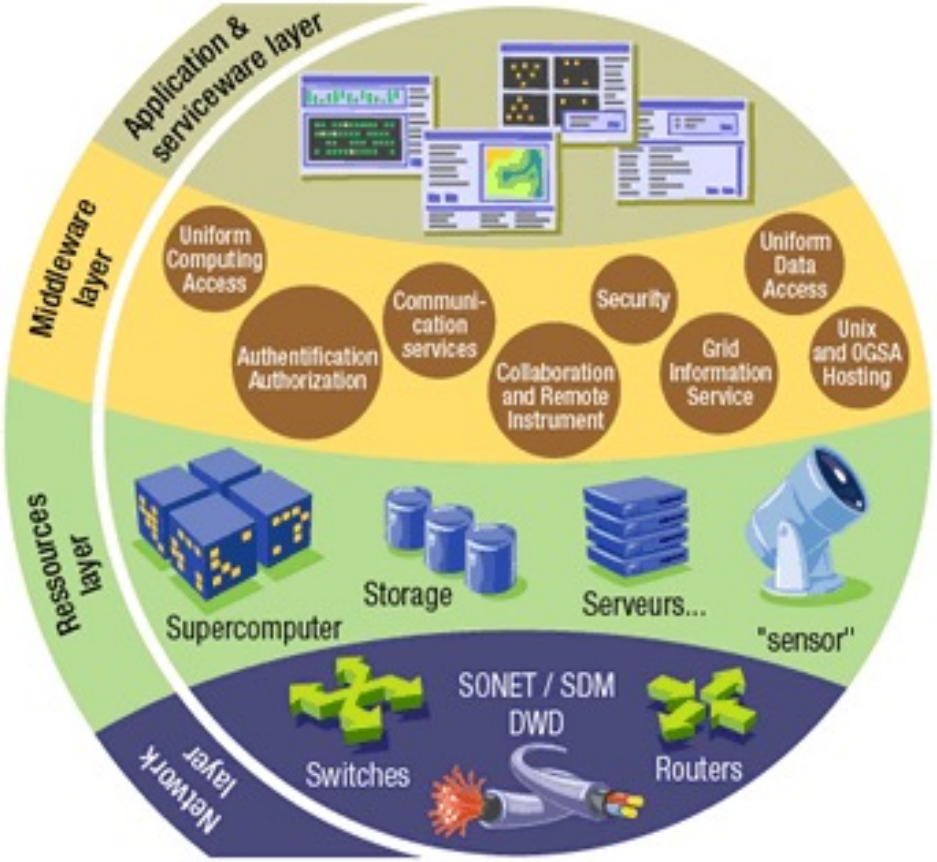
Grid Computing Feautures

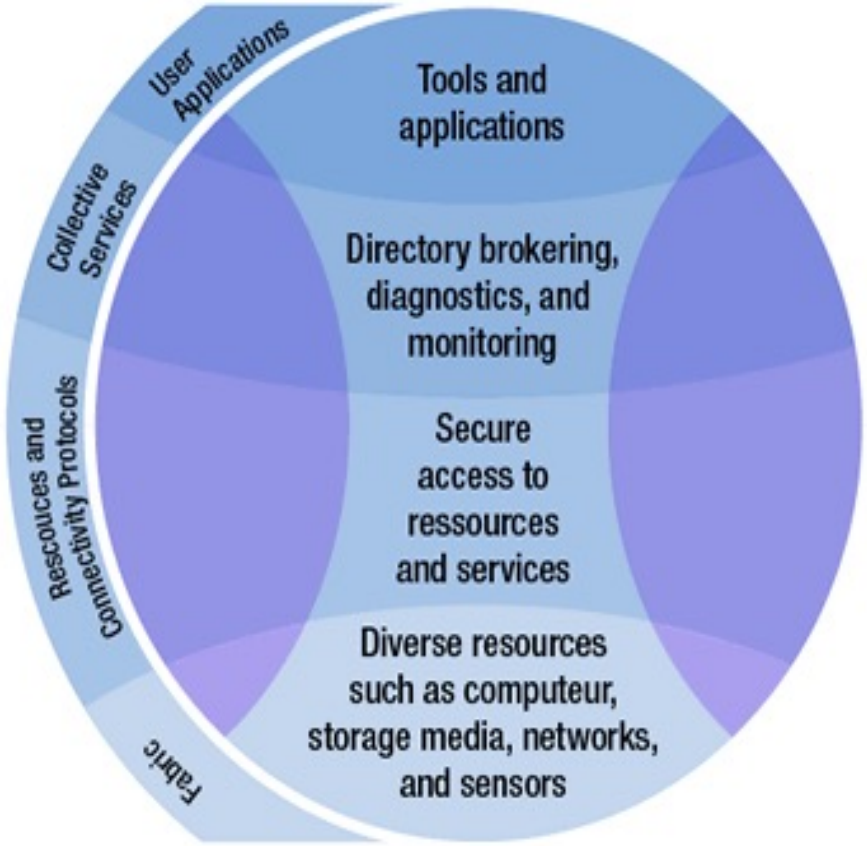


http://www.grid5000.fr

- Grid Computing Features:
- Infrastructure
  - High Availability
  - High Performance
  - Heterogeneity
  - Pervasive
  - Scalability
- Methodology
  - Different User Levels
  - Multi Administration
- Politics
  - Security
  - Use
  - Privacy

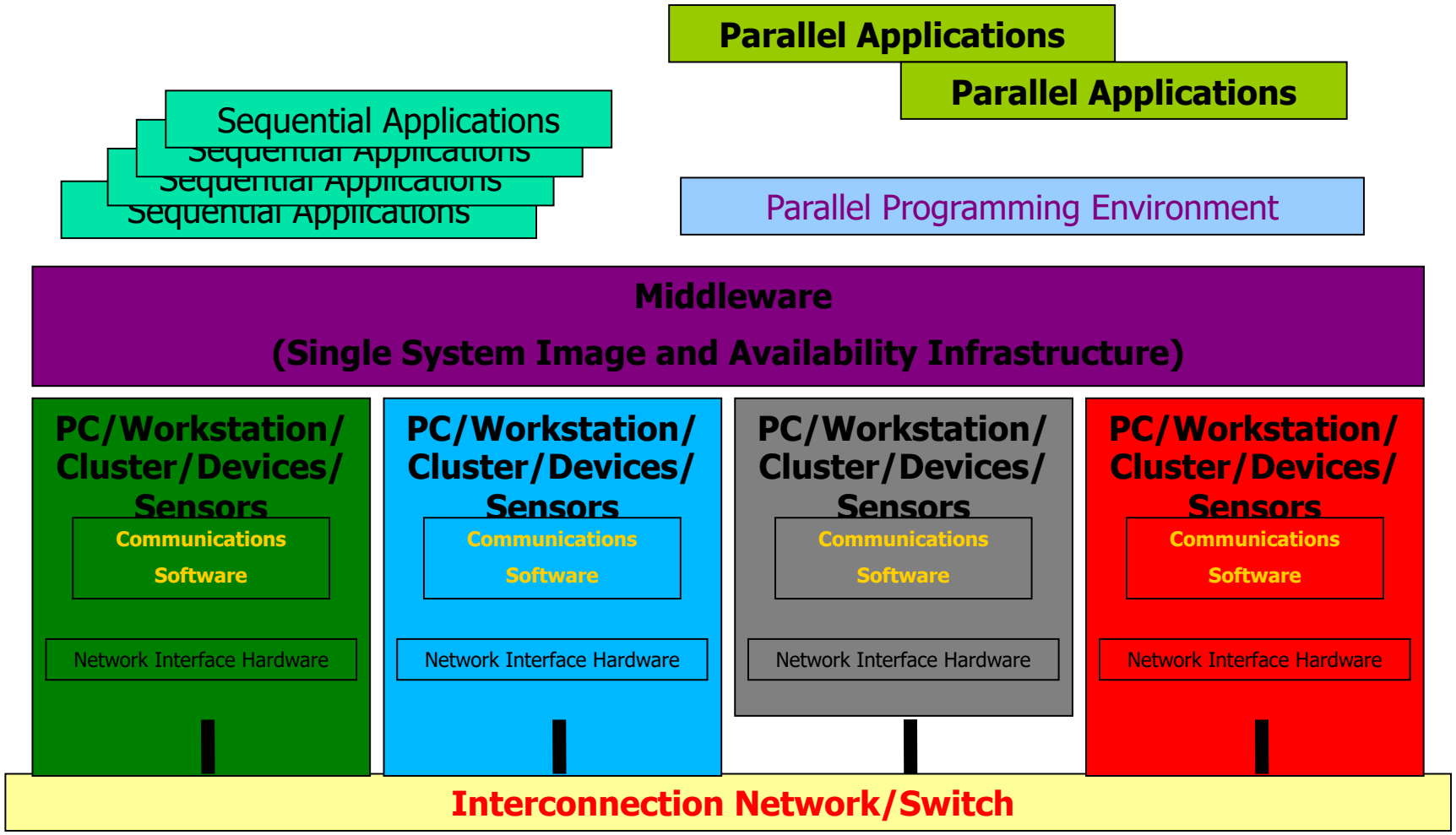
Grid Computing Architecture (Typical Diagram)





[*]From http://gridcafe.web.cern.ch

Grid Computing Architecture (Remember the Cluster Architecture)



An Example: The French Aladdin Grid5000 (G5K)

- G5K has 5000 processors distributed in 9 sites France wide, for research in Grid Computing, eScience and Cyber-infrastructures
- G5K project aims at building a highly reconfigurable, controlable and monitorable experimental Grid platform
- All clusters will be connected to Renater with a 10Gb/s link (or at least 1 Gb/s, when 10Gb/s is not available yet).
  - IntraCluster
    - Myrinet
    - GigaEthernet / Infiniband
  - Grid
    - Giga Ethernet (Best case 10GB/s, Nate case: 1GB/s)
  - Inter-Grid
    - Ethernet (~1GB/s)

Grid'5000 Aladdin

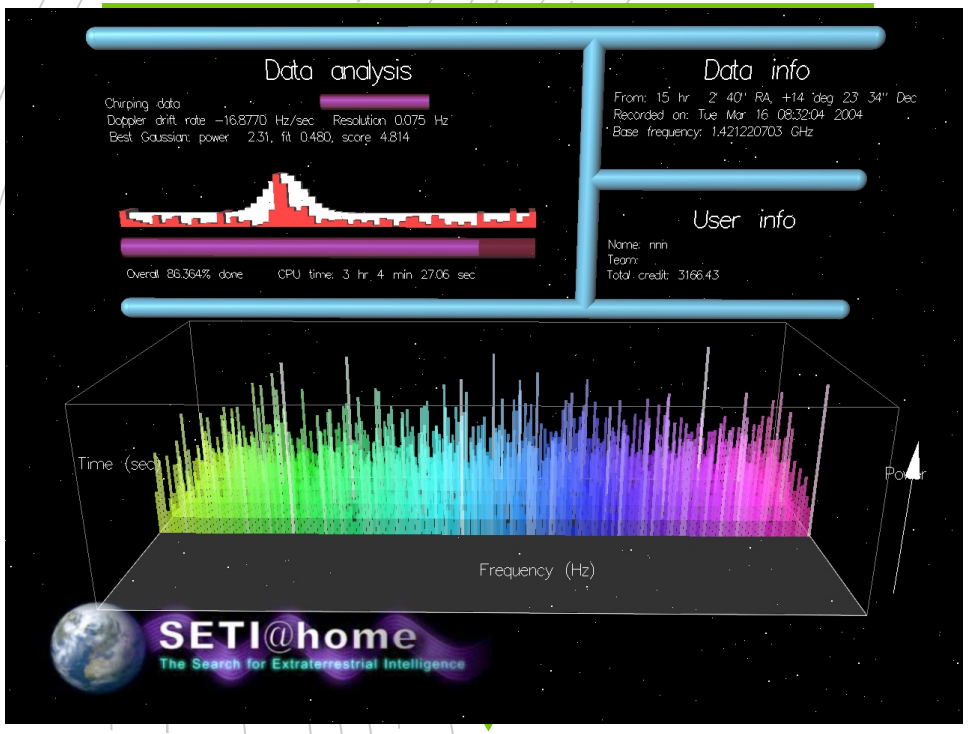
Grid'5000

www.grid5000.org

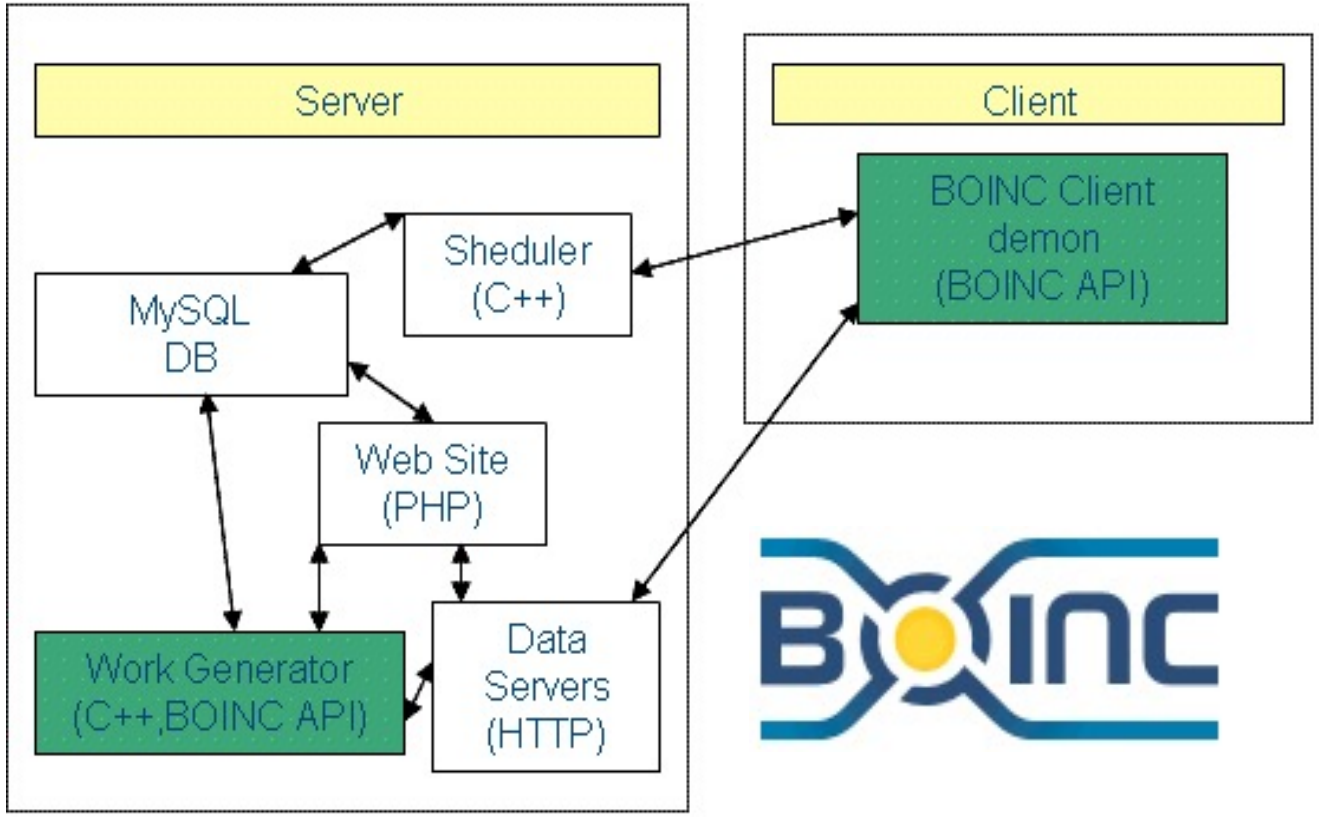
Volunteer Computing

- Volunteer computing is a type of distributed computing in which computer owners donate their computing resources (such as processing power and storage) to one or more "projects".
  - BOINC (Seti@home)
  - Xgrid
  - GridMP
- Associated with P2P
- Can be associated with High Throughput Computing (HTC) or High Performance Computing (HCP)

An Example: The BOINC Architecture



https://boinc.berkeley.edu/



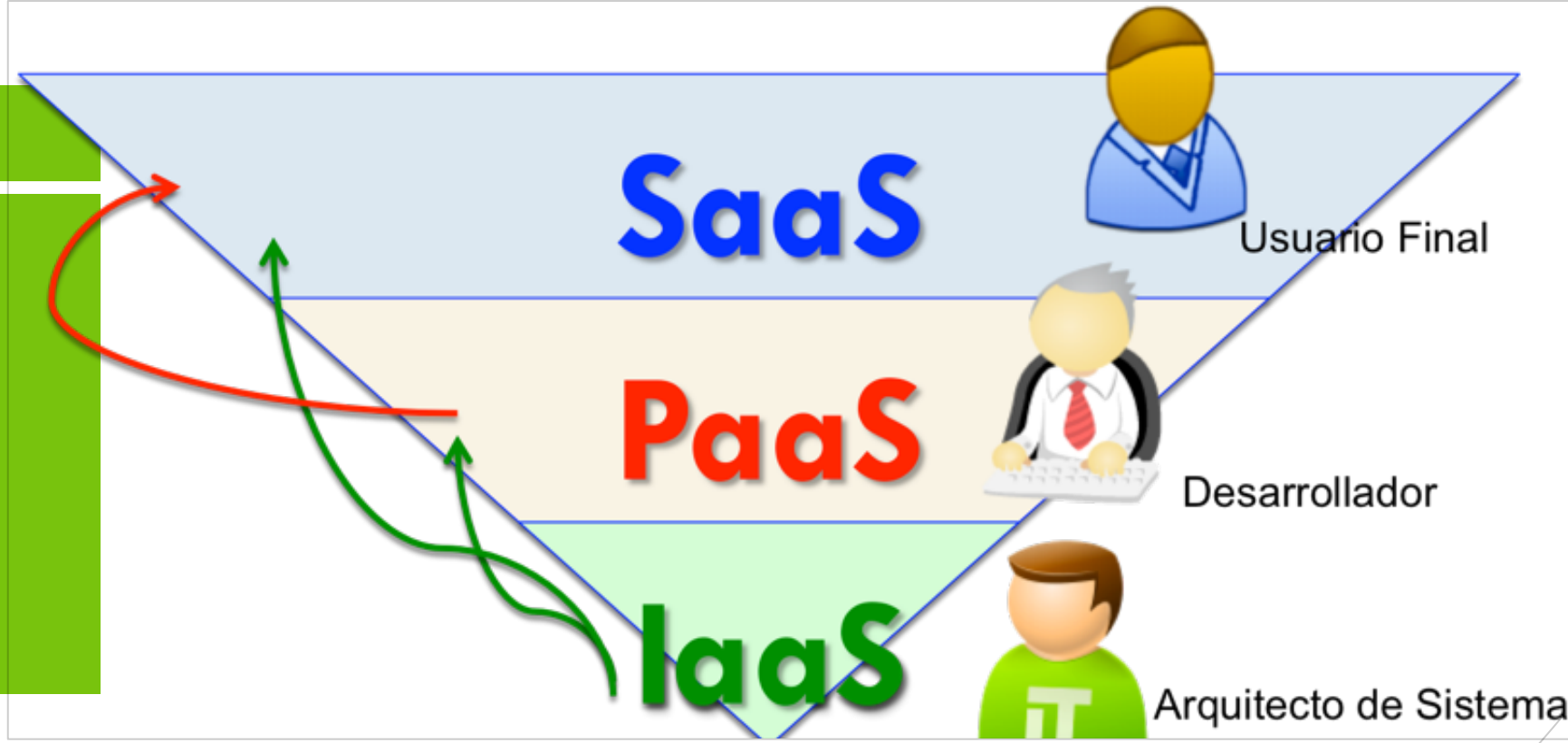
Cloud Computing

Internet-based computing, whereby shared resources, software, and information are provided to computers and other devices on demand.

Cloud computing describes a new supplement, consumption, and delivery model for IT services based on the Internet, and it typically involves over the-Internet-provision of dynamically scalable and often virtualized resources

Logical-Services Cloud View

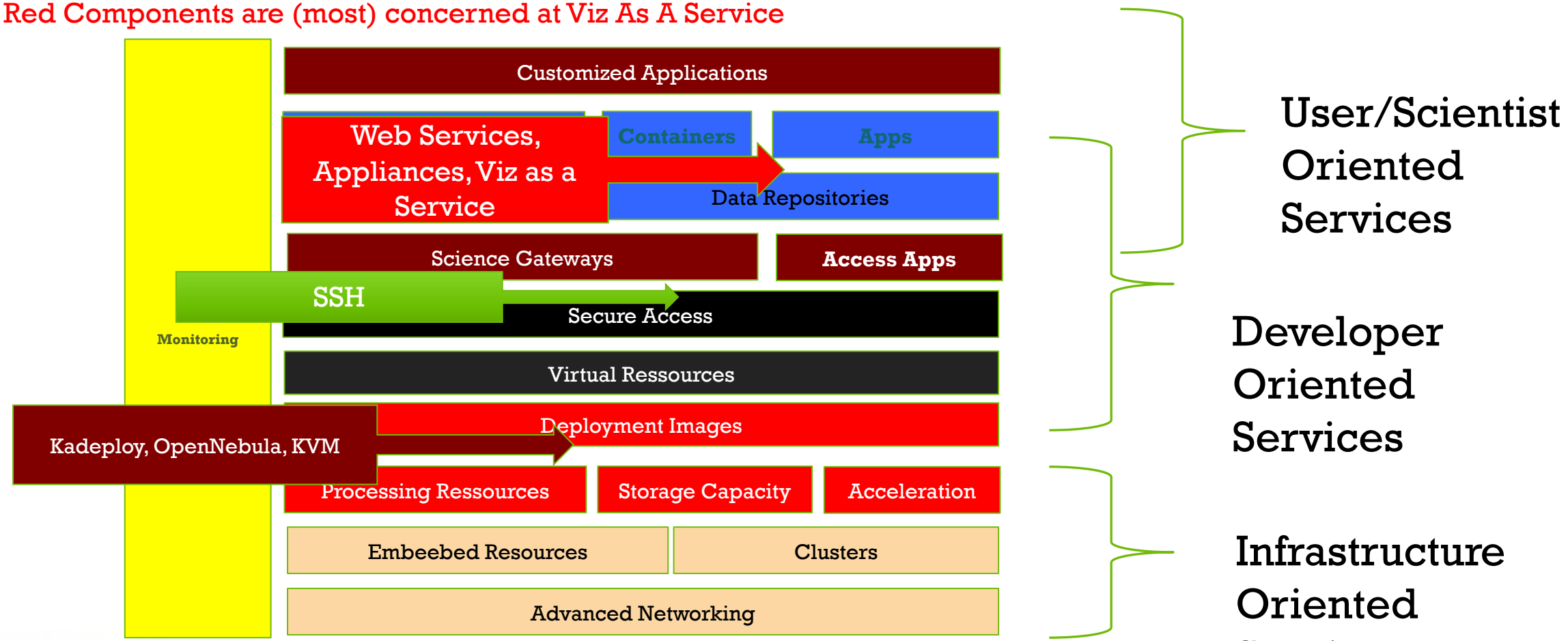
Cloud Computing Visibility



- **Visit:** http://prezi.com/i0sretldeyk7/computacion-en-la-nube-y-sus-implicaciones-para-la-industria-del-software-en-colombia/

How Exploit HPC Architectures with Cloud Visibility Models?
HPC as A Service Model

* Red Components are (most) concerned at Viz As A Service

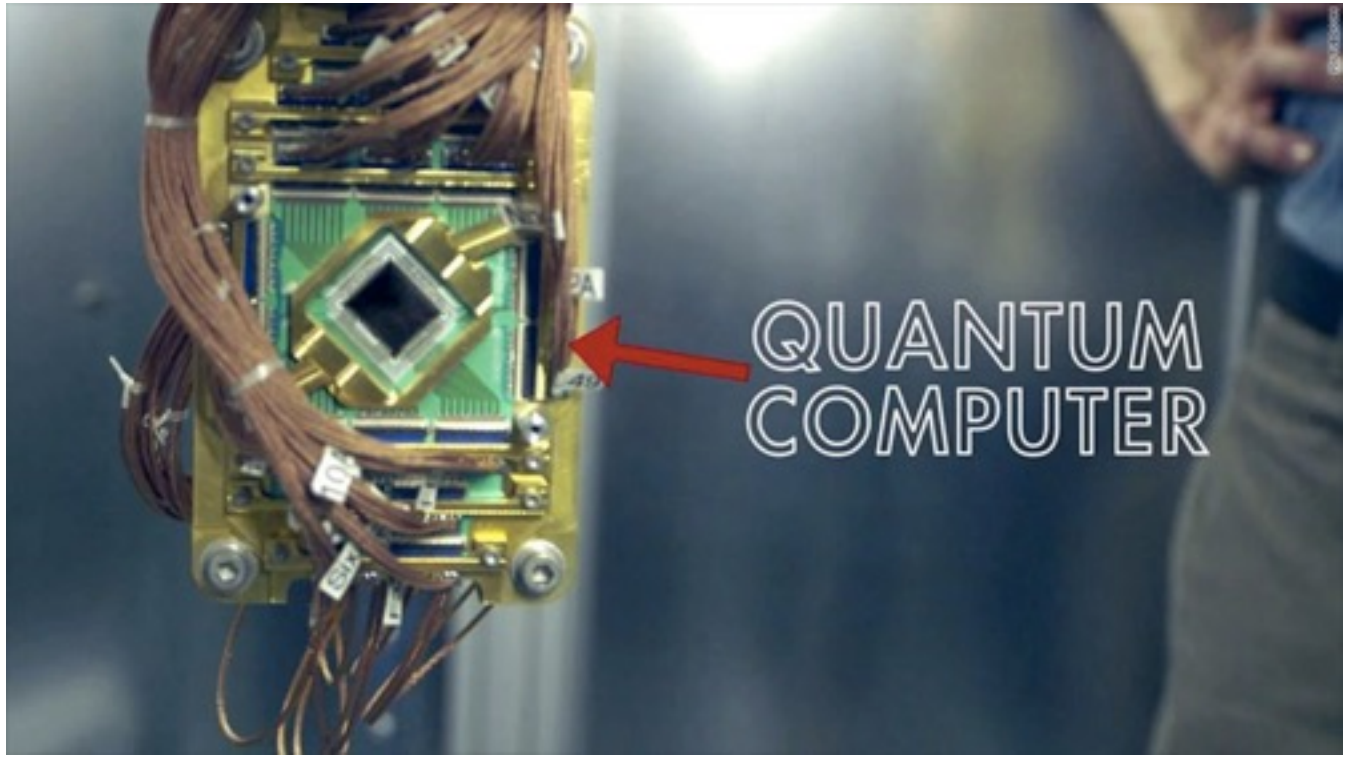


Universidad Industrial de Santander
CONSTRUIMOS FUTURO
Super Computación y Cálculo Científico UIS
Universidad de los Andes
Inria
SCALA
RICAP
NVIDIA

And… Quantum computing?

(Advanced Computing Point of view)

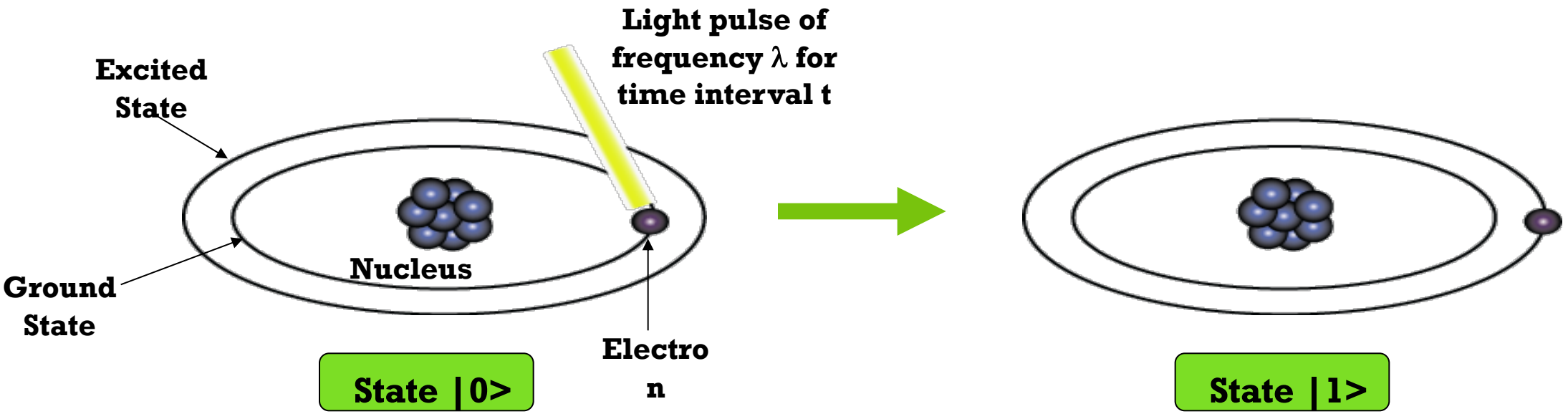
- A quantum computer is a machine that performs calculations based on the laws of quantum mechanics, which is the behavior of particles at the sub-atomic level.



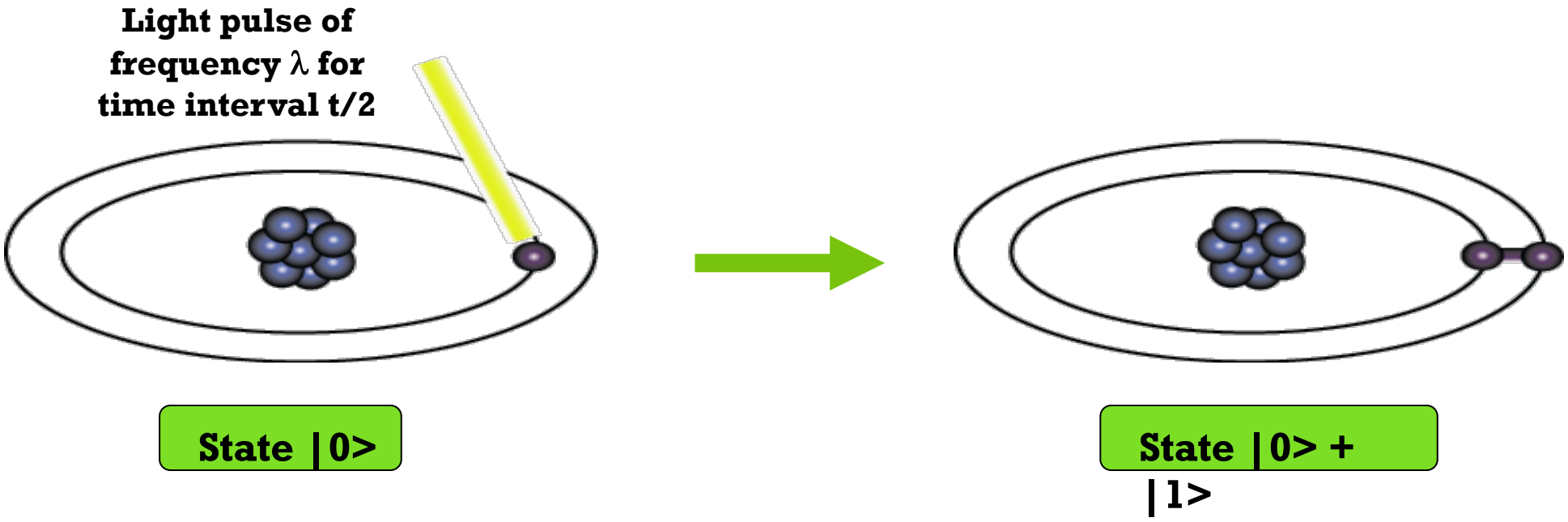
Representation of Data - Qubits

A bit of data is represented by a single atom that is in one of two states denoted by **|0>** and **|1>**. A single bit of this form is known as a ***qubit***

A physical implementation of a qubit could use the two energy levels of an atom. An excited state representing |1> and a ground state representing |0>.



Representation of Data - Superposition



- Consider a 3 bit qubit register. An equally weighted superposition of all possible states would be denoted by:

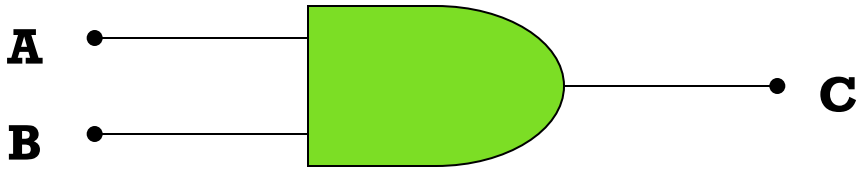
$$|\psi\rangle = \frac{1}{\sqrt{8}}|000\rangle + \frac{1}{\sqrt{8}}|001\rangle + \ldots + \frac{1}{\sqrt{8}}|111\rangle$$

Operations on Qubits - Reversible Logic

- Due to the nature of quantum physics, the destruction of information in a gate will cause heat to be evolved which can destroy the superposition of qubits.

Ex.

The AND Gate



Input		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

In these 3 cases, information is being destroyed

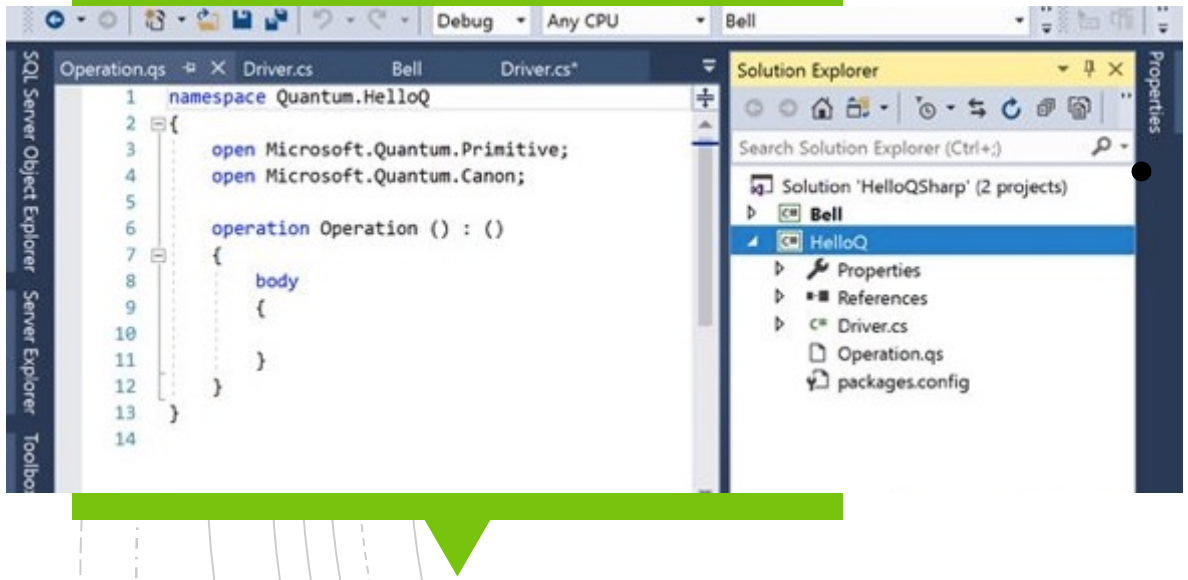
- This type of gate cannot be used. We must use ***Quantum Gates***.

Quantum Computers Today

Enterprises produce Quantum Computing Laboratory Infrastructure (Non for production)

- (Real) Quantum Computing (D-Wave, IBM)
- Quantum Computing Simulators (Atos)

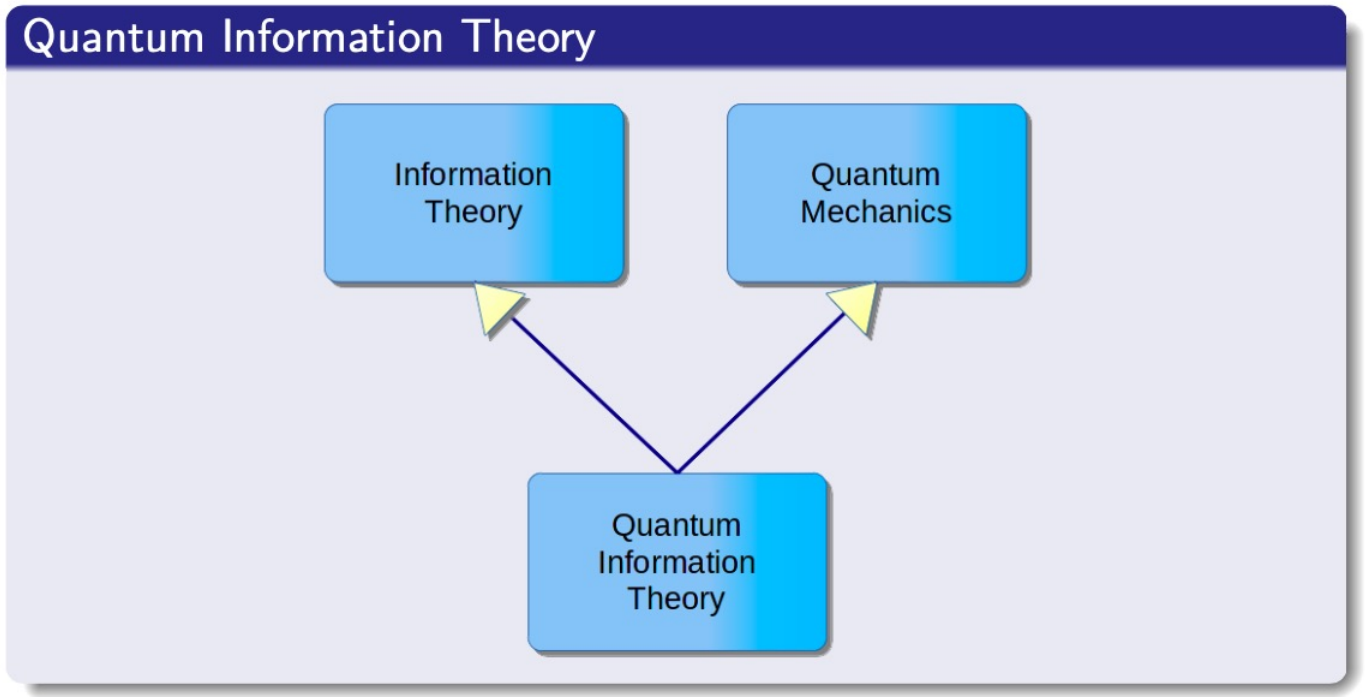
About Q… Algorithms and Code…

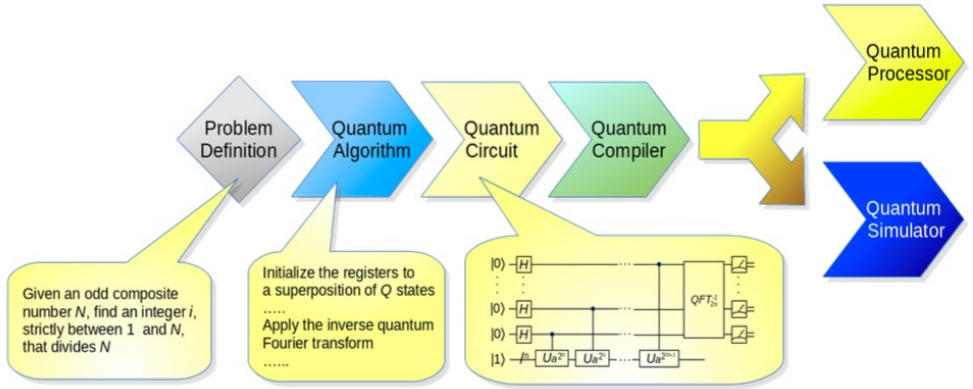


Q# Interface

- Consortiums propose Quantum Frameworks
  - IBM, Microsoft, ATOS
  - Quantum Computing Community…
    - However without a good use fo real Quantum mathematical abstraction.

Quantum Computing Theory for Quantum Computing Applications



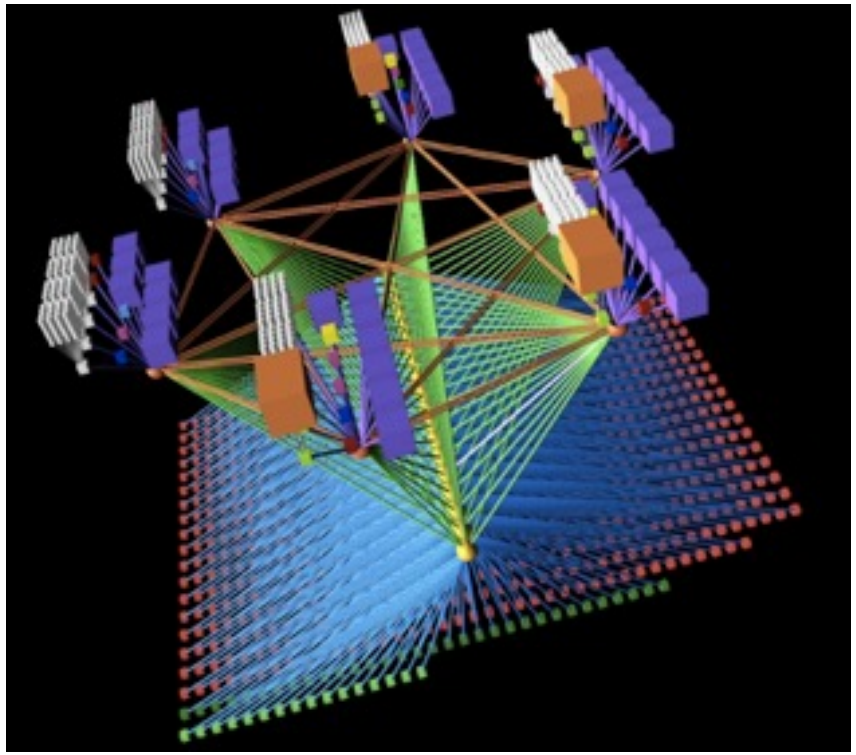


The term **quantum algorithm** is generally used for those algorithms that incorporate some essential feature of **quantum computing**, such as superposition or entanglement. By using this special features, we can speed up significantly the calculation, that is called **quantum parallelism**.

Université de Reims Champagne-Ardenne · CReSTIC · Universidad Industrial de Santander · Super Computación y Cálculo Científico UIS · CAGE · Computo Avanzado y a Gran Escala Advanced and Large Scale Computing Research group

G. Diaz PhD. Thesis about Classica Resources Consumption in Quantum Computing Simulators (Co-Advising with L. A. Steffenel)

The Challenges in Detail: Post Moore Era Architectures



A post-moore architecture schema

Sustainable-Hybrid Technology

- RISC/CISC
  - GPUs, Hybrid ARM/FPGAs, Accelerators, CPUS….
  - I/O's and Memory Management

The "Data Treatment" Goal

- Large Scale Data Sets (Supported by the Architecture
- However scale capabilities changes
- In-Situ and In-Transit Problem

Very Known Schedulers, O.S. and Package Management

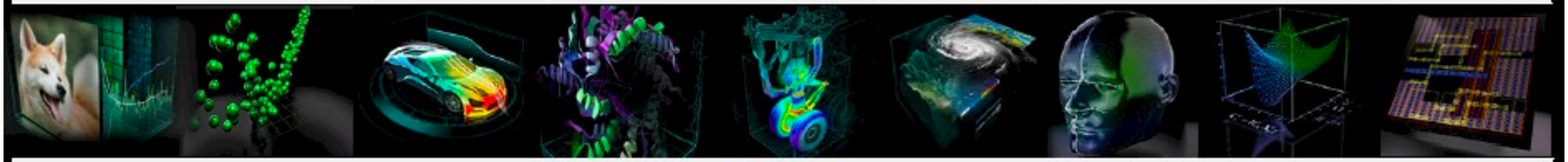
- However, it is important to observe the architecture

Exascale constrains

- Computer Efficiency (Energy Consumption / Energy Aware)

Software To Deliver Acceleration For HPC & AI Apps; 500+ New Updates

Machine Learning & Deep Learning	Computational Physics & Chemistry	Computational Fluid Dynamics	Life Sciences & Bioinformatics	Structural Mechanics	Weather & Climate	Geoscience, Seismology & Imaging	Numerical Analytics	Electronic Design Automation



600+ Apps

Linear Algebra | Parallel Algorithms | Signal Processing | Deep Learning | Machine Learning | Visualization

CUDA-X HPC & AI

40+ GPU Acceleration Libraries

CUDA

Desktop Development | Data Center | Supercomputers | GPU-Accelerated Cloud

www.nvidia.com

Concurrent Design Models Features

- **Efficiency**
  - Concurrent applications must run quickly and make good use of processing resources.
- **Simplicity**
  - Easier to understand, develop, debug, verify and maintain.
- **Portability**
  - In terms of threading portability.
- **Scalability**
  - It should be effective on a wide range of number of threads and cores, and sizes of data sets.

Design Evaluation Pattern

- Production of analysis and decomposition:
  - Task decomposition to identify concurrency
  - Data decomposition to indentify data local to each task
  - Group of task and order of groups to satisfy temporal constraints
  - Dependencies among tasks
- Design Evaluation
  - Suitability for the target platform
  - Design Quality
  - Preparation for the next phase of the design

Algorithm Structures

Organizing by Tasks

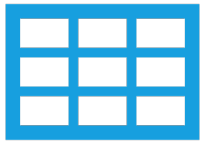
Task Parallelism

Divide and Conquer

Organizing by Data Decomposition

Geometric Decomposition

Recursive Data

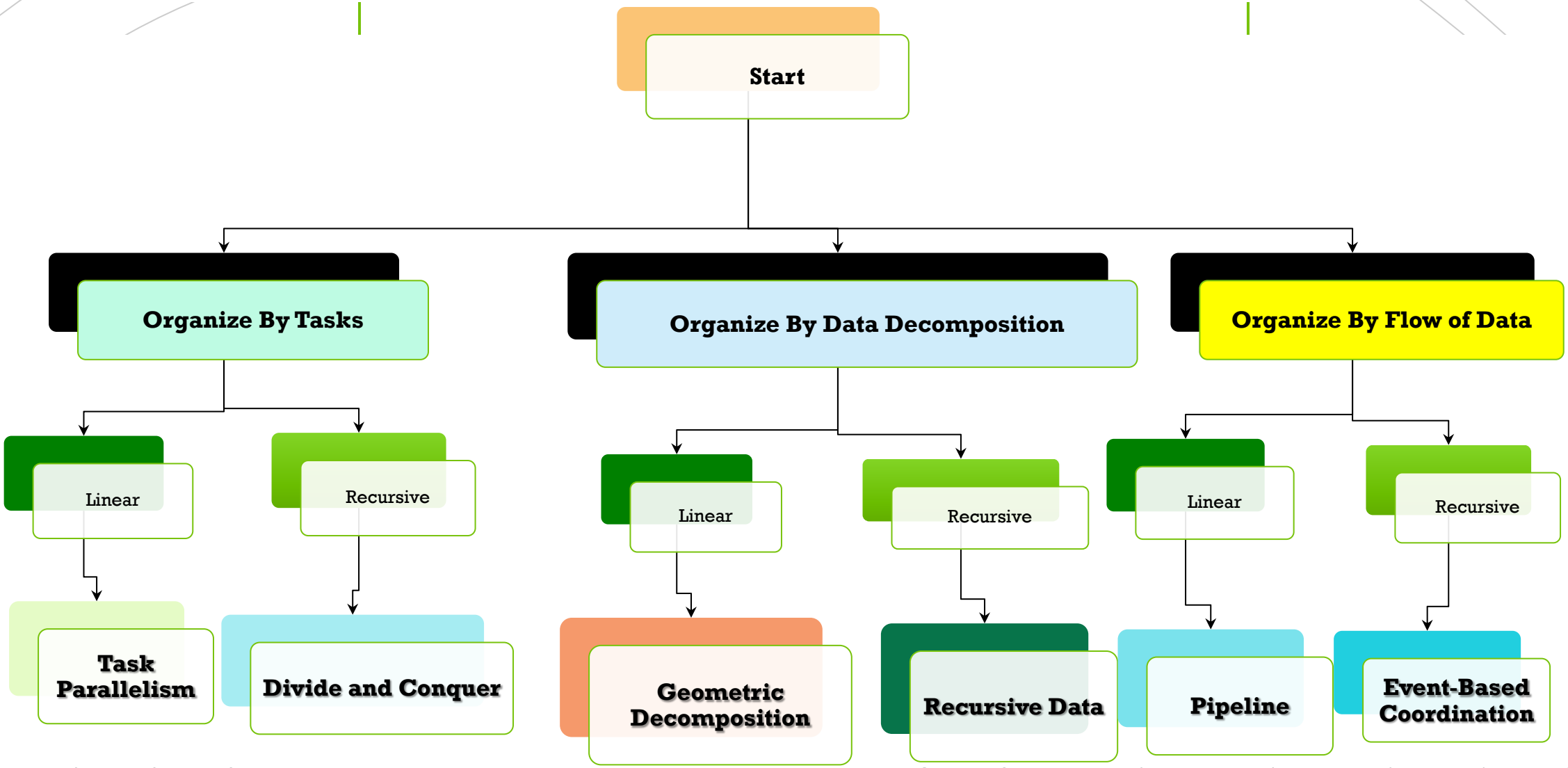


Organizing by Flow of Data

Pipeline

Event-Based Coordination

Algorithm Structure Decision Tree (Major Organizing Principle)



Universidad Industrial de Santander
CONSTRUIMOS FUTURO

Super Computación y Cálculo Científico UIS

What is SC3UIS?

R+D+i National Strategic Areas

.com and .org

.gov and .co

International R+D+i

.edu

Application Deployment

Scientific Software Development

SCI- IT Management and Support

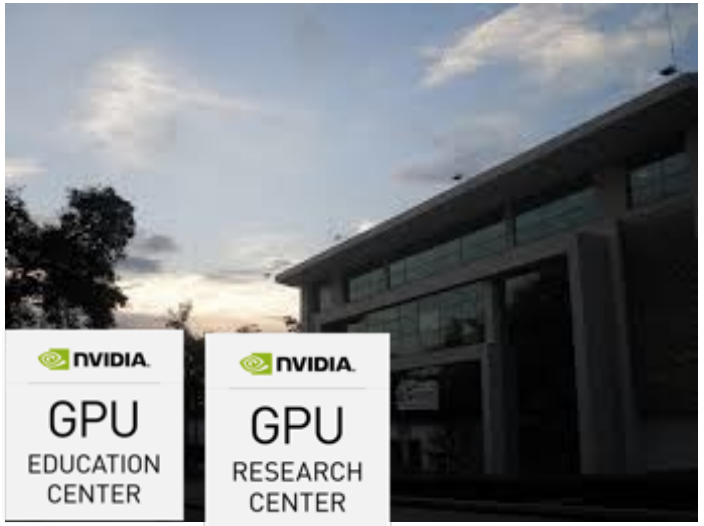
Strategic Mediation and Training

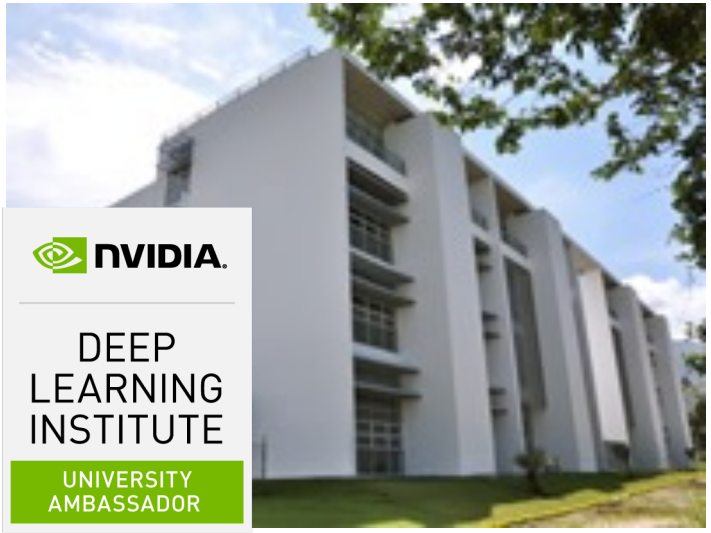
Research and Innovation

NVIDIA DEEP LEARNING INSTITUTE UNIVERSITY AMBASSADOR

CAGE

SC3UIS at UIS (@UIS) and Guatiguara Technology Park (@PTGuatiguara)





- Founded in 2012 After the Advanced Technology Project for Sciences and Engineering
- 2 Main Sites (CENTIC and PTG EDI)
- Next Step: SC3UIS becomes Colombia Advanced Computing Center
- In bioinformatics, we are experience (laureate) for genomics using Parallel Computing and Deep Learning.

Uso de CPU por Área de Investigación (horas utilizadas) 01/01/2017-31/12/2017


MÉXICO ISSSTE

EL INSTITUTO DE SEGURIDAD Y SERVICIOS SOCIALES DE LOS TRABAJADORES DEL ESTADO
OTORGA EL PRESENTE

RECONOCIMIENTO

A: JUAN ANTONIO GONZÁLEZ BARRIOS, MARÍA DEL ROCÍO THOMPSON BONILLA, FRANCISCO JOSÉ MARTÍNEZ PÑÉREZ, CARLOS JAIME BARRIOS HERNÁNDEZ LOLA XIOMARA BAUTISTA ROSO, REFUGIO RODRÍGUEZ VÁZQUEZ, DANIEL MARTÍNEZ FONG.

PRIMER LUGAR EN INVESTIGACIÓN BASICA

CON EL TRABAJO DE INVESTIGACIÓN:

Patente MX/a/2016/005610*
"OLIGONUCLEÓTIDOS Y PROCESO PARA DETECTAR EL VIRUS DE LA INFLUENZA A H1N1"

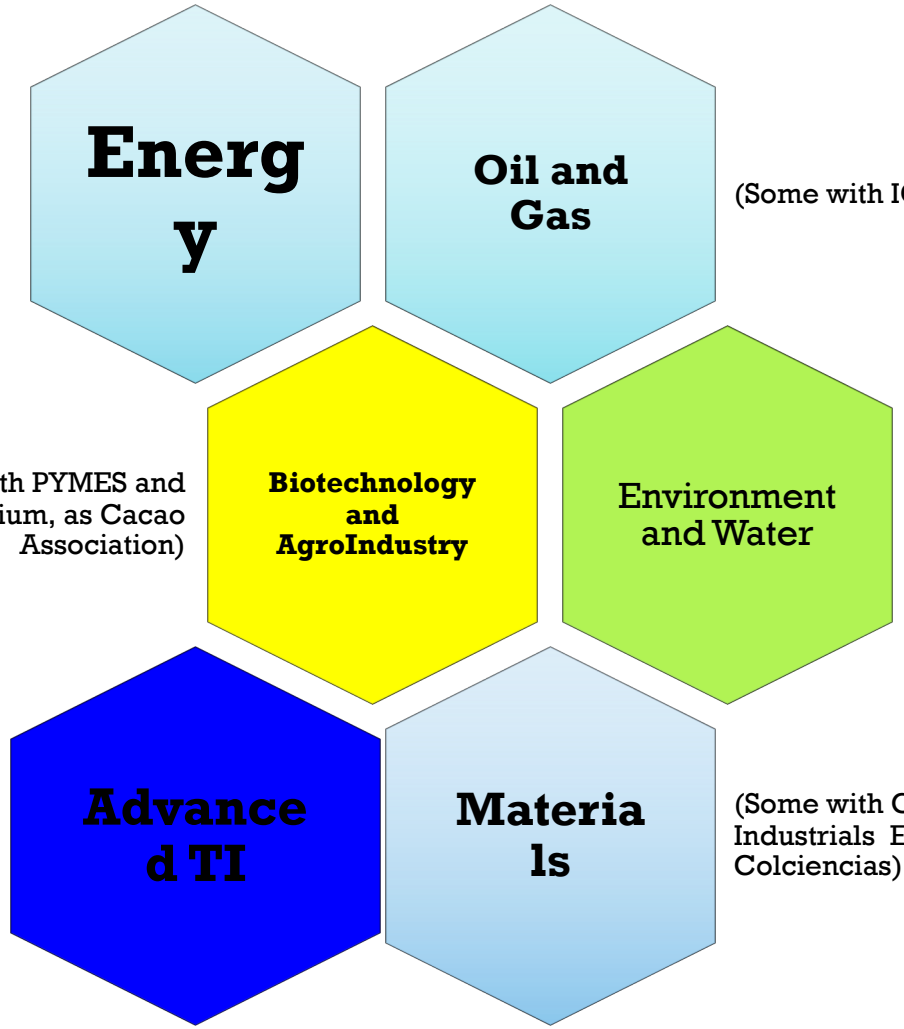
"10° ENCUENTRO NACIONAL DE INVESTIGACIÓN Y PREMIO NACIONAL DE INVESTIGACIÓN ISSSTE 2017"
CIUDAD DE MÉXICO, OCTUBRE 26 Y 27 DE 2017

DR. MIGUEL ÁNGEL SERRANO BERRONES
JEFE DE SERVICIOS DE ENSEÑANZA E INVESTIGACIÓN

DR. SAMUEL FUENTES REYNA
SUBDIRECTOR DE REGULACIÓN Y ATENCIÓN HOSPITALARIA

DR. MACRO JIMÉNEZ PONCE
JEFE DEL DEPARTAMENTO DE INVESTIGACIÓN

R+D+i Axes (@PTGuatiguara)



2017 Important Numbers

4 Patents
5 Spin Off in Incubation Process (Potentially for 2018 more than 10)
3 Big International Collaborations (more than 5M USD)

2018 New Axes:
Healthcare
New Generation of Automotive Motors
Human and Social Development

Parnertship

Some Conclusions

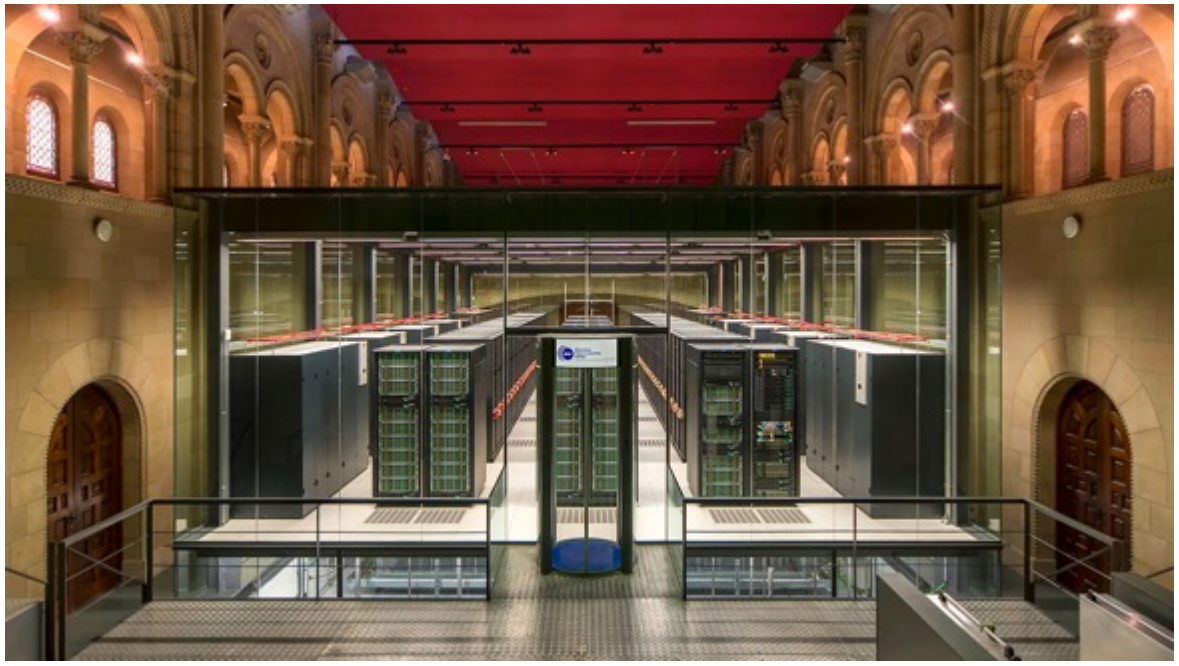
- High Performance Computing allows science and mathematics dreams and implementations… i.e. Artificial Intelligence Implementation, data analytics, blockchain and more…
- Computer systems involve different technologies and hybrid architectures, demanding sustainability, dynamicity and they need support changes in the scale of data and processing…. And all processing in parallel.
- Power consumption, energy aware and computational efficiency reach sustainability. It is proposed from the design of the architecture and it must be dynamic.
- Big and little (embedded) HPC Architectures with the same challenges (memory contention, stable speed – up, parallel coherence) follows same kind of solutions, but with different scale of treatment observing the data level.
- HPC is expensive (but It is more expensive to not have HPC Knowledge and Resources)
- Parallel Computing is not a tendency. (From 2015 is mandatory in all universities and colleges in USA parallel computing, scientific computing and advanced computing courses in science and engineering programs (programming computing is mandatory also in high school from 2009).

Class work (in groups)

- Revision of Chapter 2 of Designing and Building Parallel Programs, by Ian Foster in http://www.mcs.anl.gov/~itf/dbpp/
- Solve in the Exercises Section the 1 and 2 numerals.
- Imagine a solution for your final Project (conceptually)

Recommended Lectures

- **The Art of Concurrency "A thread Monkey's Guide to Writing Parallel Applications"**, by *Clay Breshears* (Ed. O Reilly, 2009)
- **Writing Concurrent Systems. Part 1.**, by *David Chisnall* (InformIT Author's Blog: http://www.informit.com/articles/article.aspx?p=1626979)
- **Patterns for Parallel Programming**., by T. Mattson., B. Sanders and B. MassinGill (Ed. Addison Weslley, 2009) Web Site: http://www.cise.ufl.edu/research/ParallelPatterns/
- Designing and Building Parallel Programs, by Ian Foster in http://www.mcs.anl.gov/~itf/dbpp/



From: www.bsc.es

@carlosjaimebh