An Introduction to HPC and Advanced Computing

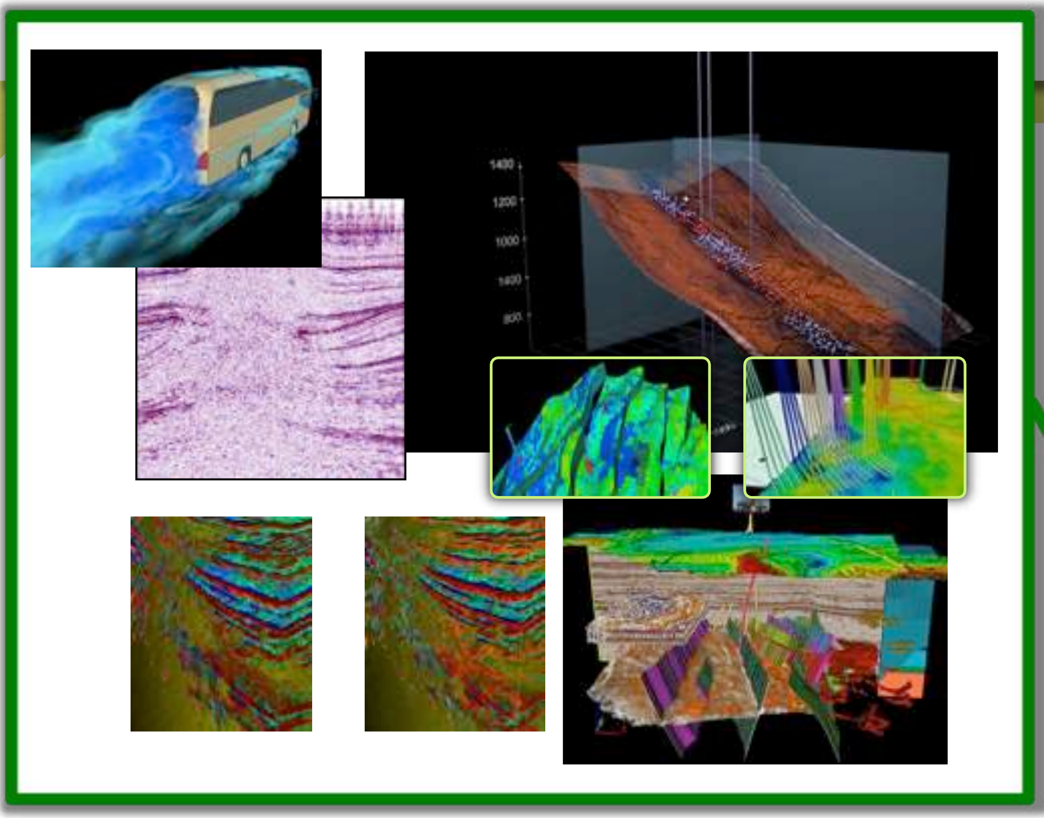
In 105 Slides – Part 1

Carlos Jaime Barrios Hernández, PhD

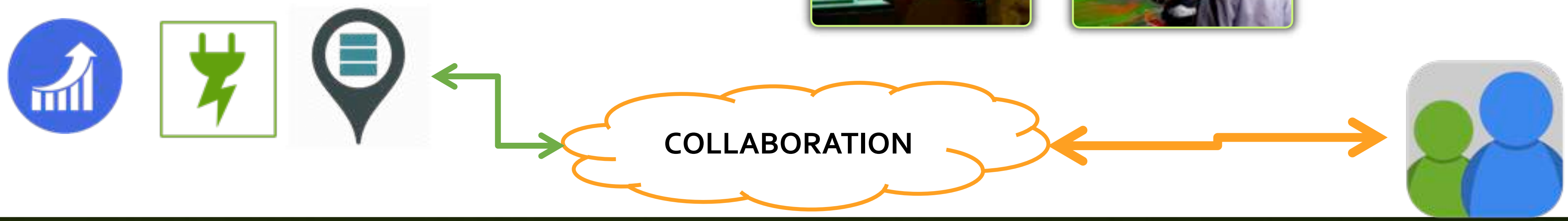
@carlosjaimebh

The (Big) Questions: What and How?

Why?

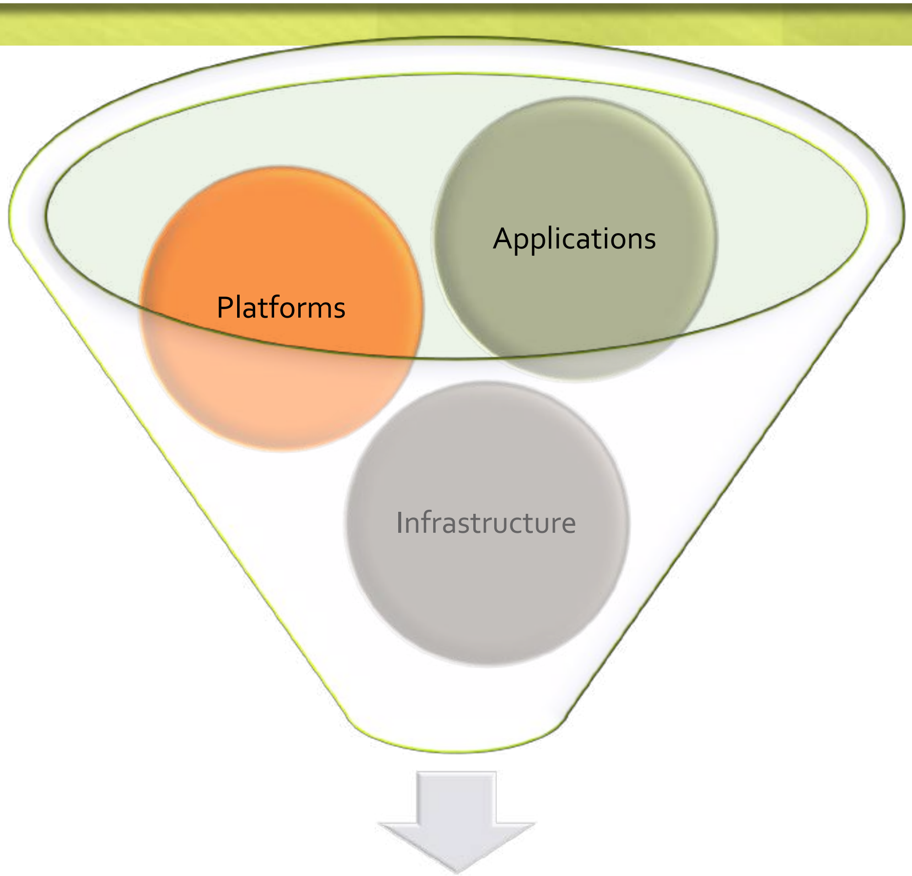


- Large Data Sets
- Complex Mathematics
- Complex Models
- Real Time
- Interaction and Confrontation
- Large Scale Visualization
- High Resolution
- High Performance and Capacity
  - VR Needs
  - Big Data and Deep Learning



COLLABORATION

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

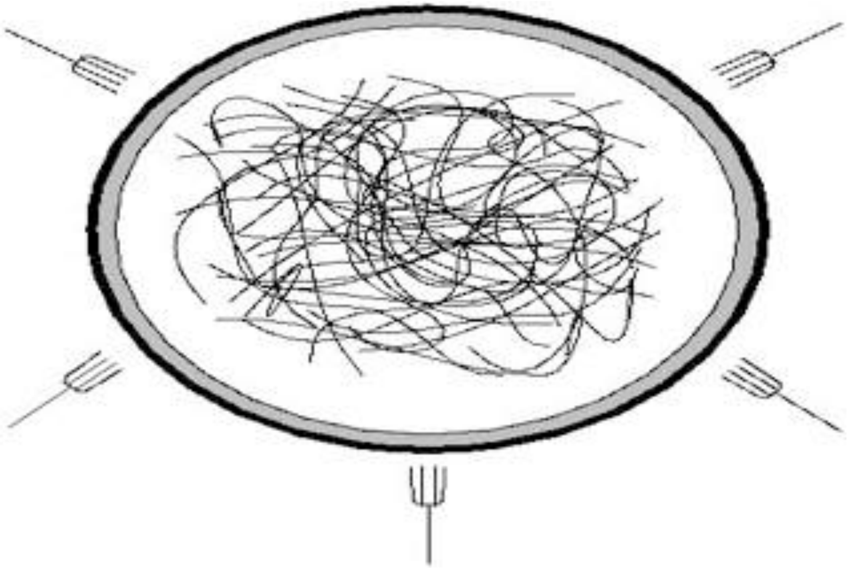
Platform

- □ 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.

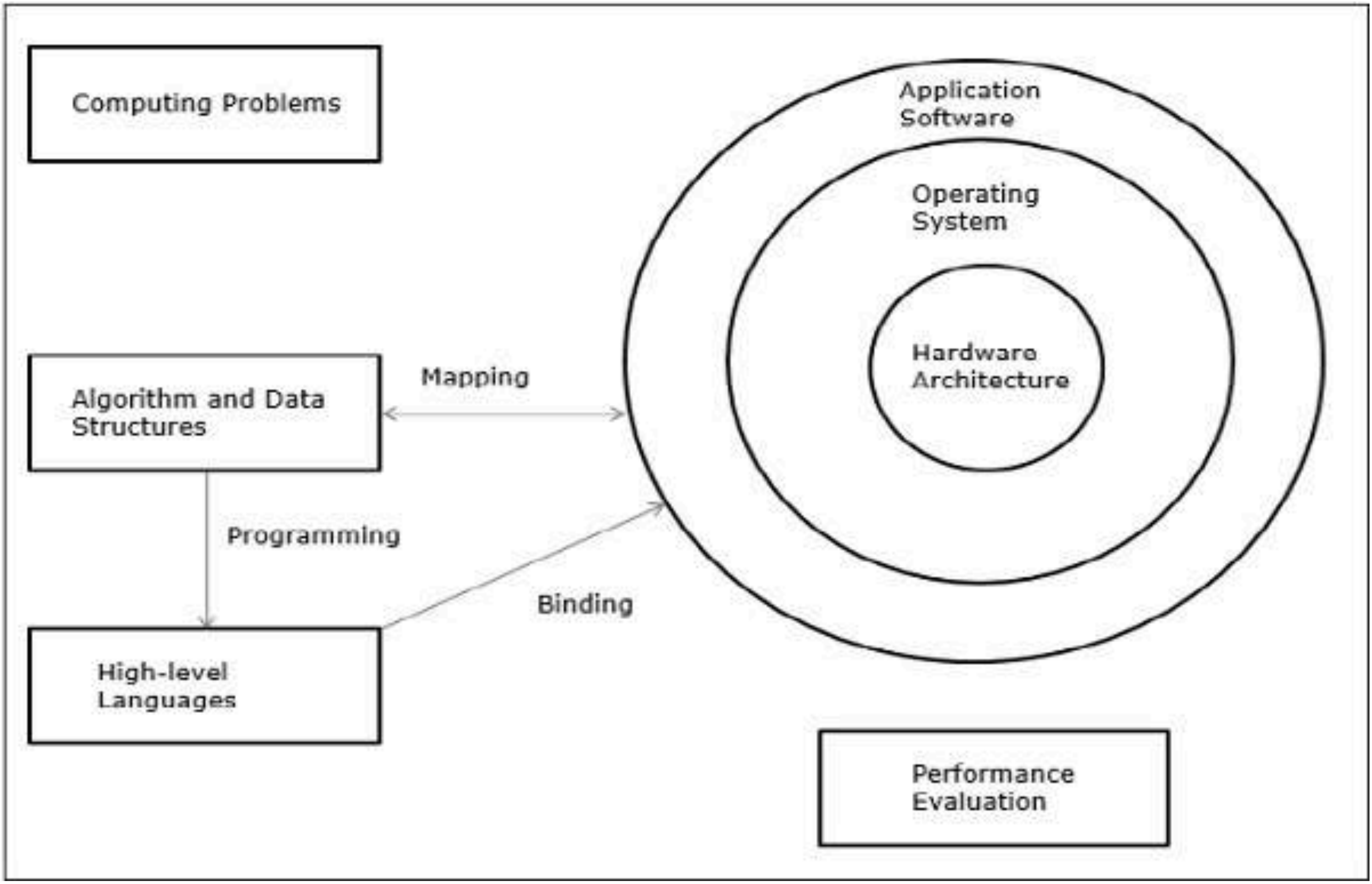
- **Implicit parallelism** is a characteristic of a programming language that allows a compiler or interpreter to automatically exploit the parallelism inherent to the computations expressed by some of the language's constructs.

- **Explicit parallelism** is the representation of concurrent computations by means of primitives in the form of special-purpose directives or function calls.

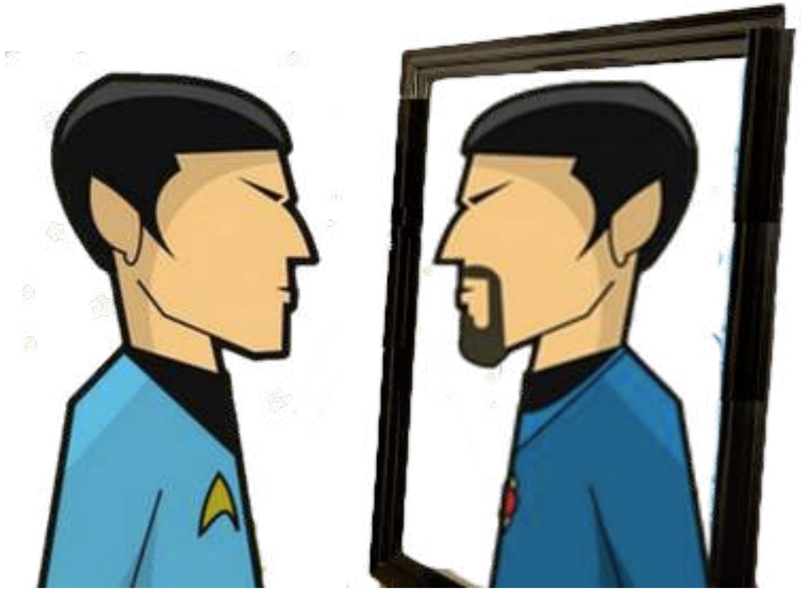
- We need two (mixed) approach in Architecture: Applications and Hardware (system).

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 Probler
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
   - Memory
   - Network and I/O
   - Storage

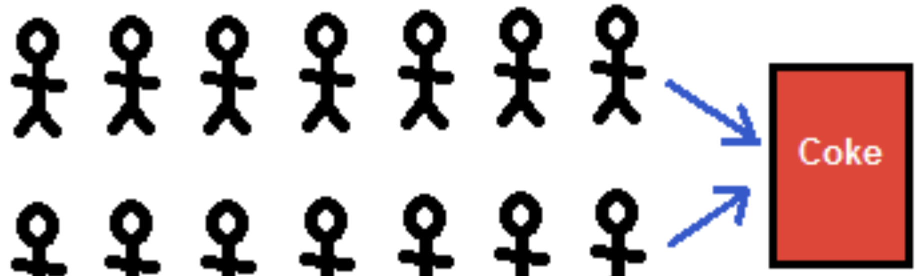


Pervasive and Thinking Parallelism

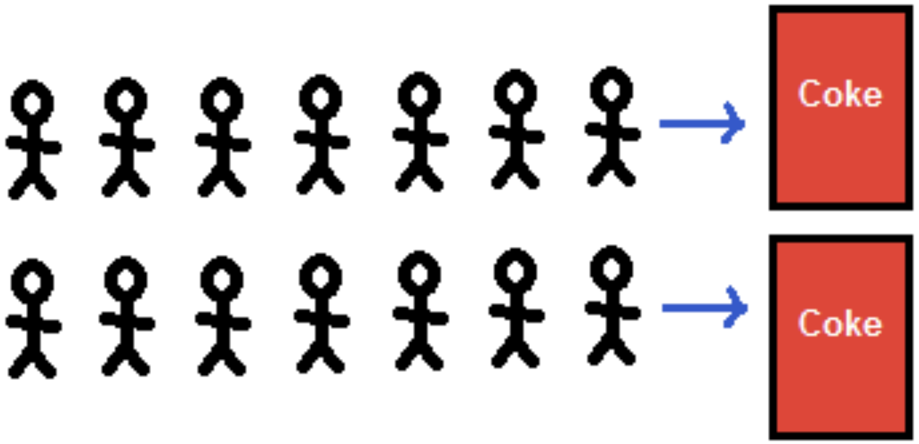


- It is not a question of « Parallel Universes » (Almost)
- Data Sources
- Processing and Treatment
- Resources (Available and Desire)
- Energy Consumption
- Natural "thinking" (Natural Compute?)

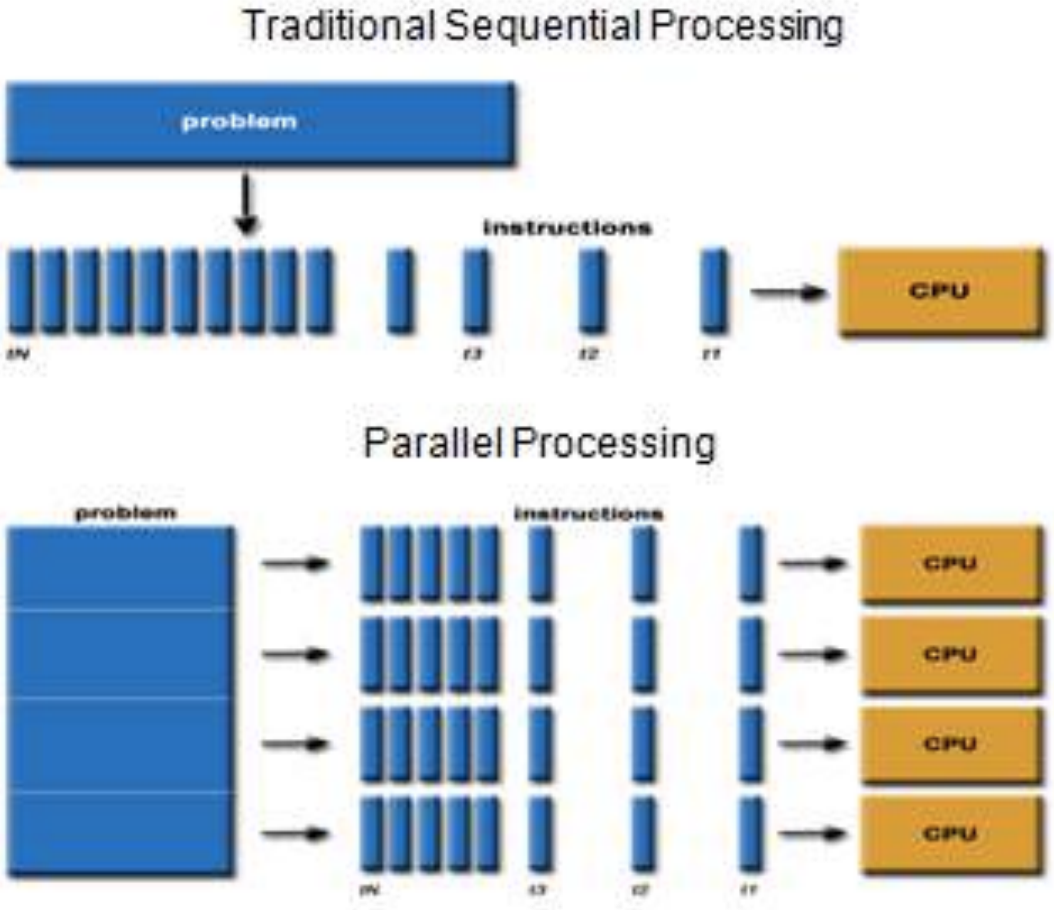
Thinking in Parallel (computing) – The Typical Visions



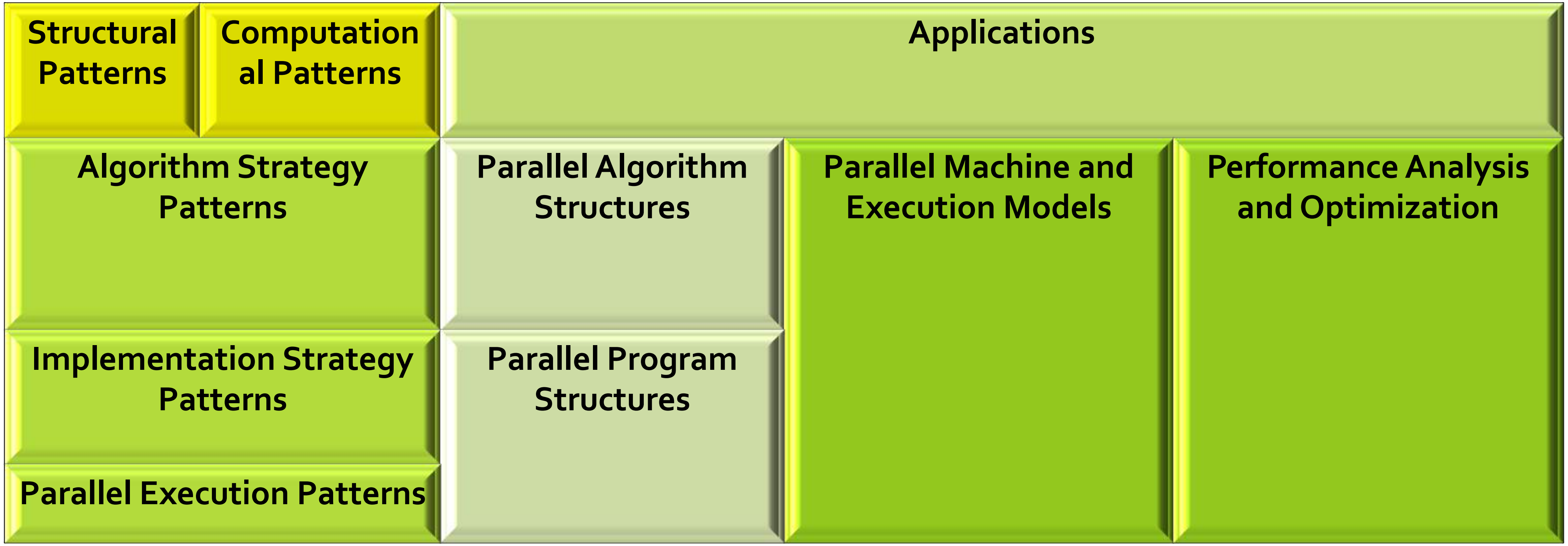
Concurrent: 2 queues, 1 vending machine



Parallel: 2 queues, 2 vending machines



Thinking in Parallel (computing) – an OPL hierarchy



Structural Patterns

Computational Patterns

Applications

Algorithm Strategy Patterns

Implementation Strategy Patterns

Parallel Execution Patterns

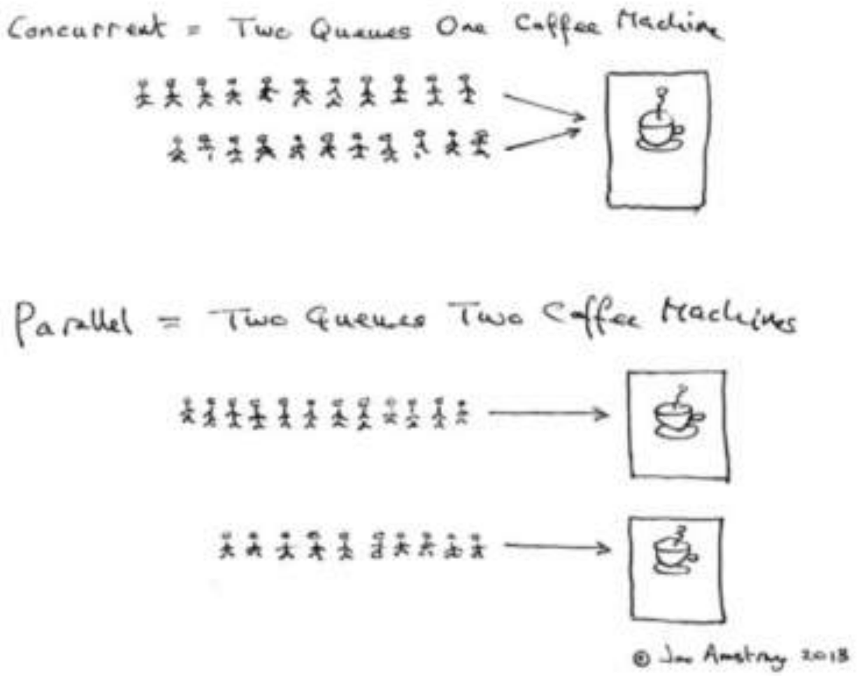
Parallel Algorithm Structures

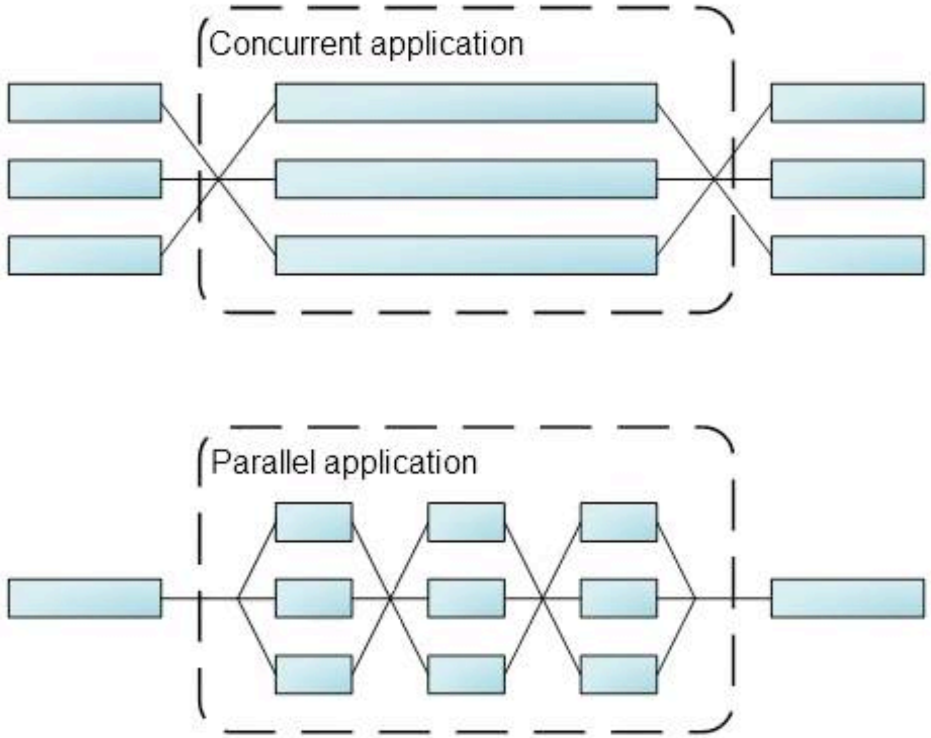
Parallel Program Structures

Parallel Machine and Execution Models

Performance Analysis and Optimization

CONCURRENCY | PARALLELISM

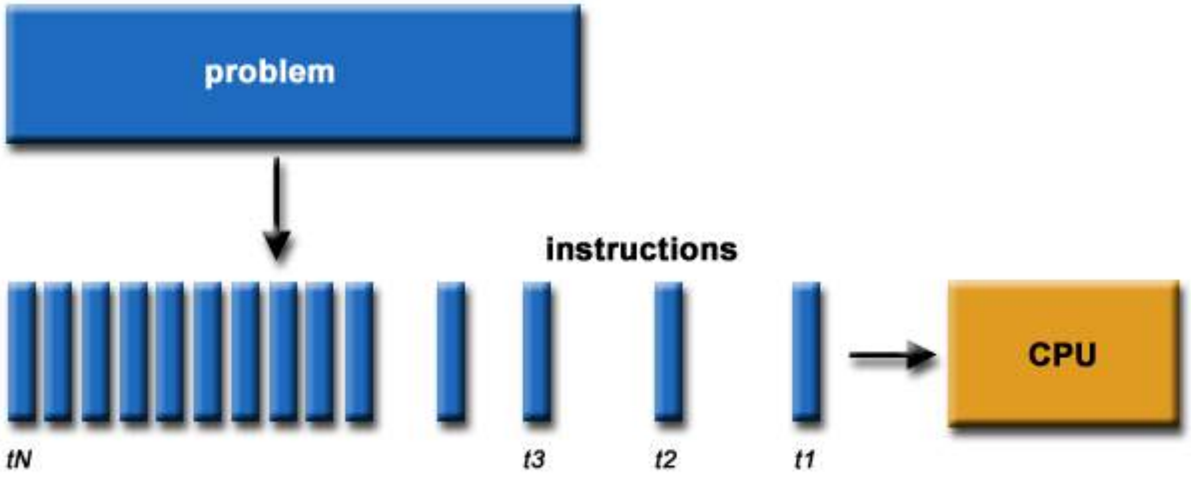

Concurrent = Two Queues One Coffee Machine
Parallel = Two Queues Two Coffee Machines
© Joe Armstrong 2013

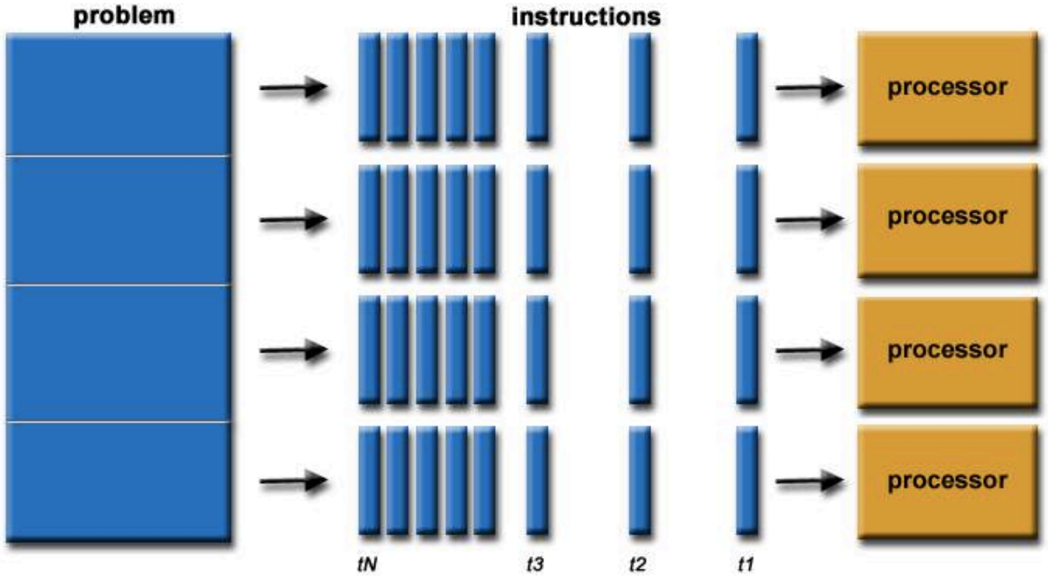


From J. Armstrong Notes: http://joearms.github.io/2013/04/05/concurrent-and-parallel-programming.html

Any Parallel System is concurrent: Simulatenous Processing, Parallel but limited ressources.

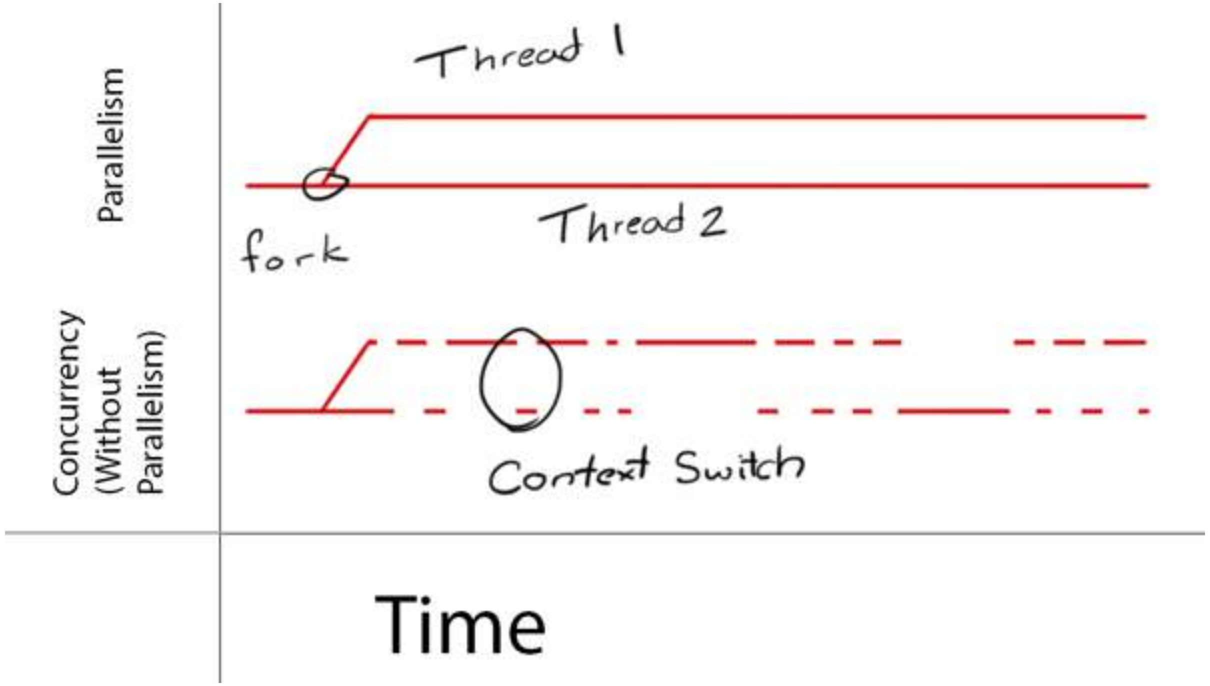
Serial vs Concurrent/Parallel Approach





Reduction in Execution Time (However, overhead problem)
Instructions to Multithreading (To exploit Parallelism)
Syncrhonization (with all derivated concerns...)

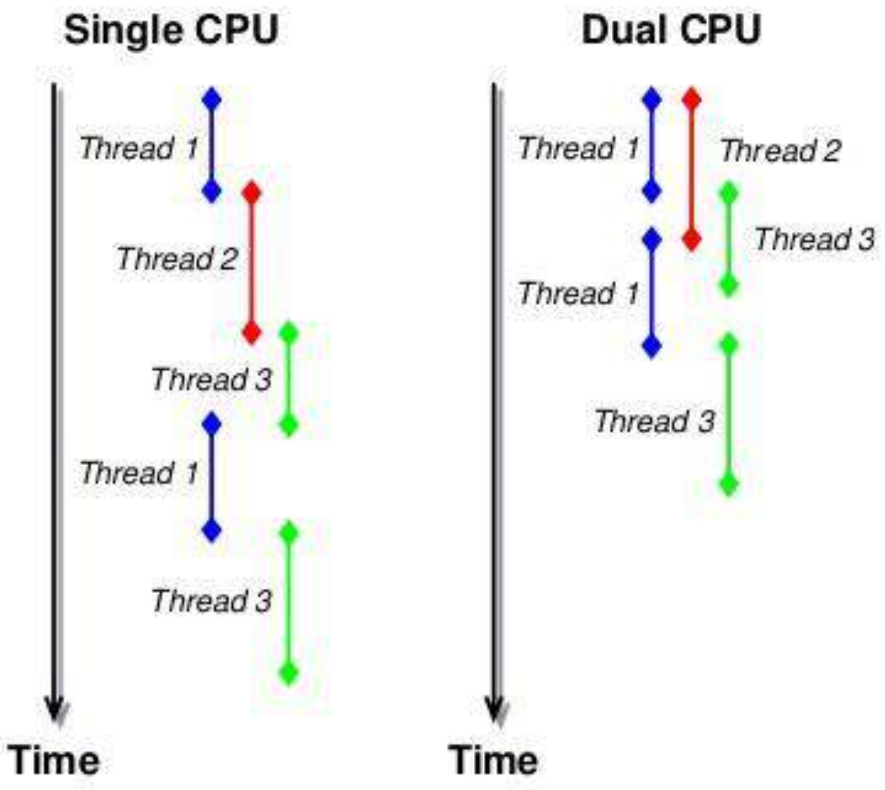
Concurrency vs Concurreny/Parallelism Behavior



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



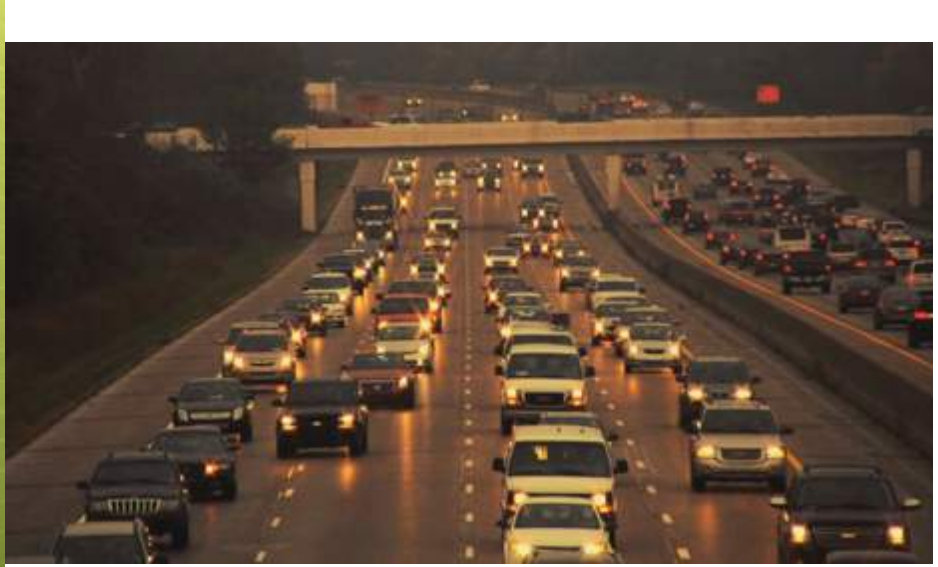
Single System
- Multiple Threads in Runtime
- Almost Synchronization Strategies
- Memory Allocation

Dual System
- Multiple Parallel 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

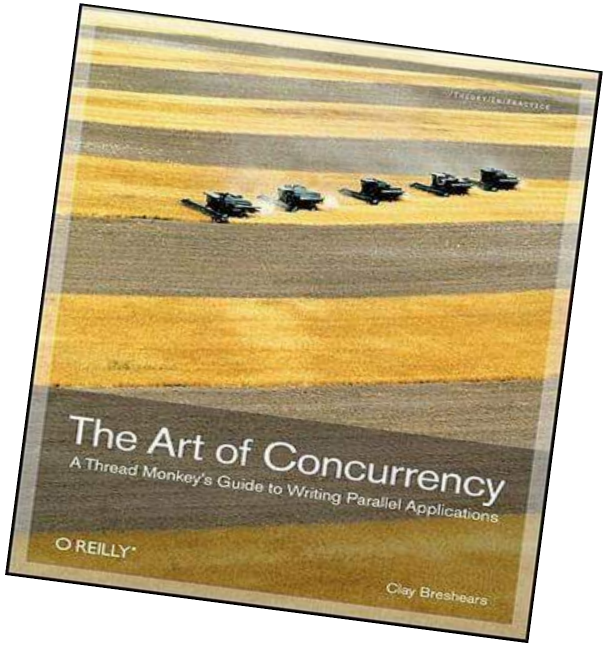
- Concurrent processes can reduce duplication in code.
- The overall runtime of the algorithm can be significantly reduced.
- More real-world problems can be solved than with sequential algorithms alone.
- Redundancy can make systems more reliable.

- 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

Parallel Computing

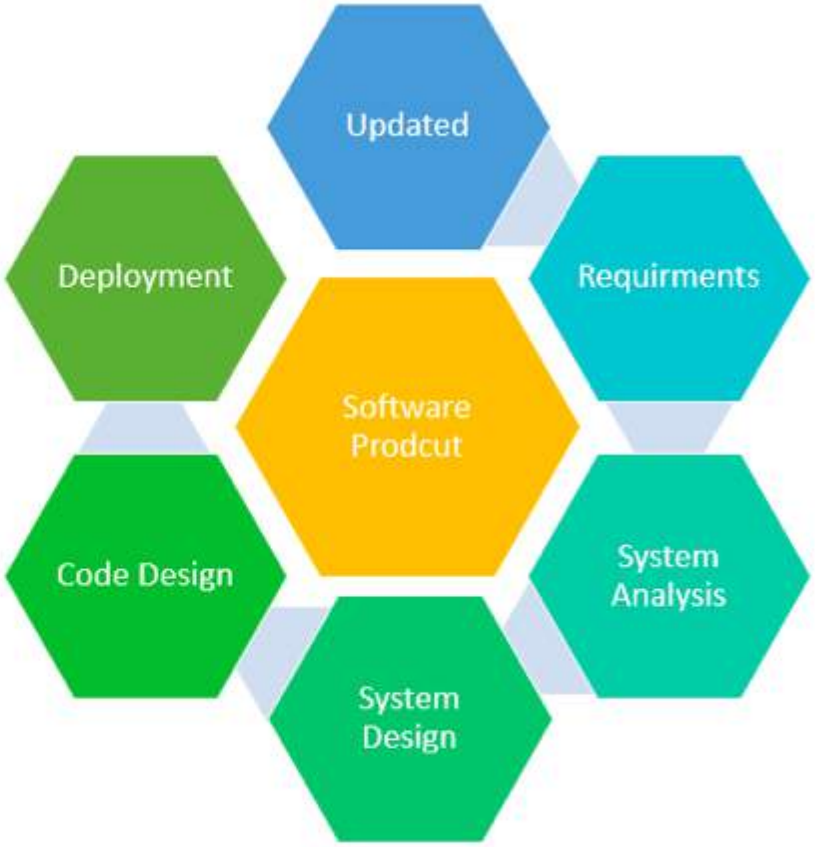
- Parallel Computing exploit Concurrency
  - In “system” terms, concurrency exists when a problem can be decomposed in sub problems that can safely executed at same time (in other words, concurrently)



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

How to Exploit (Better) Concurrency

- (Remember) Mixed Approach (Algorithms/Applications - Hardware/System.
- Good Techniques from Software Engineering
- Good Problem knowledge from scientific (domain) expertise
- Confrontation and Performance Evaluation



Questions?

EL MARE NOSTRUM ES EL CUARTO SUPERORDENADOR DEL MUNDO
DESPLIEGA 4.586 PROCESADORES Y PUEDE RESOLVER CUARENTA BILLONES DE OPERACIONES POR SEGUNDO
¿ES CAPAZ DE REDACTAR UN ESTATUT EN DOS DÍAS?

From: www.bsc.es

@carlosjaimebh