An Introduction to Parallelism, Concurrency and Acceleration (1)

CERN Academic Training – Jan 2016

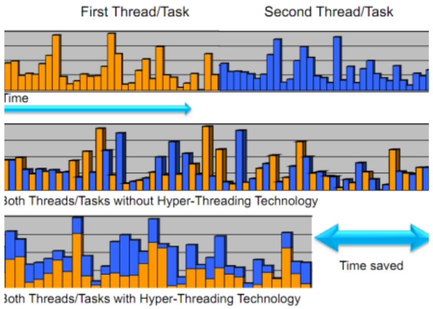
Andrzej Nowak

http://tik.services

tik.

technology
innovation
knowledge

Outline



Day 1:

Concurrency and Parallelism

Day 2:

Acceleration

A blessing and a curse

1950s

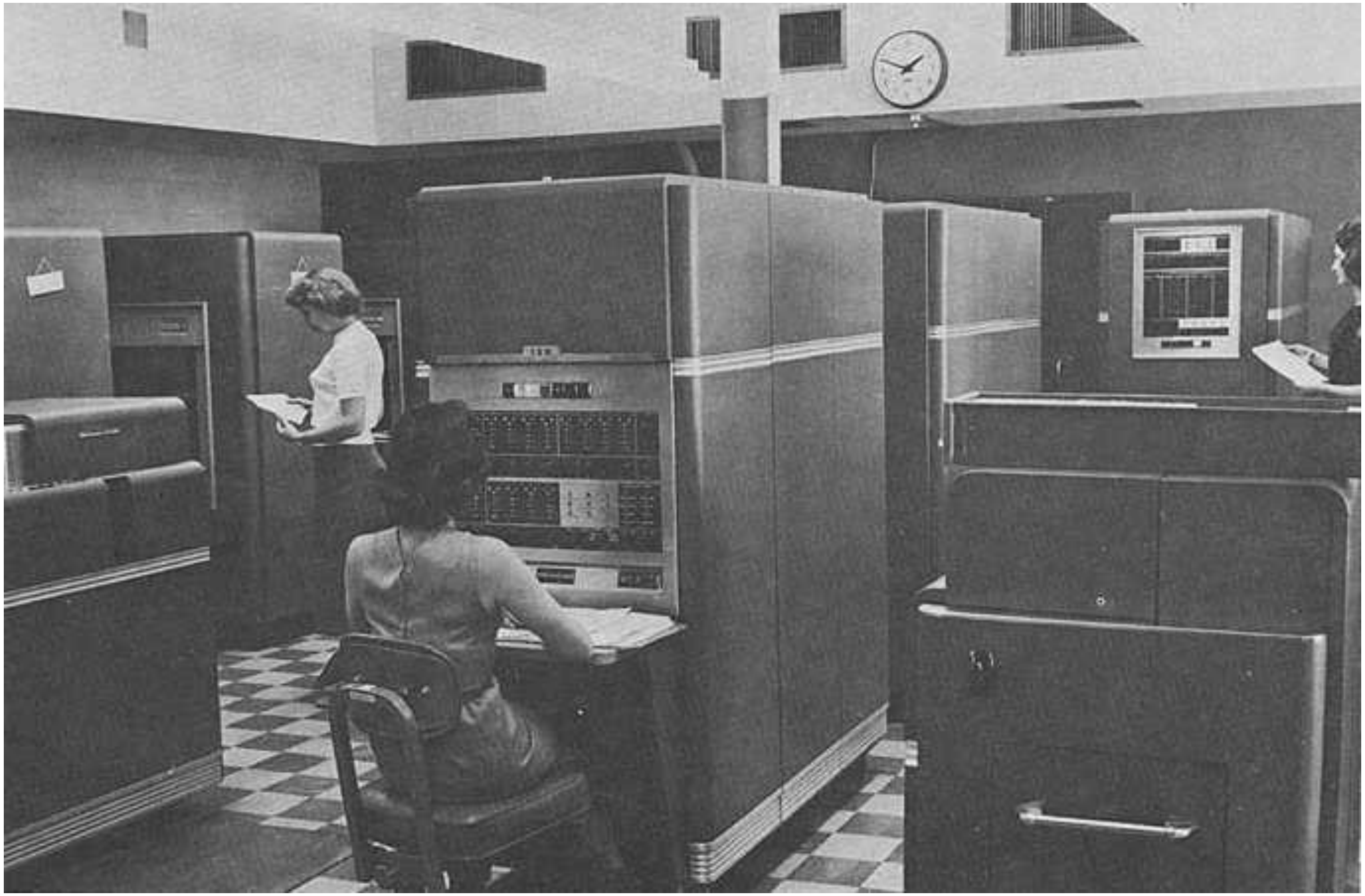


Image: IBM

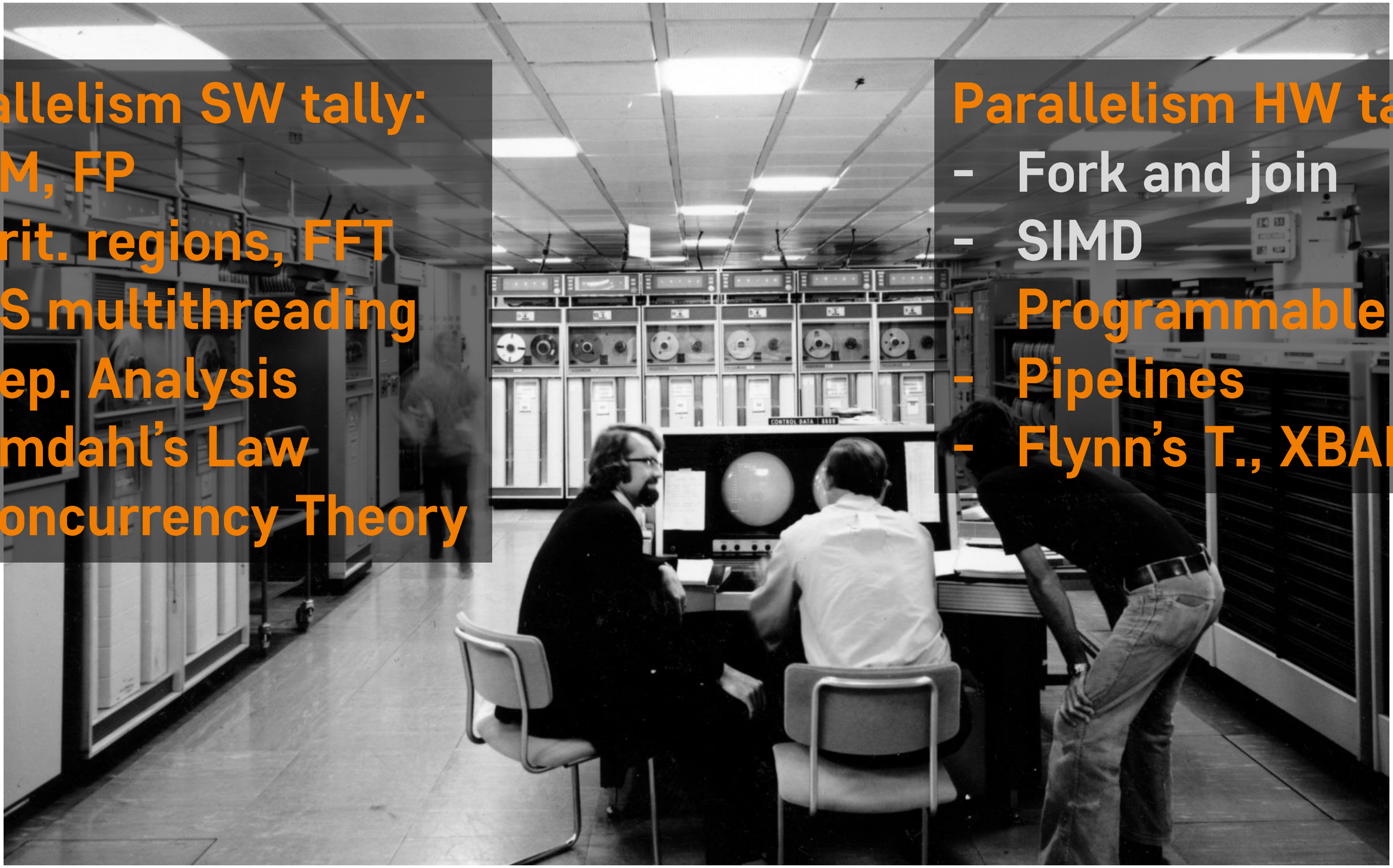
1950s

Parallelism tally:

- Fork and join
- SIMD

Image: LLNL

1960s

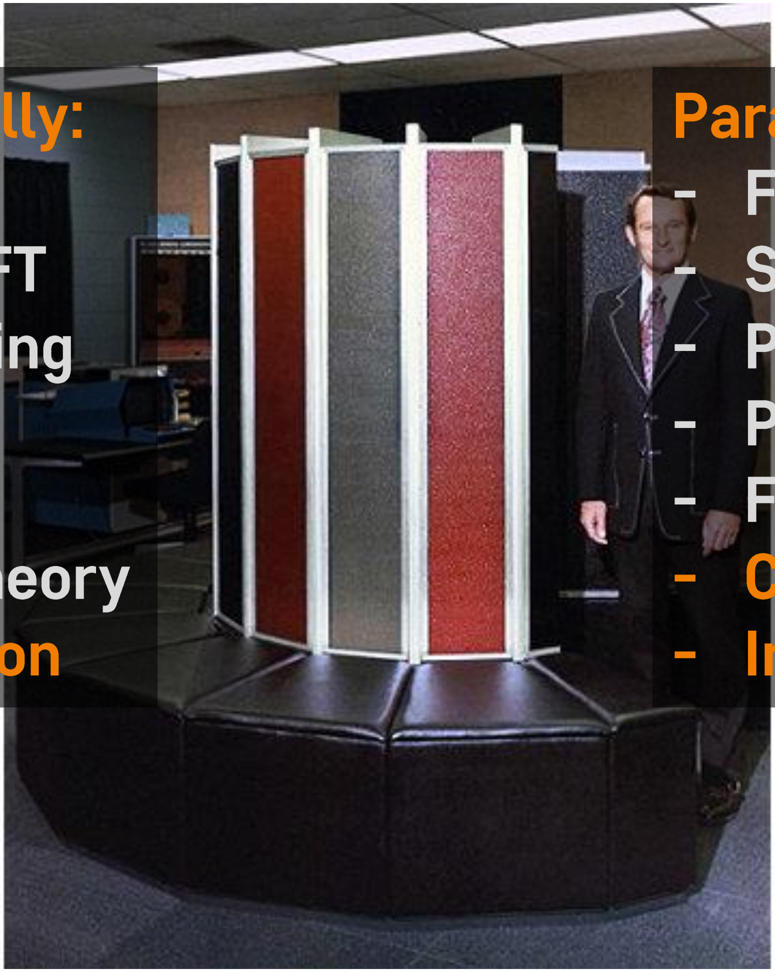


Parallelism SW tally:
- VM, FP
- Crit. regions, FFT
- OS multithreading
- Dep. Analysis
- Amdahl's Law
- Concurrency Theory

Parallelism HW tally:
- Fork and join
- SIMD
- Programmable IC
- Pipelines
- Flynn's T., XBARS

Image: CERN

1970s



Parallelism SW tally:

- VM, FP
- Crit. regions, FFT
- OS multithreading
- Dep. Analysis
- Amdahl's Law
- Concurrency Theory
- Autovectorization

Parallelism HW tally:

- Fork and join
- SIMD
- Programmable IC
- Pipelines
- Flynn's T., XBARS
- Co-processors
- Intel 4004

Image: Cray Research

1980s

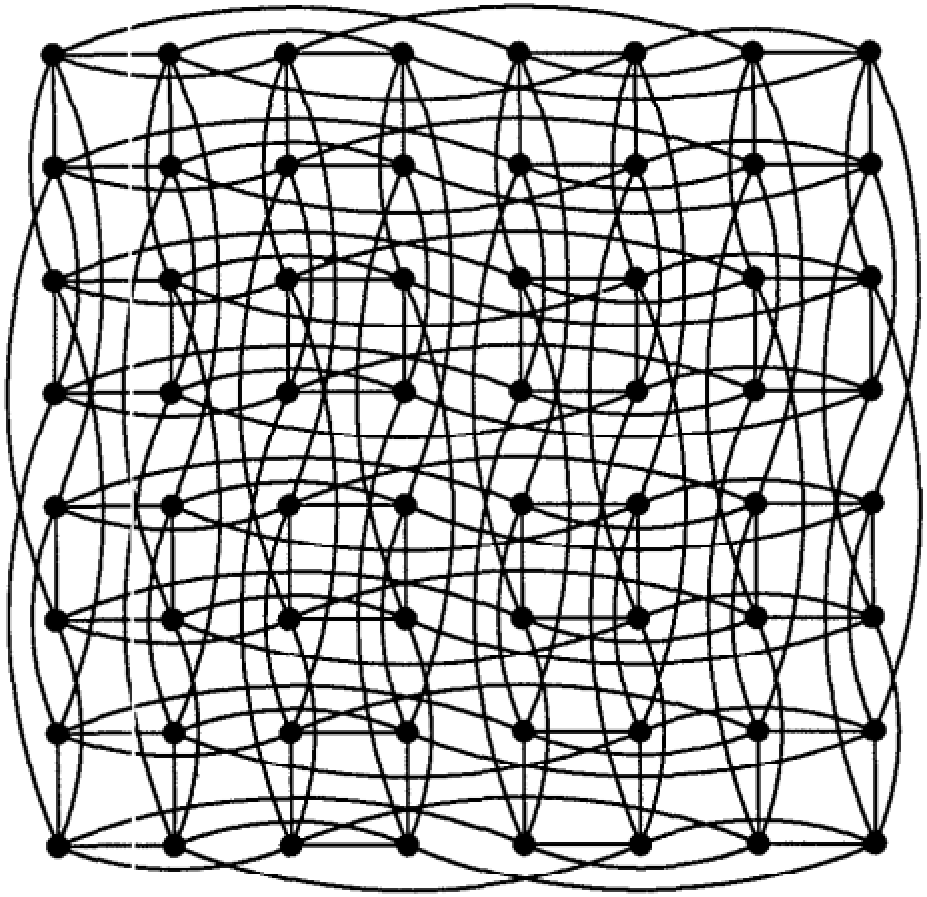
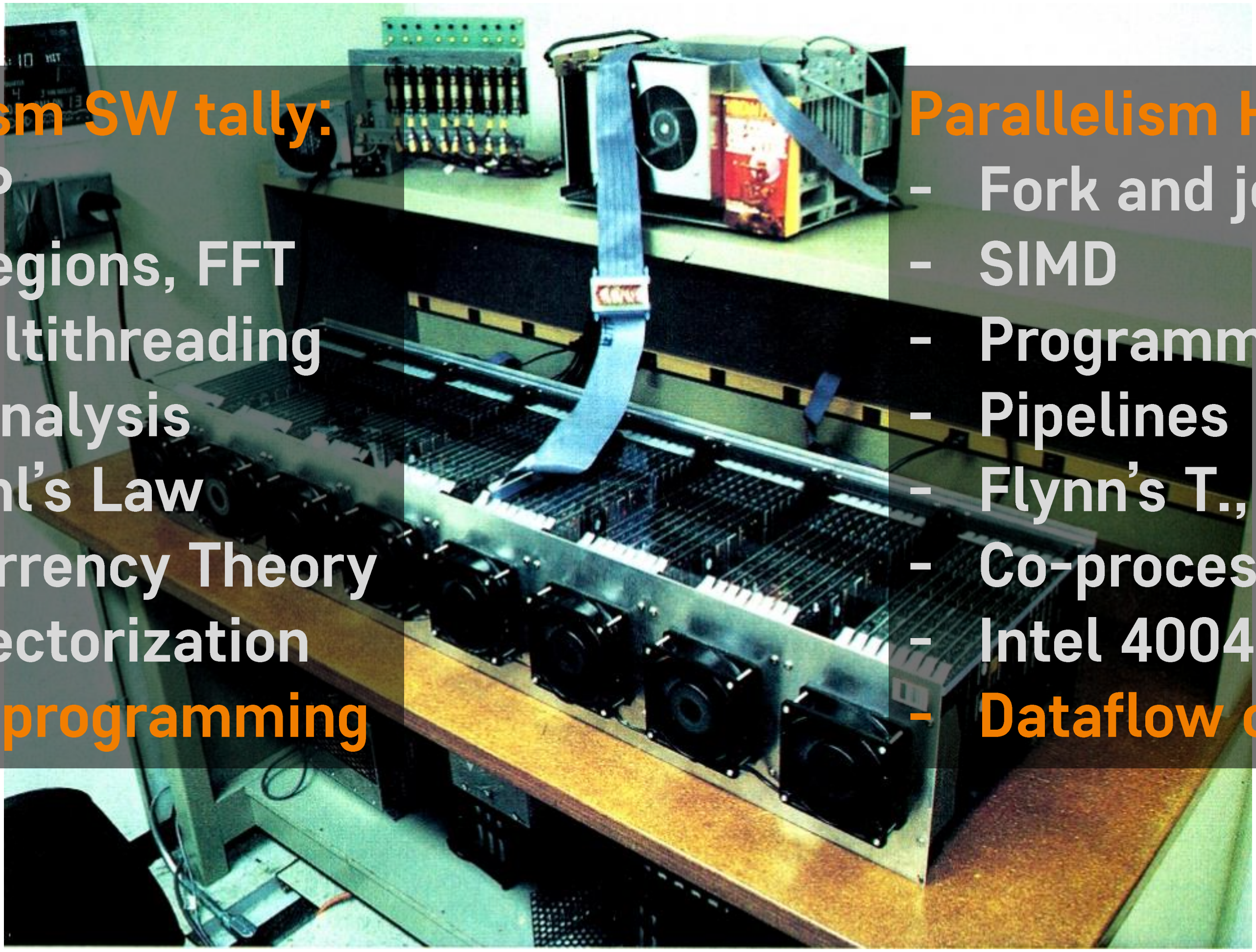


Image: C. L. Seitz, ACM

1980s



Parallelism SW tally:

- VM, FP
- Crit. regions, FFT
- OS multithreading
- Dep. Analysis
- Amdahl's Law
- Concurrency Theory
- Autovectorization
- SPMD programming

Parallelism HW tally:

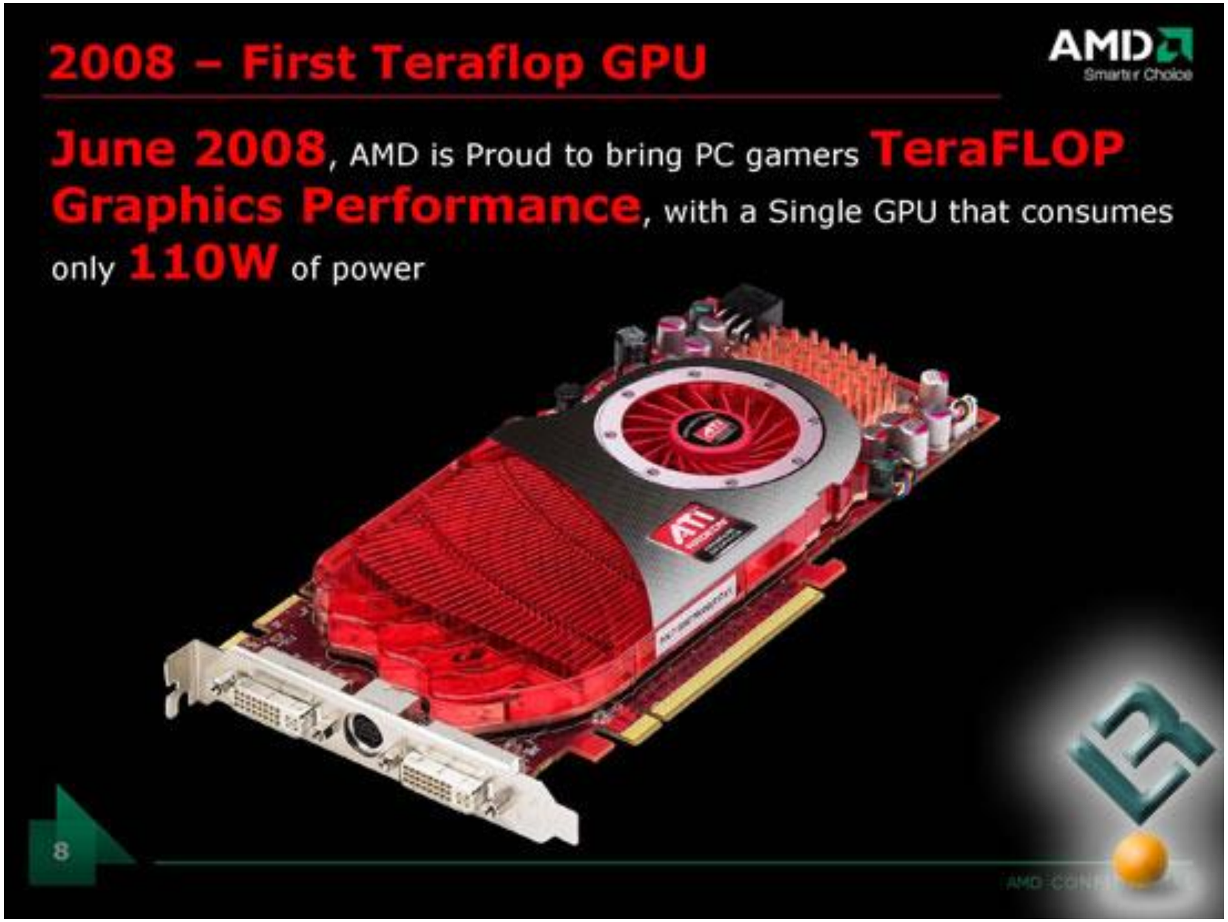
- Fork and join
- SIMD
- Programmable IC
- Pipelines
- Flynn's T., XBARS
- Co-processors
- Intel 4004
- Dataflow computer

Image: C. L. Seitz, ACM

1990s

Image: Top500

2000+



2008 – First Teraflop GPU

June 2008, AMD is Proud to bring PC gamers TeraFLOP Graphics Performance, with a Single GPU that consumes only 110W of power

Image: AMD/LR

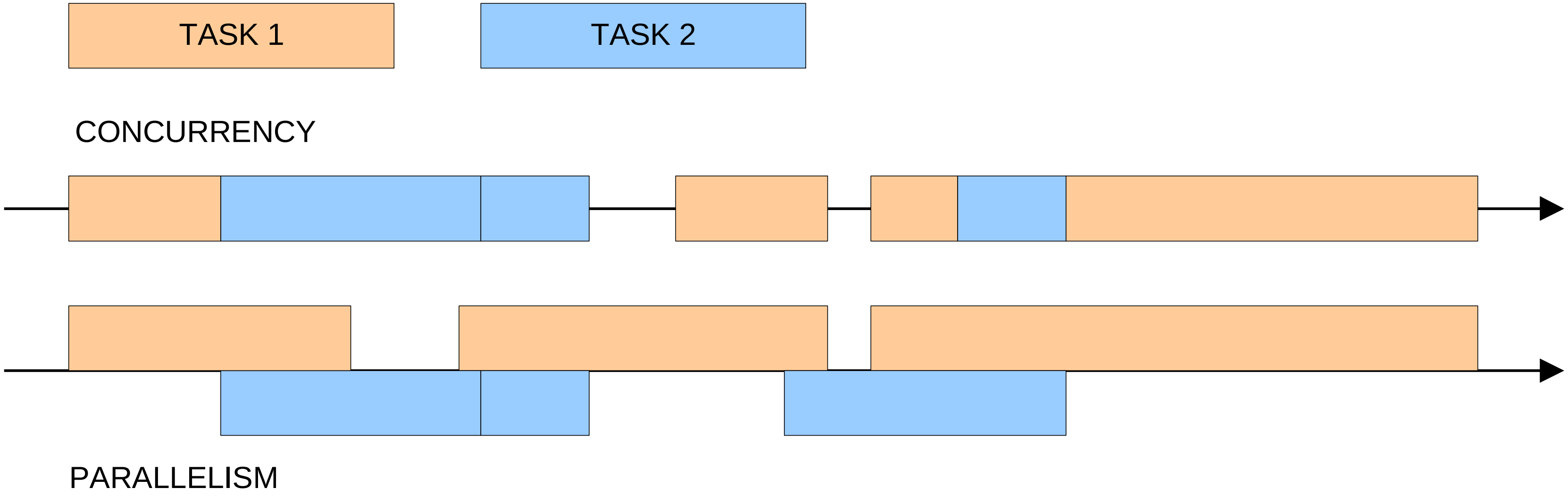
Base theory

Scalability

“Readiness for enlargement”

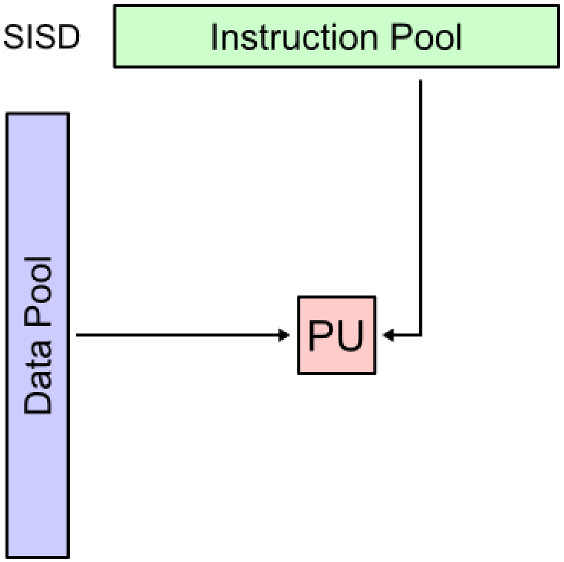
Concurrency and Parallelism

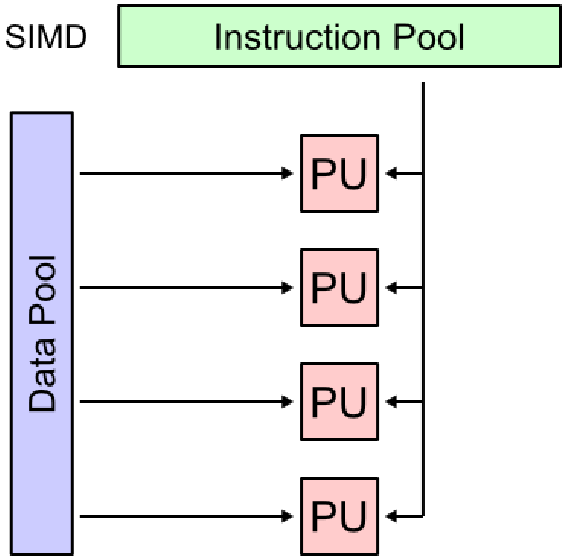
- Concurrency – interleaved execution
- Parallelism – parallel execution and concurrency



Flynn’s taxonomy (1)

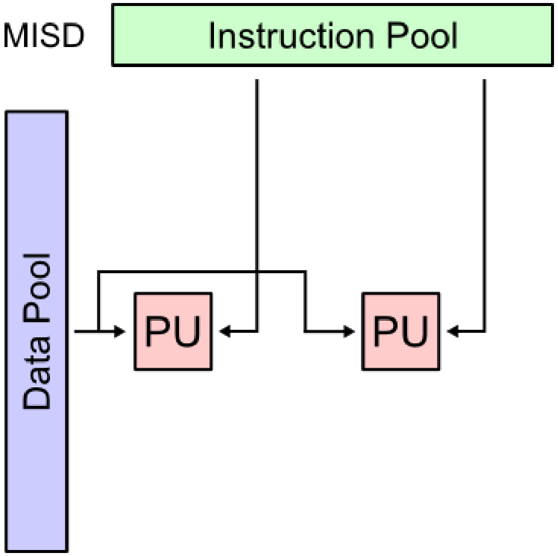
- SISD – Single Instruction, Single Data
  - Classical Von Neumann’s model
- SIMD – Single Instruction, Multiple Data
  - A GPU

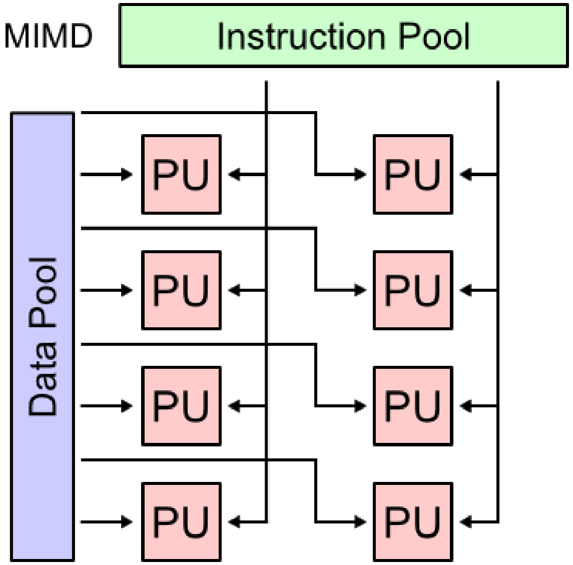




Flynn's taxonomy (2)

- MISD – Multiple Instruction, Single Data
  - Redundant systems, pipeline systems (disputable)
- MIMD – Multiple Instruction, Multiple Data
  - Distributed systems





Zooming in on a modern CPU

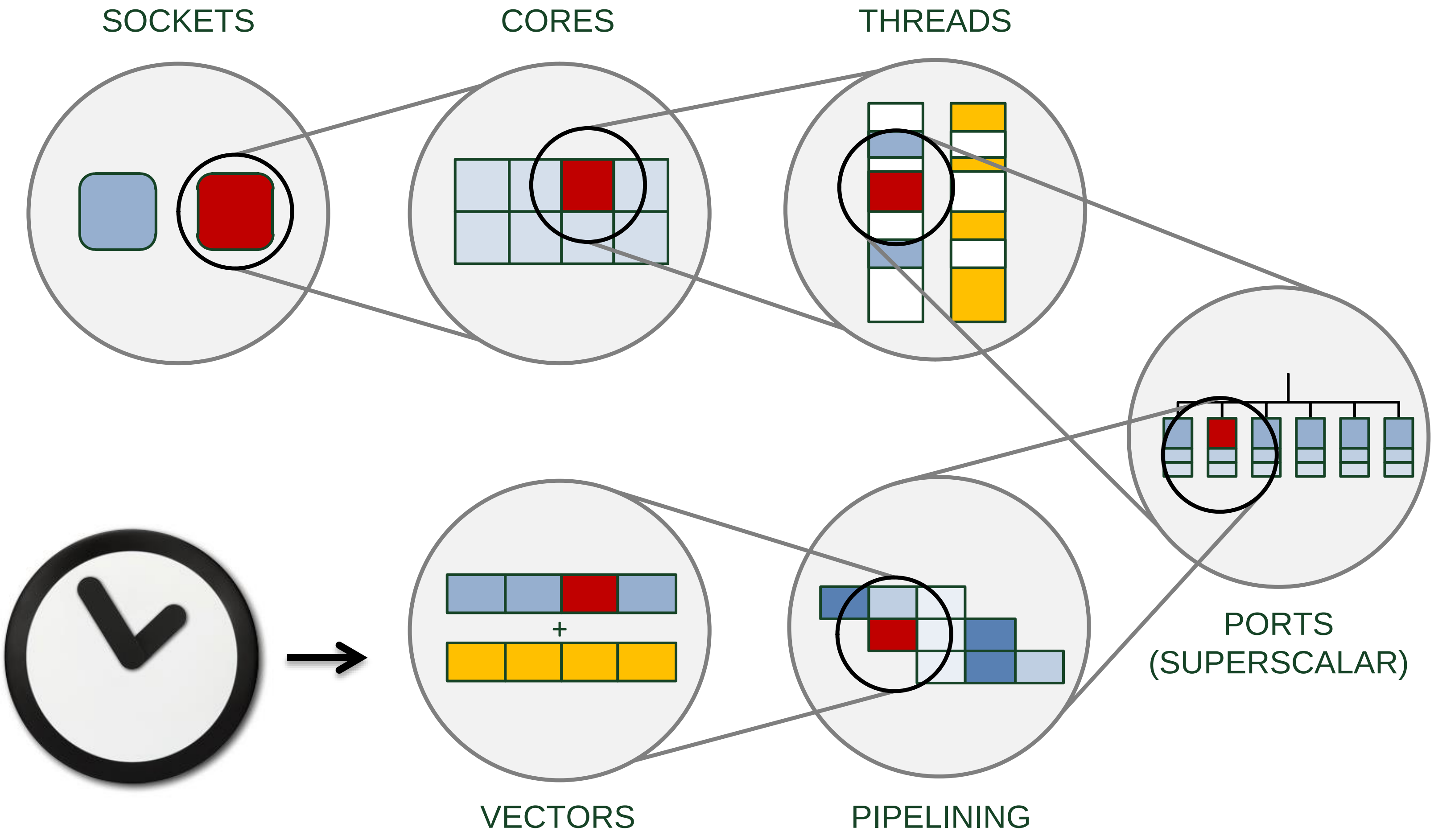


Image: A. Nowak / TIK

Amdahl's Law

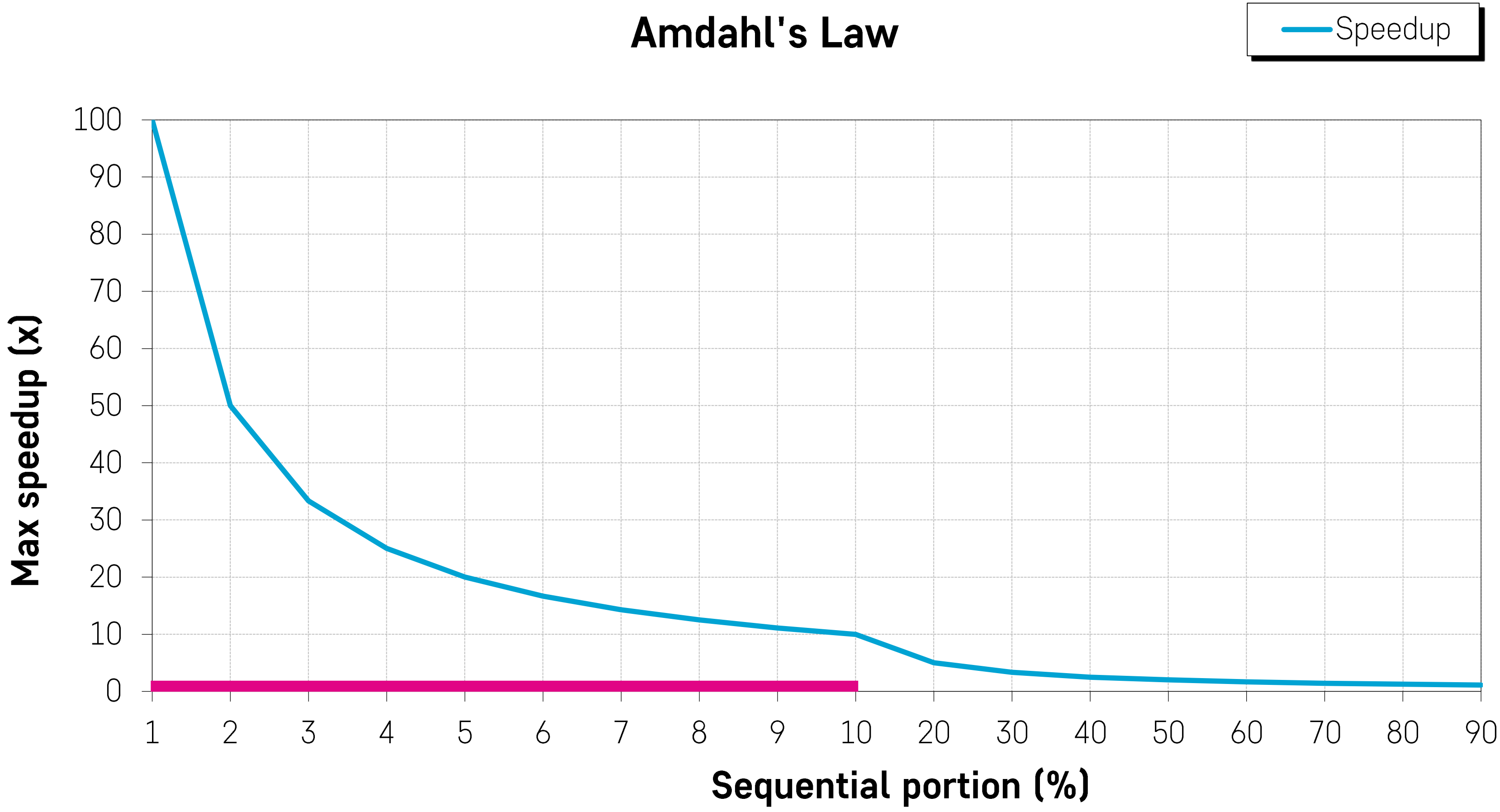
The Law of Strong Scaling

- Parallelized portion vs. the expected speedup
  - P – parallelized %
  - S – the speedup of that part

$$Speedup = \frac{1}{(1-P) + \frac{P}{S}}$$

Amdahl’s Law

The Law of Strong Scaling



Gustafson’s Law

The Law of Weak Scaling

- Observation: we can often grow the parallel portion

$$Speedup = a(n) + N(1 - a(n))$$

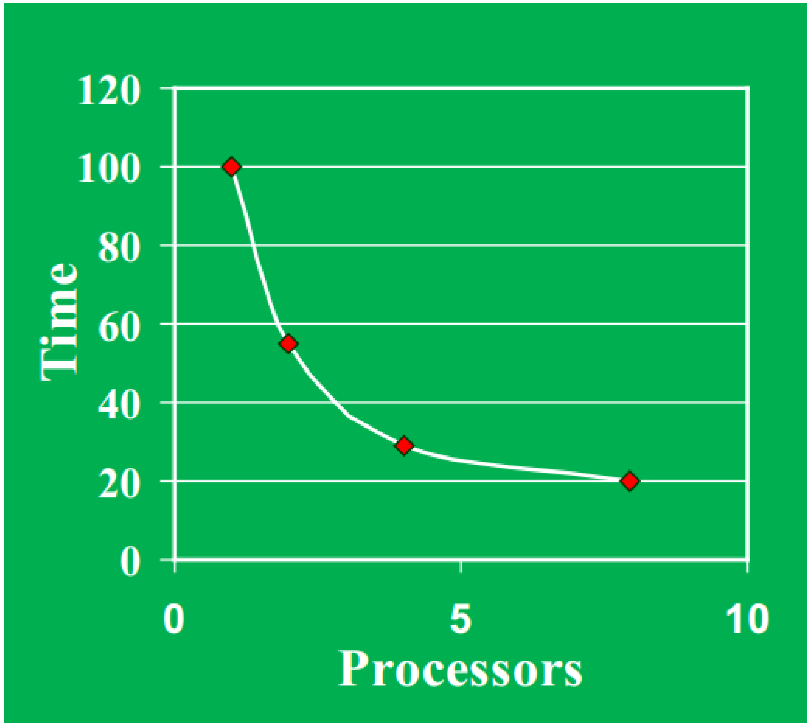
Strong vs. Weak scaling

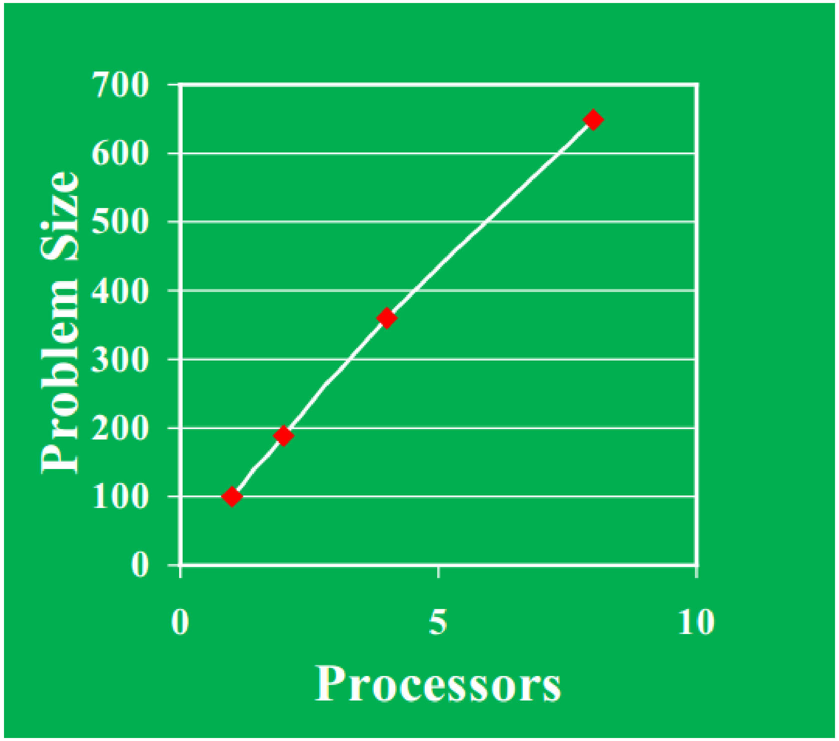
- A: “Warm body” waiting in front of the computer: problem size is constant
  - Strong scaling
  - Best modeled with Amdahl’s law

- B: Want to get the most done in a certain amount of time: compute time is constant
  - Weak scaling
  - Best modeled with Gustafson’s law

SPEEDUP vs. THROUGHPUT

Speedup vs. Throughput





Graphs from S. Jarp

Gustafson's Law

Consequences

Practice

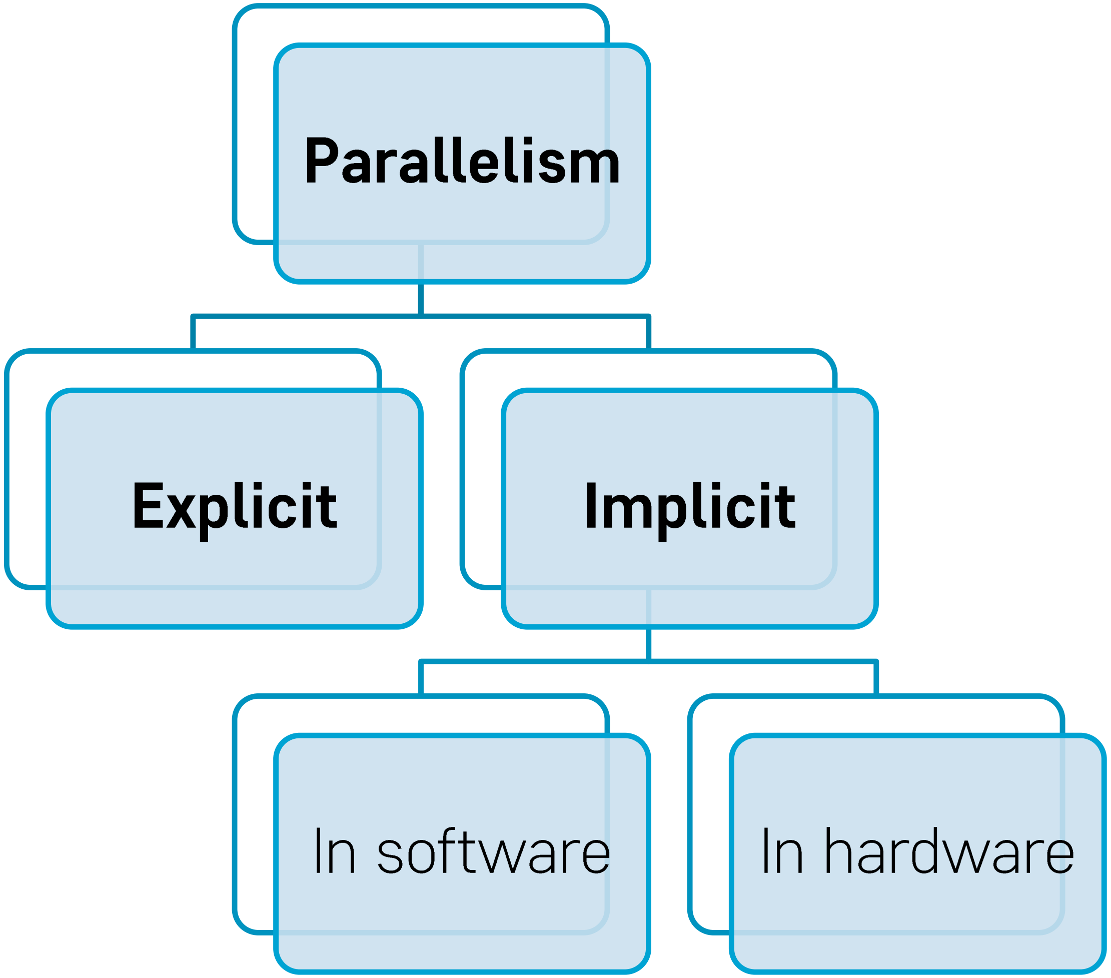
A thousand questions

- Which hardware? How many cores? Where?
- Which software? Will it scale?
- How to share data? How to communicate?
- Which OS?
- Which libraries? Are they thread-safe? Will they generate contention? Which API?
- How to express parallelism? Which programming language?
- Established standards or novel trinkets?
- How do I pay for the power?

Algorithms and structures
Parallel vs. Sequential

- Simple example: add N numbers
  - Sequential: N operations
  - Parallel: log N operations
- Structures
  - Private or shared
  - Maps, arrays, copy and write algorithms
  - Exchangers
  - Pools, arenas

Expressing parallelism



Parallelism primitives

- Threads
- Mutexes
  - Standard
  - Spinlocks
  - Recursive
  - Timed
  - Hierarchical
- Semaphores, barriers...

It's not all unicorns and rainbows

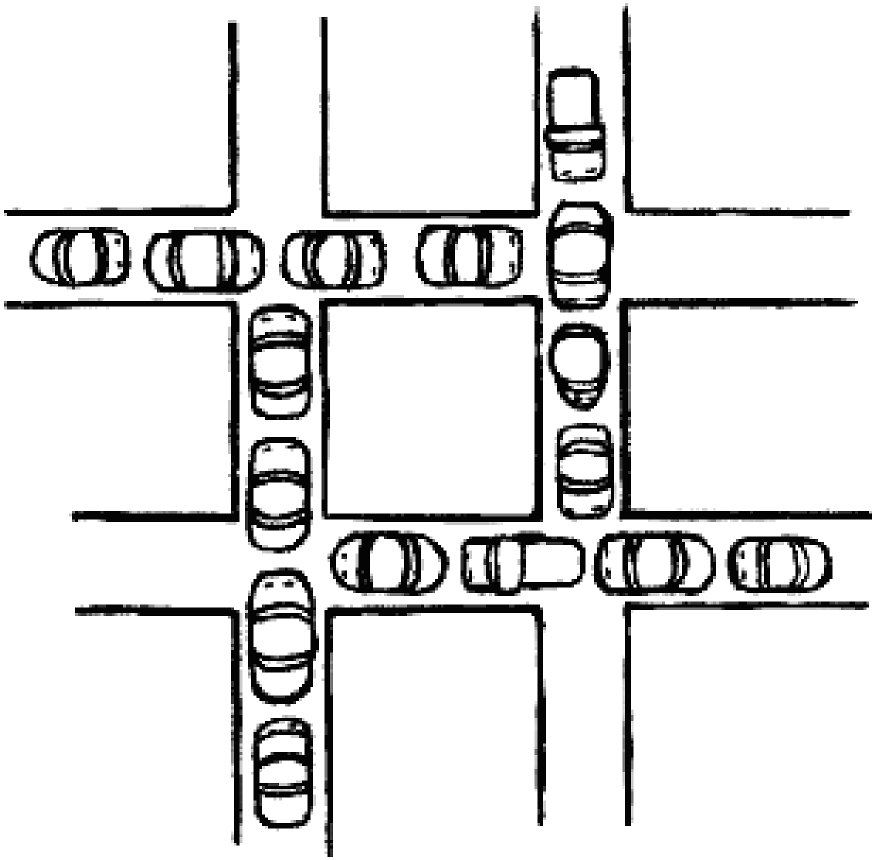
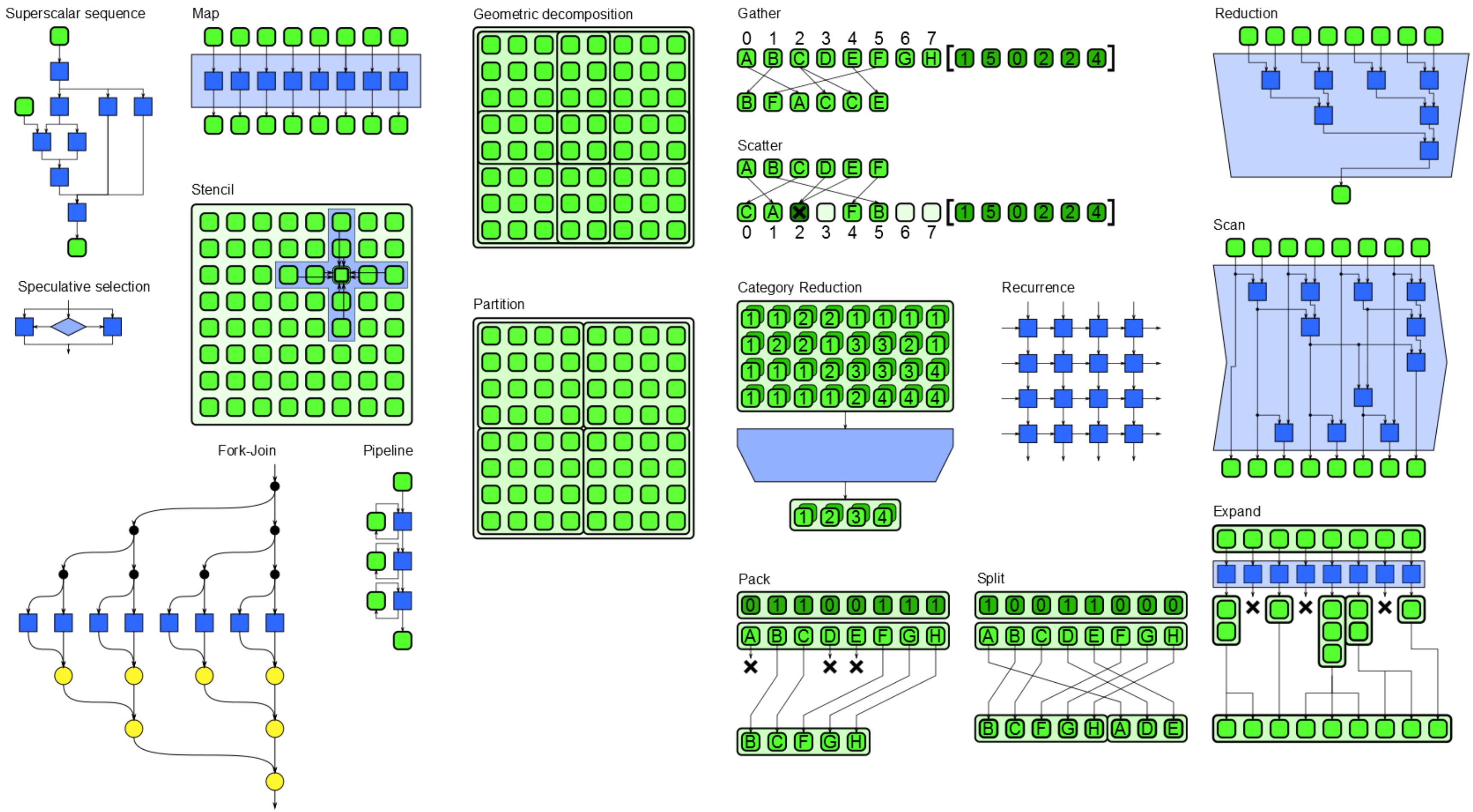


Image: Denning Institute

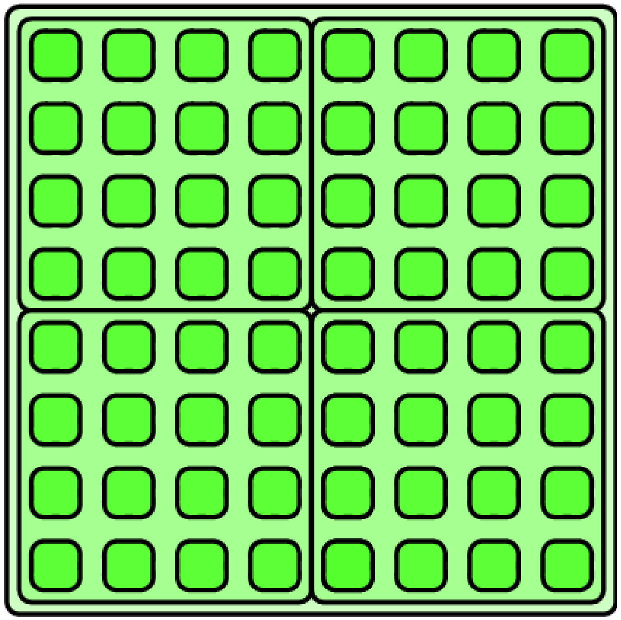
Parallel patterns

According to McCool, Robinson, Reinders

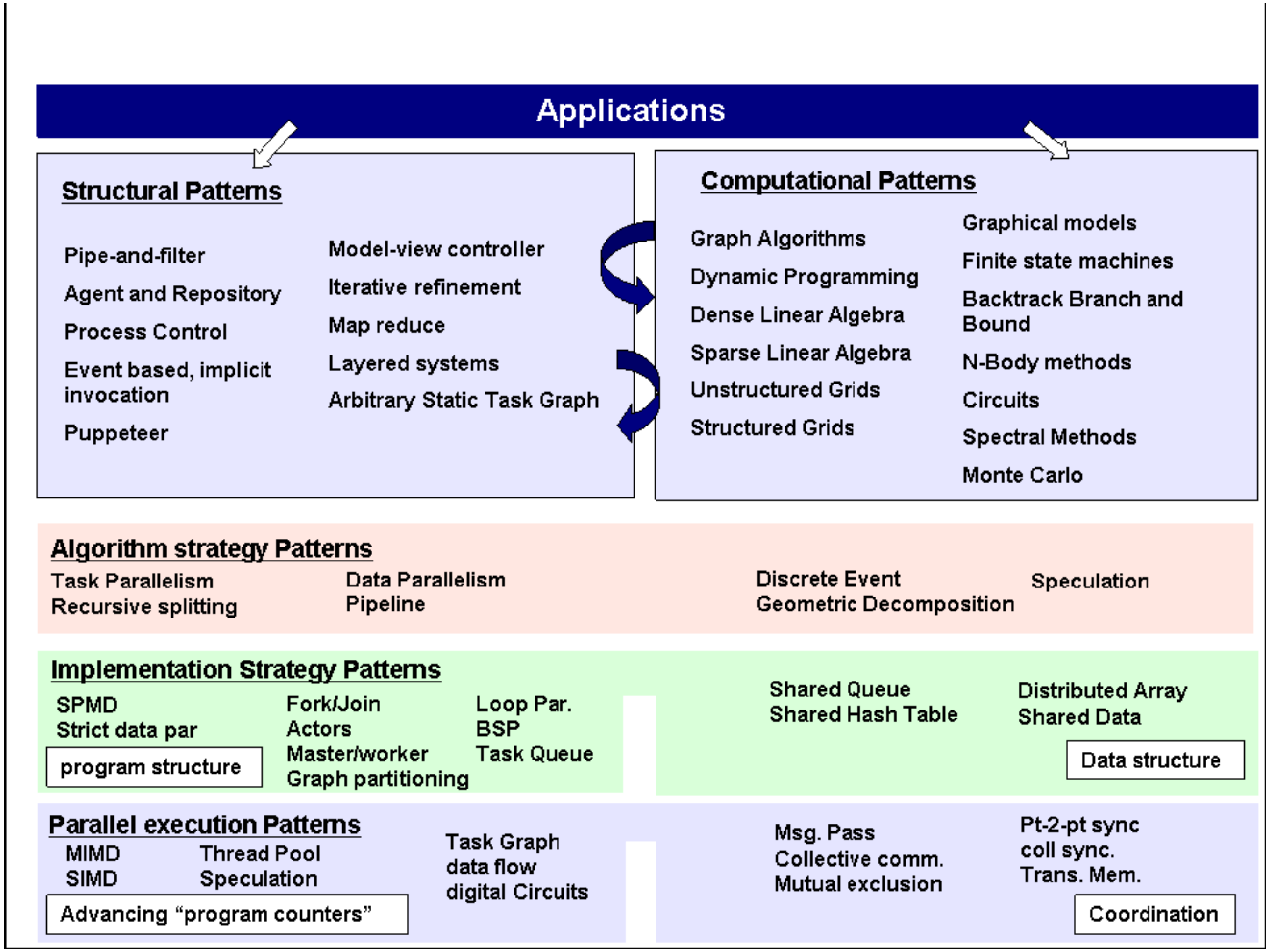


Example: Partitioning

- Divide and conquer – break the input into smaller pieces
- Data movement is not necessarily required, it could be just a matter of the view
- Can be in N dimensions



A Pattern Language for Parallel Programming



Applications

Structural Patterns

Pipe-and-filter
Agent and Repository
Process Control
Event based, implicit invocation
Puppeteer

Model-view controller
Iterative refinement
Map reduce
Layered systems
Arbitrary Static Task Graph

Computational Patterns

Graph Algorithms
Dynamic Programming
Dense Linear Algebra
Sparse Linear Algebra
Unstructured Grids
Structured Grids

Graphical models
Finite state machines
Backtrack Branch and Bound
N-Body methods
Circuits
Spectral Methods
Monte Carlo

Algorithm strategy Patterns

Task Parallelism
Recursive splitting

Data Parallelism
Pipeline

Discrete Event
Geometric Decomposition

Speculation

Implementation Strategy Patterns

SPMD
Strict data par

program structure

Fork/Join
Actors
Master/worker
Graph partitioning

Loop Par.
BSP
Task Queue

Shared Queue
Shared Hash Table

Distributed Array
Shared Data

Data structure

Parallel execution Patterns

MIMD
SIMD

Thread Pool
Speculation

Advancing “program counters”

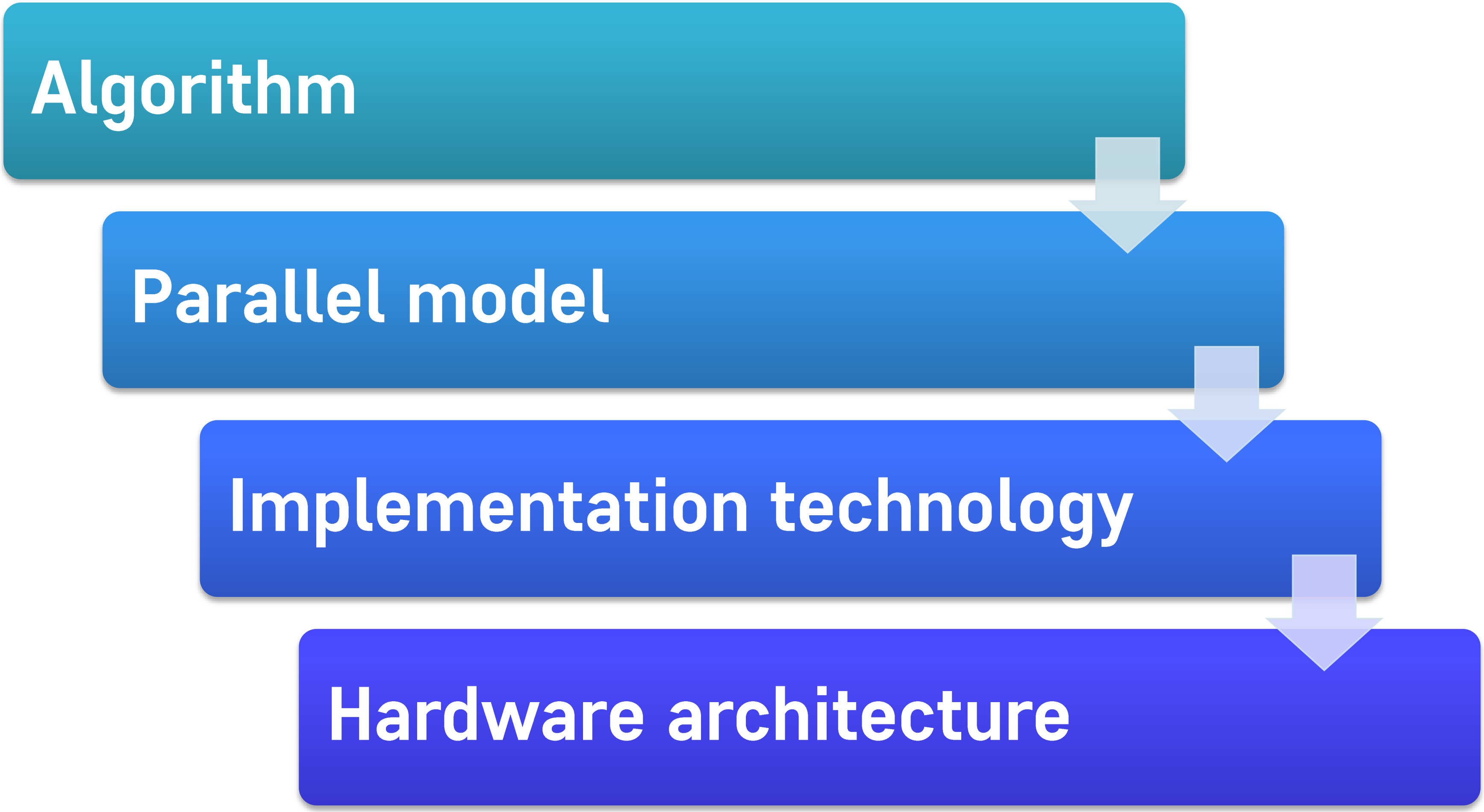
Task Graph
data flow
digital Circuits

Msg. Pass
Collective comm.
Mutual exclusion

Pt-2-pt sync
coll sync.
Trans. Mem.

Coordination

Decomposition and mapping



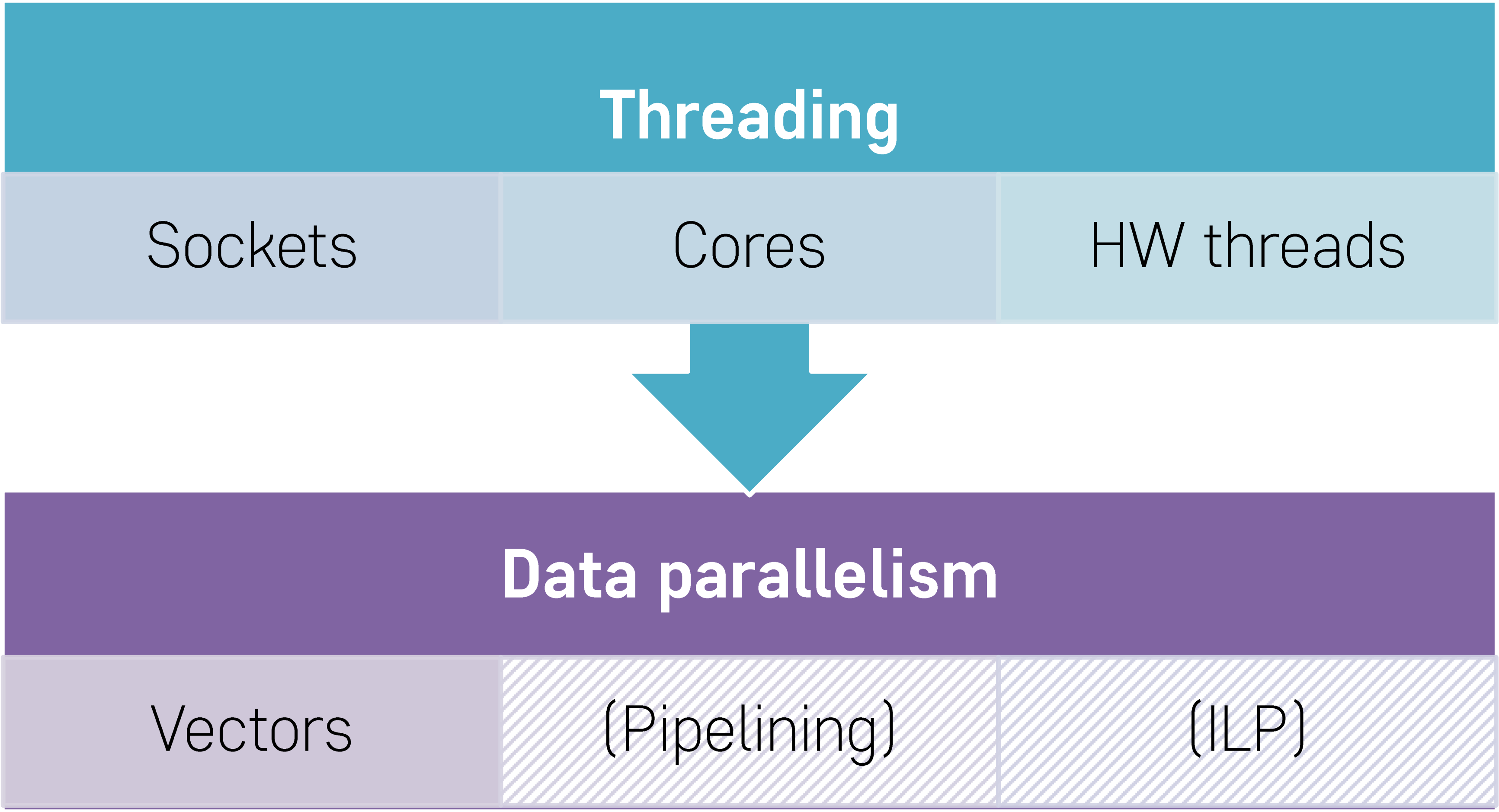
Algorithm

Parallel model

Implementation technology

Hardware architecture

Decomposition and mapping



Threading

Sockets

Cores

HW threads

Data parallelism

Vectors

(Pipelining)

(ILP)

Thank you

e-mail: an@tik.services

http://tik.services