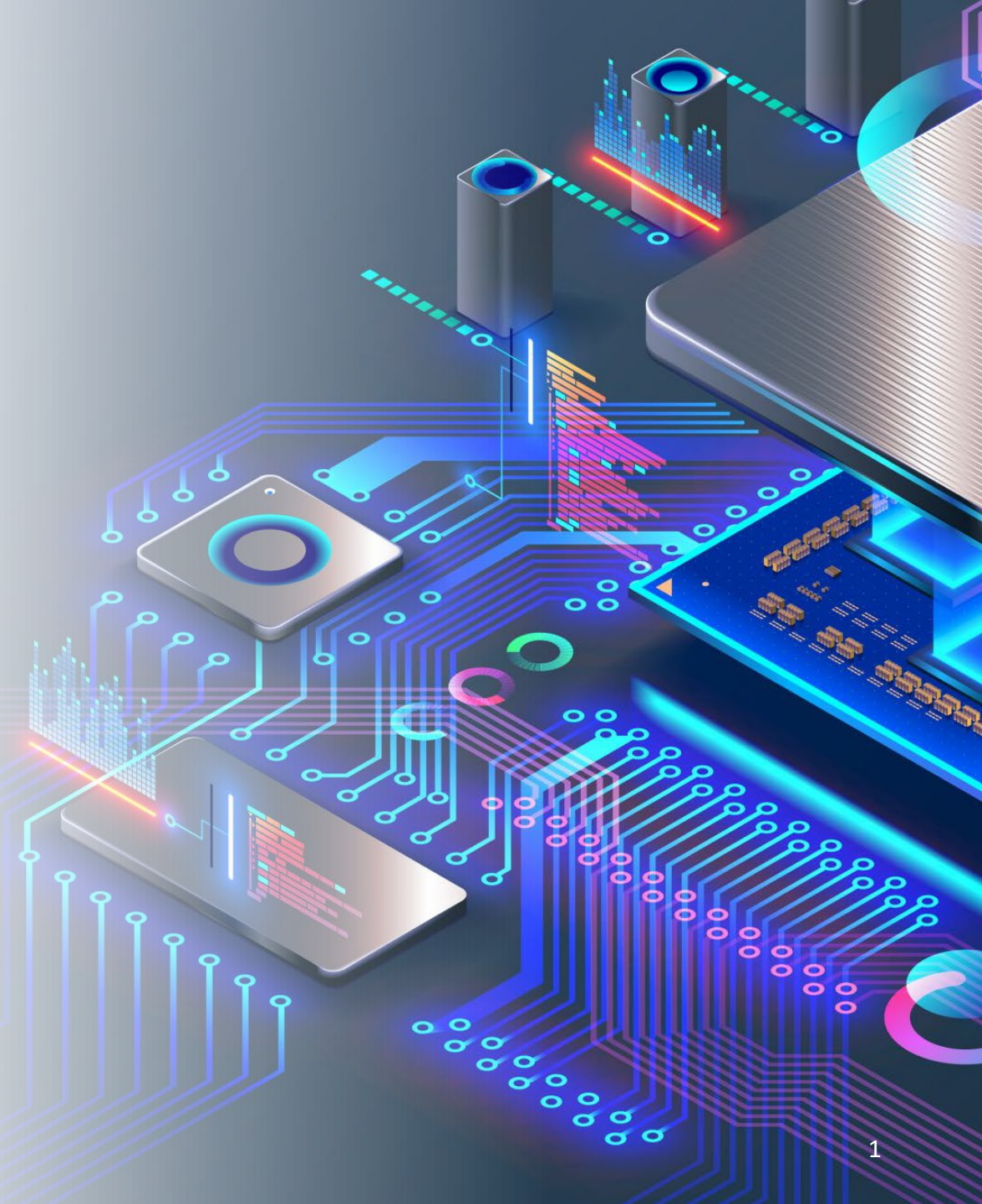
Introduction

KT44502 DISTRIBUTED SYSTEM

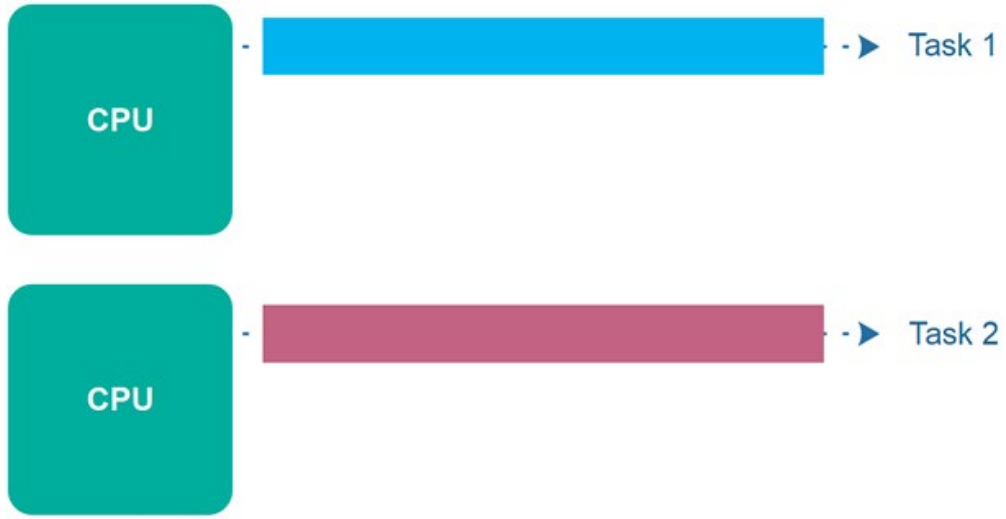
Semester 1 2022/2023



Parallel

A parallel computer is a type of computer that uses multiple processors or cores to perform tasks simultaneously. This allows the computer to perform multiple tasks at the same time and can potentially improve the speed and performance of certain types of computations. Parallel computers can be useful for a variety of applications, including scientific simulations, data analysis, and machine learning.

- A parallel computer is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem
- Parallel processing includes techniques and technologies that make it possible to compute in parallel
  - Hardware, networks, operating systems, parallel libraries, languages, compilers, algorithms, tools, …
- Parallel computing is an evolution of serial computing
  - Parallelism is natural
  - Computing problems differ in level / type of parallelism

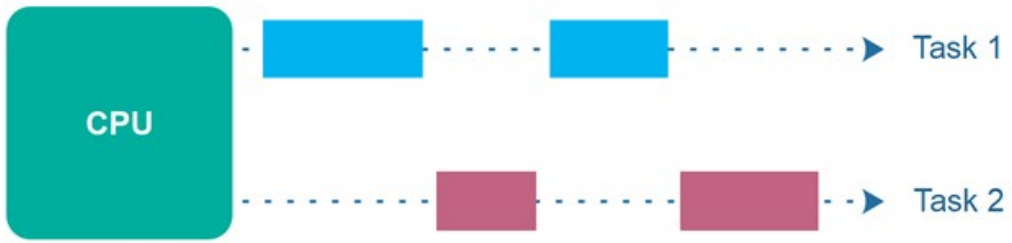


Concepts

- **Parallel computer** - A multiple-processor computer system that supporting parallel programming and computation.
- **Parallel Computing** - The use of parallel computer to reduce the time needed to solve a single computational problem.
- **Parallel Programming** – Computer programming in a language that allows programmers to indicate how different portions of computation may be executed concurrently by different processors.

Concurrency

- Consider multiple tasks to be executed in a computer
- Tasks are concurrent with respect to each if
  - They can execute at the same time (concurrent execution) Implies that there are no dependencies between the tasks
- Dependencies
  - If a task requires results produced by other tasks in order to execute correctly, the task's execution is dependent
  - If two tasks are dependent, they are not concurrent Some form of synchronization must be used to enforce (satisfy) dependencies
- Concurrency is fundamental to computer science
  - Operating systems, databases, networking, …



Task need results from others task=dependent
Dependent=no concurrent

Concurrency and Parallelism

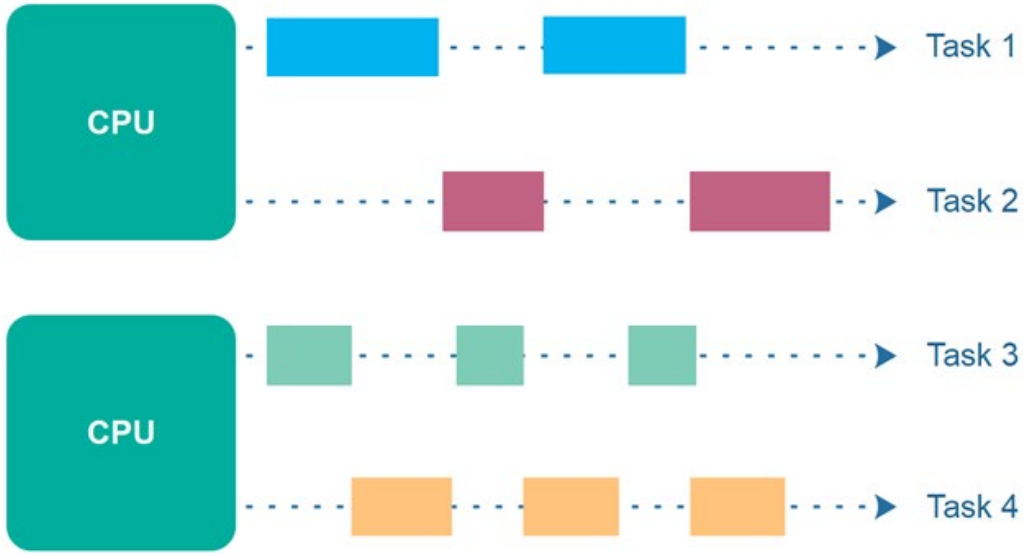
- Refer to related but different things.
- Concurrent (multiple threads control, thread x run in parallel)
  - A concurrent program has multiple logical threads of control. These threads may or may not run in parallel.
- Parallel

(run fast = executing diff part of computation in parallel)

  - A parallel program potentially runs more quickly than a sequential program by executing different parts of the computation simultaneously (in parallel).
  - It may or may not have more than one logical thread of control.

Parallel Concurrent Execution

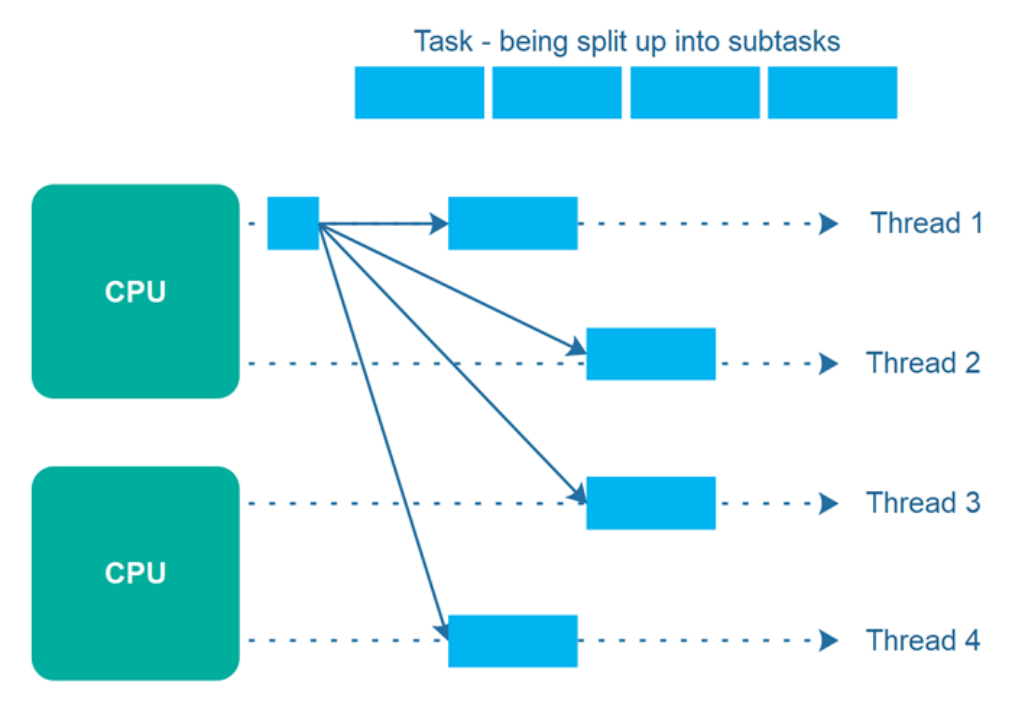
- Making progress on more than one task seemingly at the same time on more than on CPU.
- Concurrent is not the same as parallel!
- Why?
- Parallel execution:
  - Concurrent tasks execute at the same time
  - Multiple (processing) resources must be available



Concurrency =computer program or system to handle multiple tasks at the same time//multiple processors or cores, or by using a single processor to switch rapidly between tasks
Parallelism, on the other hand, refers specifically to the execution of multiple tasks simultaneously, using multiple processors or cores

Parallelism

- An application splits its tasks up into smaller subtasks which can be processed in parallel, for instance on multiple CPUs at the exact same time.
- There are granularities of parallelism (parallel execution) in programs
  - Processes, threads, routines, statements, instructions, …
  - Think about what are the software elements that execute concurrently
- These must be supported by hardware resources
  - Processors, cores, … (execution of instructions)
  - Memory, DMA, networks, … (other associated operations)
  - All aspects of computer architecture offer opportunities for parallel hardware execution
- Concurrency is a necessary condition for parallelism

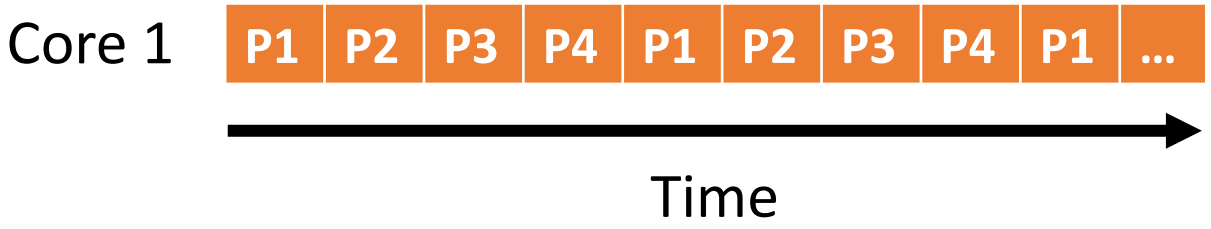


Concurrency and Parallelism

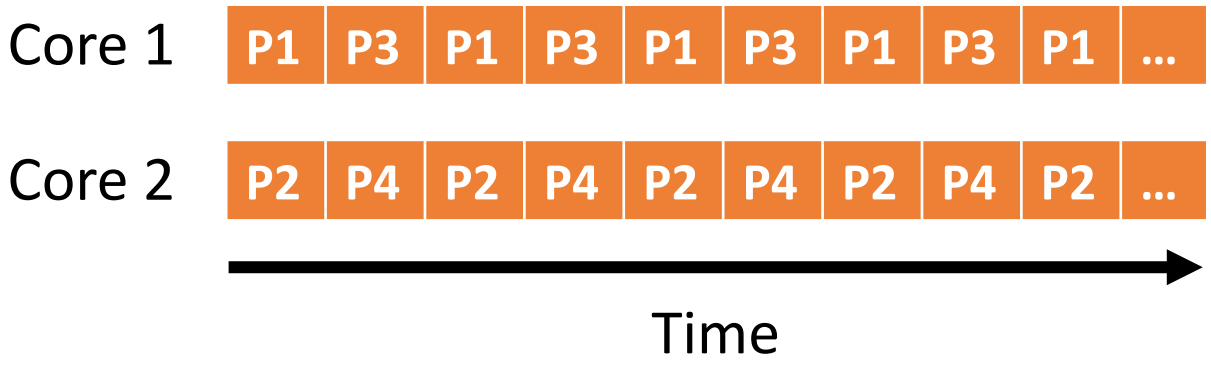
- Running parallel threads of execution is important
- Making use of parallelism implies concurrency
  - Multiple actions running at the same time
  - Or executed, or appears to do certain tasks
- This can be difficult
  - Two tasks or executions that may depends on the other executions
  - Thus, parallelism will make the execution order is non-deterministic
  - If it not controlled (synchronized), output could be different and intractable

Concurrency vs. Parallelism

- Concurrent execution on a single-core system:



- Parallel execution on a dual-core system:



Joe asks:

Concurrent or Parallel?

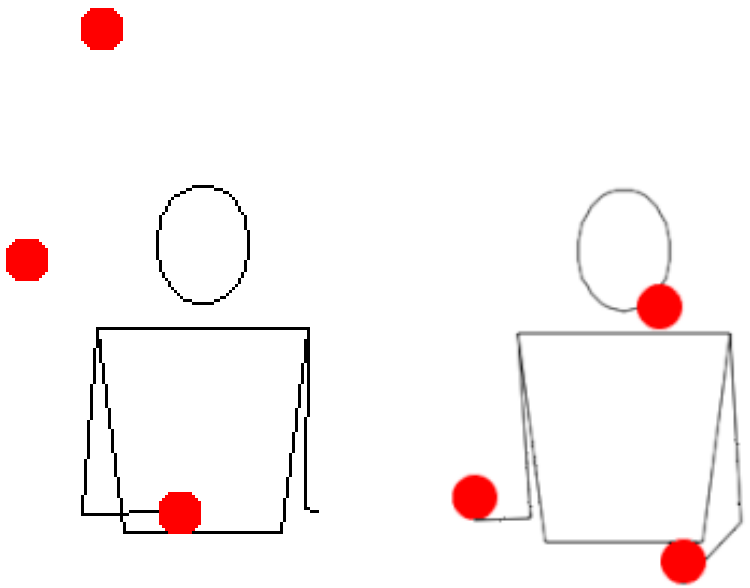
My wife is a teacher. Like most teachers, she's a master of multitasking. At any one instant, she's only doing one thing, but she's having to deal with many things concurrently. While listening to one child read, she might break off to calm down a rowdy classroom or answer a question. This is concurrent, but it's not parallel (there's only one of her).

If she's joined by an assistant (one of them listening to an individual reader, the other answering questions), we now have something that's both concurrent and parallel.

Imagine that the class has designed its own greeting cards and wants to mass-produce them. One way to do so would be to give each child the task of making five cards. This is parallel but not (viewed from a high enough level) concurrent—only one task is underway.

Other examples

- In juggling a 3 balls by a person
  - How is it performed?
- In construction of houses or buildings
  - What tasks?
- In manufacturing
  - What example?
- In customer service such as call center
  - How parallel is achieved?



Activity

- Give your own example, each for concurrent and parallel application/process
- When is it become concurrent and parallel?

- Questions
  - Is a parallel application a concurrent application?
  - Is a parallel application run with one processor parallel? Why or why not?
  - Where can you find concurrency?
  - How is concurrency expressed to exploit parallel systems?

Why Parallel Processing?

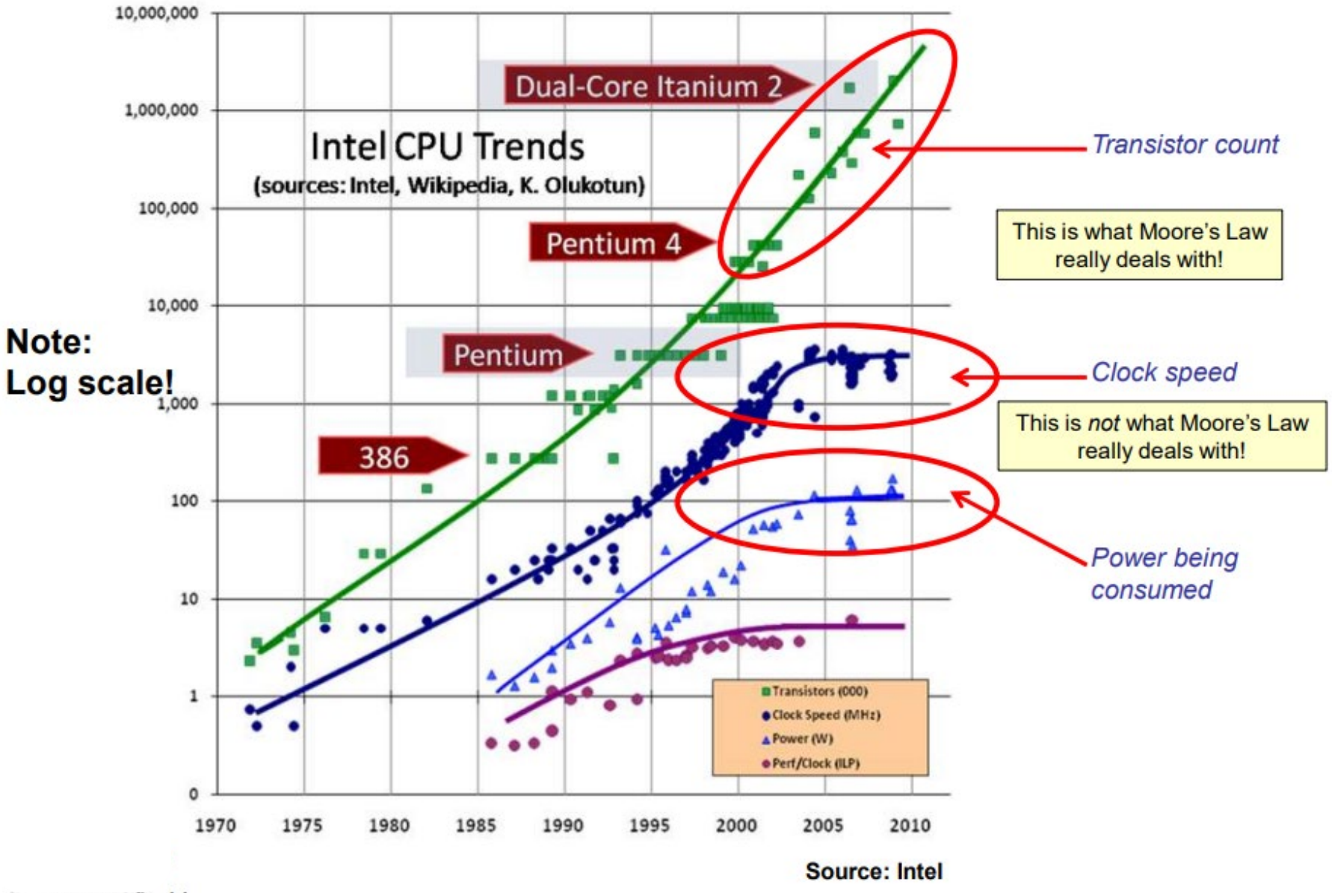
- Two primary reasons (both performance related)
  - Faster time to solution (response time)
  - Solve bigger computing problems (in same time)
- Other factors motivate parallel processing
  - Effective use of machine resources Cost efficiencies
  - Overcoming memory constraints
- Serial machines have inherent limitations
  - Processor speed, memory bottlenecks, … Parallelism has become the future of computing Performance is still the driving concern

Why Parallel Processing?

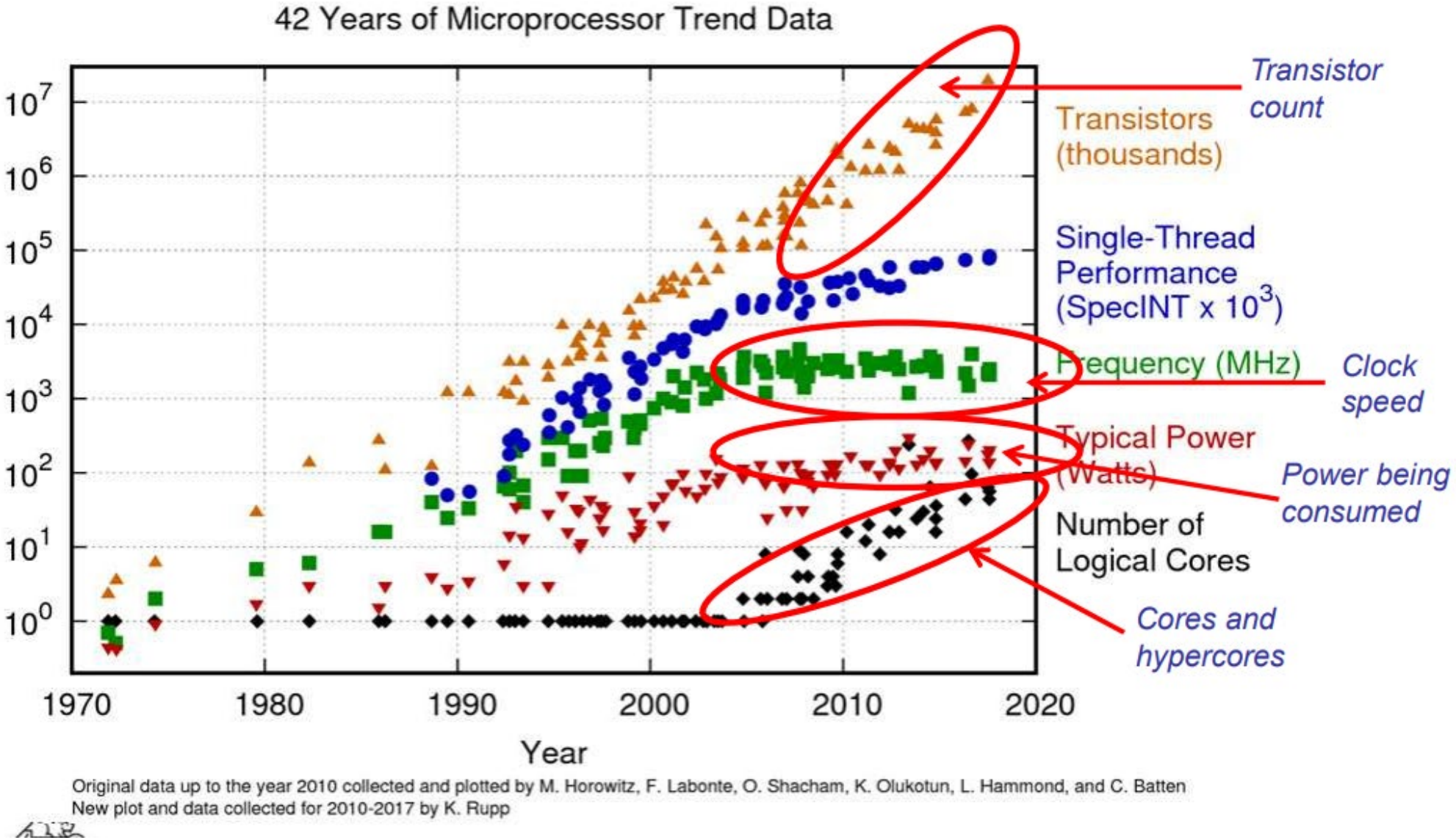
- Moore's law is the observation that the number of transistors in a dense integrated circuit doubles about every two years.
- Fabrication process size (“gate pitch”) has fallen from 65 nm, to 45 nm, to 32 nm, to 22 nm, to 16 nm, to 11 nm, to 8 nm, to 7 nm, to 5 nm, to 4 nm. This translates to more transistors on the same size die.
- From 1986 to 2002, processor performance increased an average of 52%/year, but then virtually plateaued.

Year	Processor Name	Clock speed
1981	IBM PC	5 MHz
1995	Pentium	100 MHz
2002	Pentium 4	3000 MHz (3 GHz)
2007		3800 MHz (3.8 GHz)
2009		4000 MHz (4.0 GHz)

Why Parallel Processing?



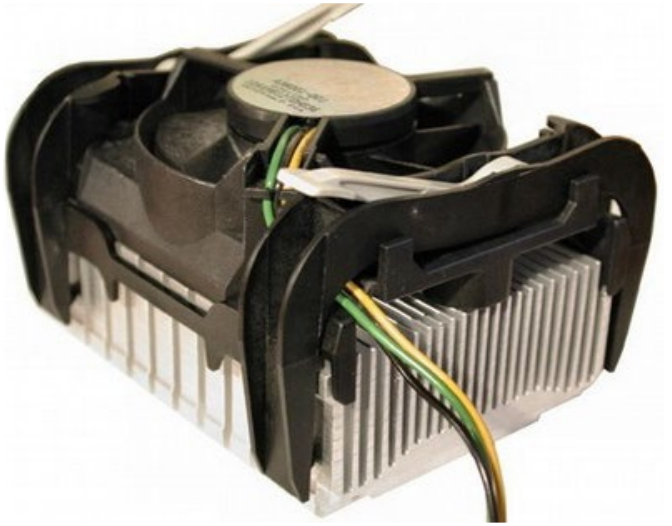
Why Parallel Processing?



And, speaking of “exotic”, AMD set the world record for clock speed (8.429 GHz) using a Liquid Nitrogen-cooled CPU

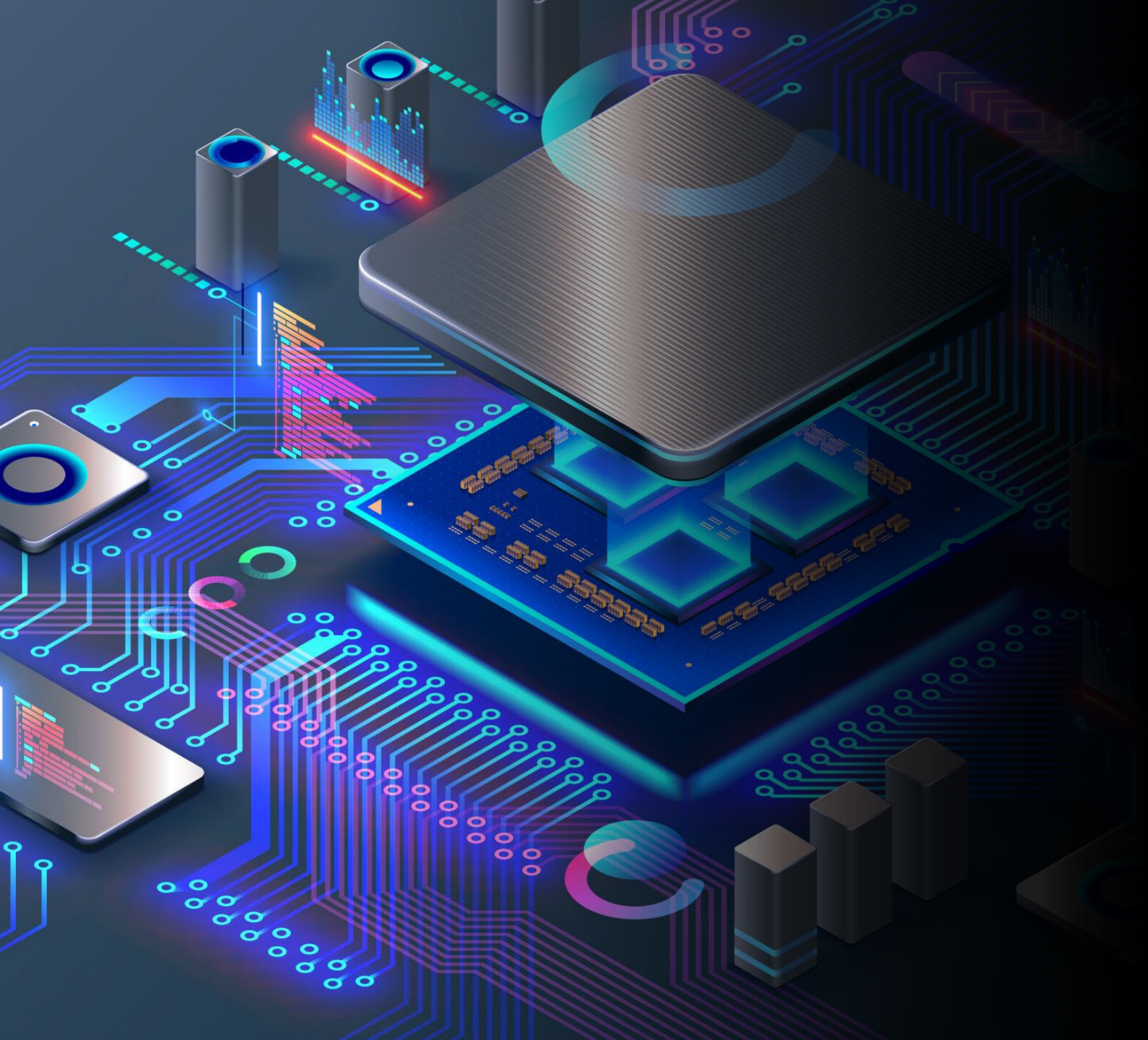
https://www.youtube.com/watch?v=UKN4VMOenNM&ab_channel=AMDUnprocessed

What has happened in the last several years?



- Processing chip manufacturers increased processor performance by increasing CPU clock frequency
  - Riding Moore's law
- Until the chips got too hot!
  - Greater clock frequency = greater electrical power
  - Pentium 4 heat sink = Frying an egg on a Pentium 4

- Add multiple cores to add performance
  - Keep clock frequency same or reduced
  - Keep lid on power requirements



Thank You

Next: Parallel Computer Architecture