Lecture 1.1
Introduction to Parallel Programming

EN 600.320/420

Instructor: Randal Burns

29 January 2018

Department of Computer Science, *Johns Hopkins University*

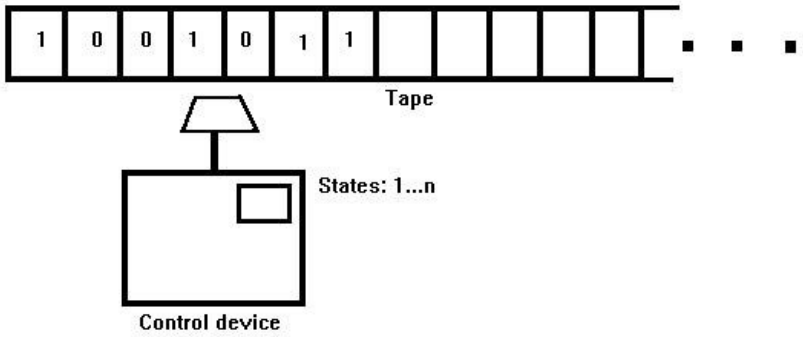
Why this class?

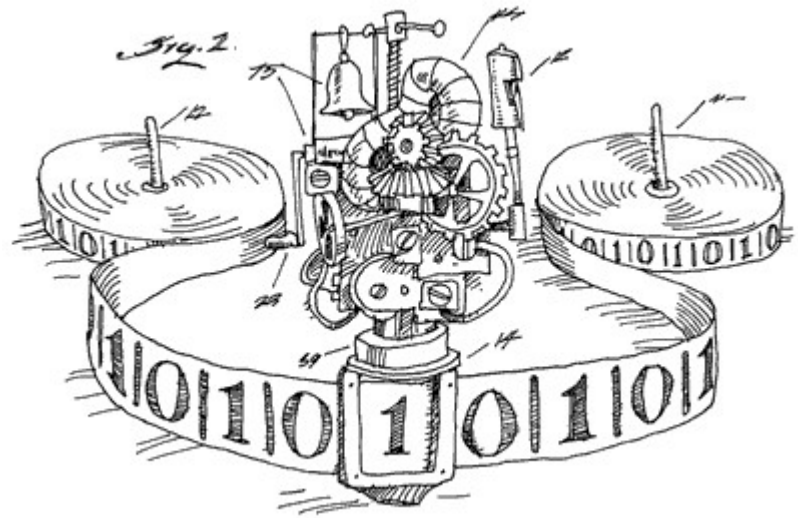
- Parallelism is everywhere!
  - Every program/programmer needs to address it
- Traditional CS curriculum totally misses the point

Who should take it?

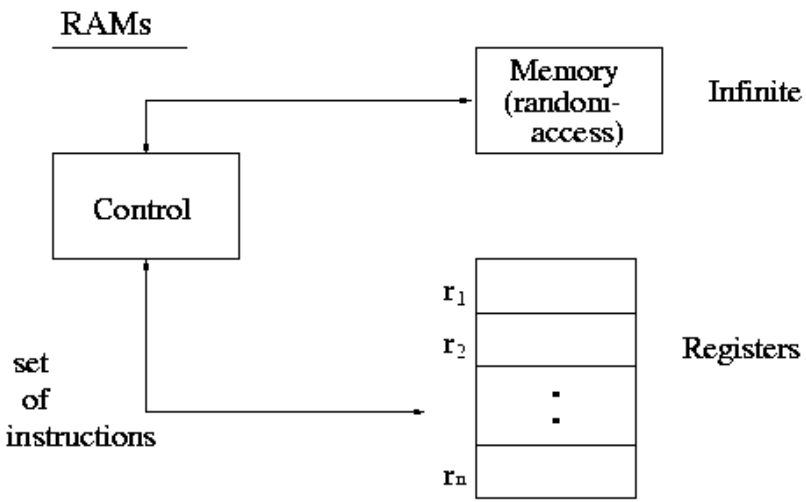
- Anyone who needs to program ever. Period! Exclamation point!! …
- Designed for Sophomores: quick lift of skills good for employers and internships

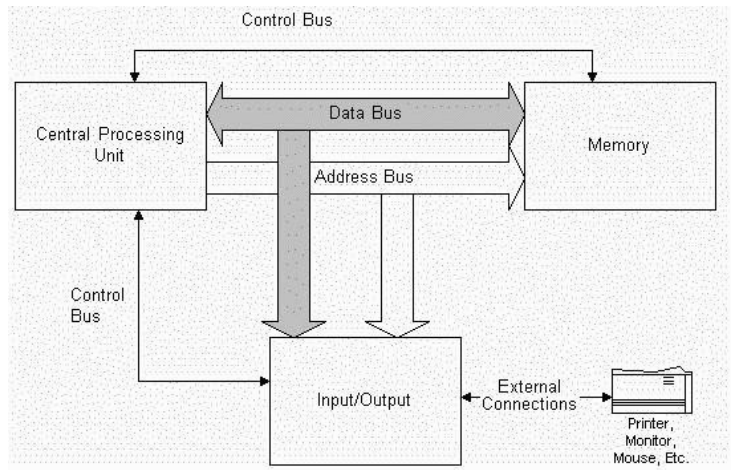
What's a computer look like?





http://www.cis.upenn.edu/~dietzd/CIT596/turingMachine.gif





Notions of a Computer

- Sequential processing
  - Control or logic flow
- Algorithm costs measured in this model
  - Big-O notation counts number of sequential steps
- This is the basis for the CS curriculum
  - And it's just wrong
  - Computers are not sequential and performance is more nuanced than counting the number of steps
- I demand a new curriculum
  - That treats computers as parallel entities, programs concurrency and parallelism, and makes the world a better place

Wrong you say?

- Realities of computing
  - We have tons of wasted cycles
  - CPU utilization typically <10% (of useful work)
- Many other things limit performance
  - Pipeline stalls
  - Lock interference
  - Waiting for I/O and network
  - Data dependencies
- Writing serial programs is broken
  - Parallelism is everywhere
  - Must exploit it to realize time efficiency, power savings

MonetDB/X100: Hyper-Pipelining Query Execution

Peter Boncz, Marcin Zukowski, Niels Nes

Modern CPUs can perform enormous amounts of calculations per second, but only if they can find enough independent work to exploit their parallel execution capabilities. Hardware developments during the past decade have significantly increased the speed difference between a CPU running at full throughput and minimal throughput, which can now easily be an order of magnitude.

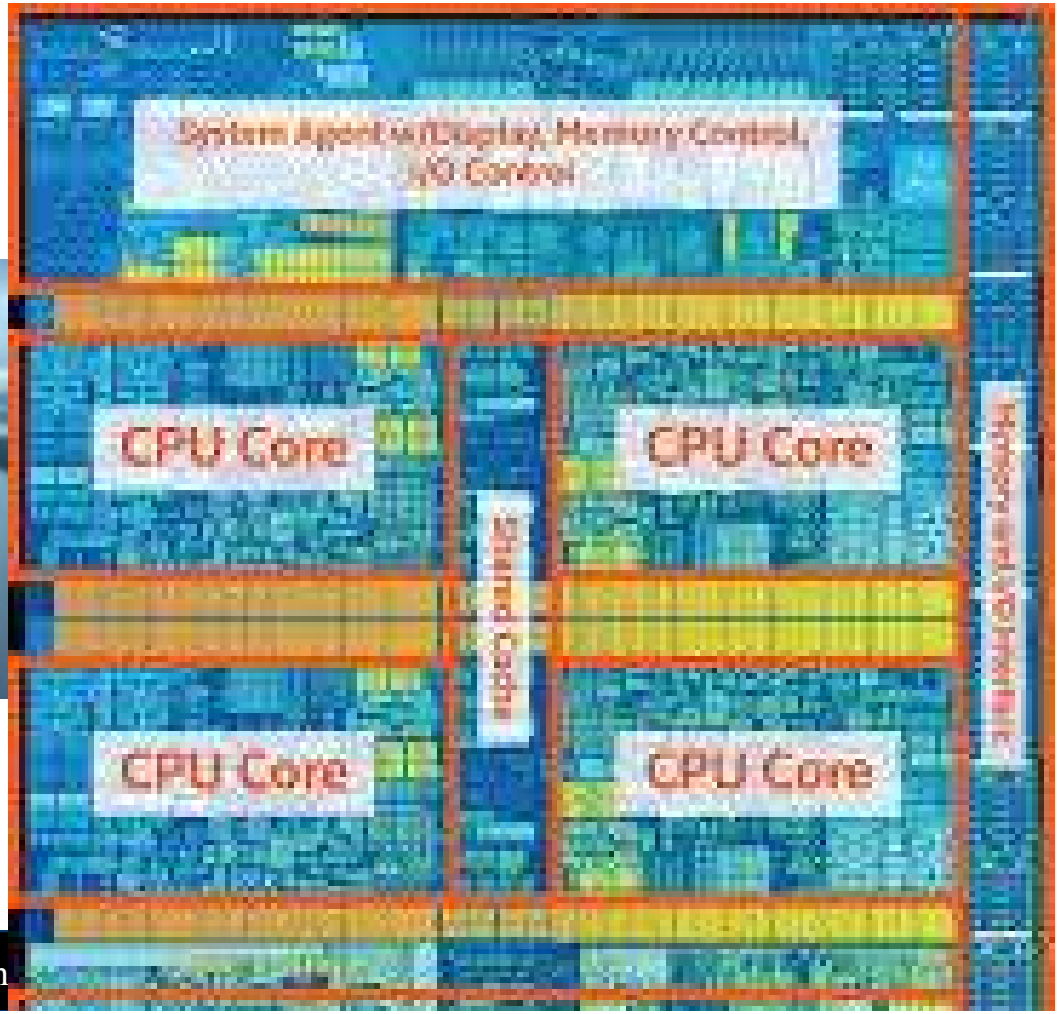
Modern CPUs are balanced in different ways. The Intel Itanium2 processor is a VLIW (Very Large Instruction Word) processor with many parallel pipelines (it can execute up to 6 instructions per cycle) with only few (7) stages, and therefore a relatively low clock speed of 1.5GHz. In contrast, the Pentium4 has its very long 31-stage pipeline allowing for a 3.6GHz clock speed, but can only execute 3 instructions per cycle. Either way, to get to its theoretical maximum throughput, an Itanium2 needs 7x6 = 42 independent instructions *at any time*, while the Pentium4 needs 31x3 = 93.

One would expect that query-intensive database workloads such as decision support, OLAP, data-mining, but also multimedia retrieval, all of which require many independent calculations, should provide modern CPUs the opportunity to get near optimal IPC (instructions-per-cycle) efficiencies.

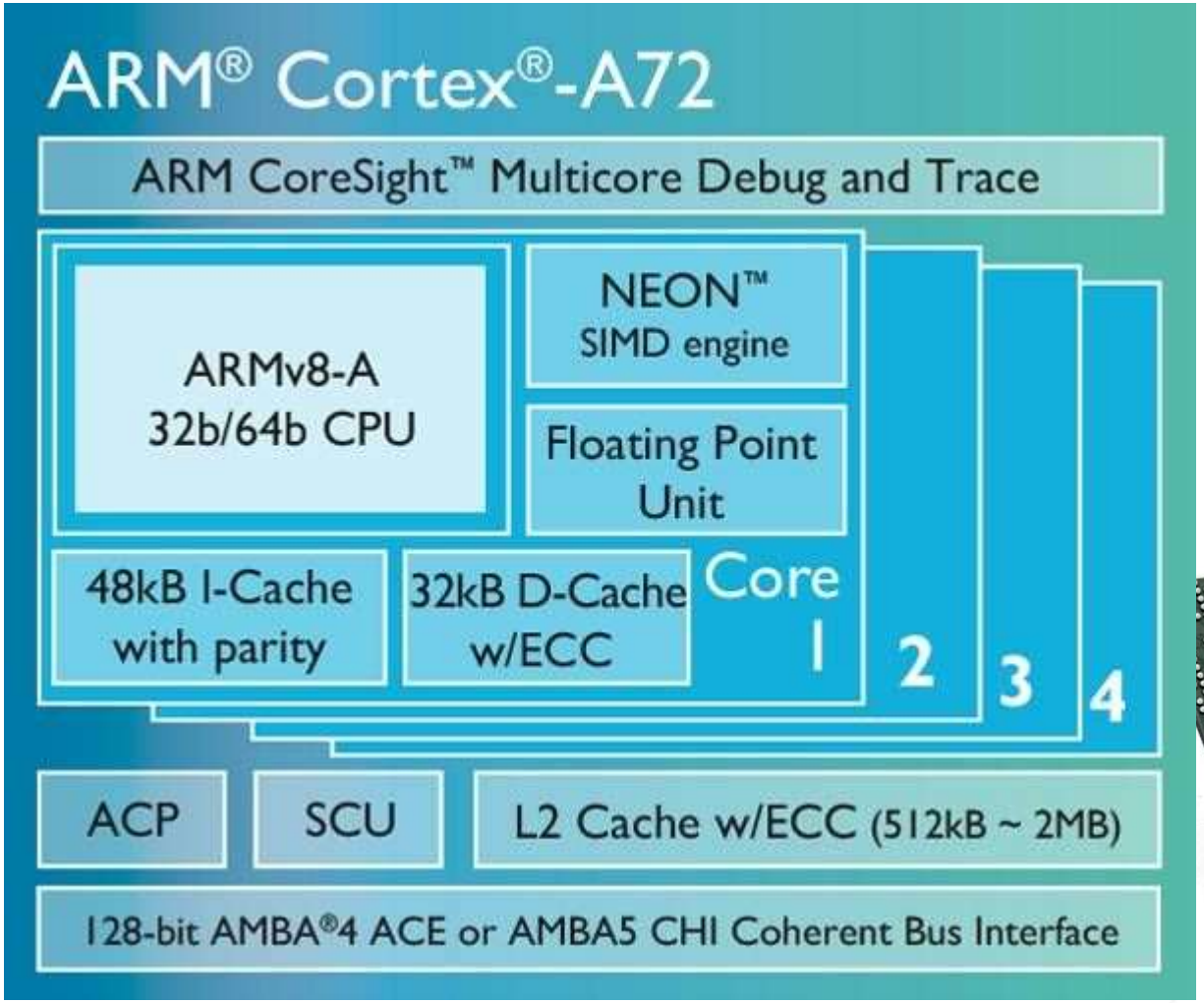
However, research has shown that database systems tend to achieve low IPC efficiency on modern CPUs in on average. It has been shown that query execution in commercial DBMS systems get an IPC of only 0.7 [6], thus executing *less* than one instruction per cycle. In contrast, scientific computation (e.g. matrix multiplication) or multimedia processing does extract average IPCs of up to 2 out of modern CPUs. We argue that

Multi-Core

- Independent processing units on one chip

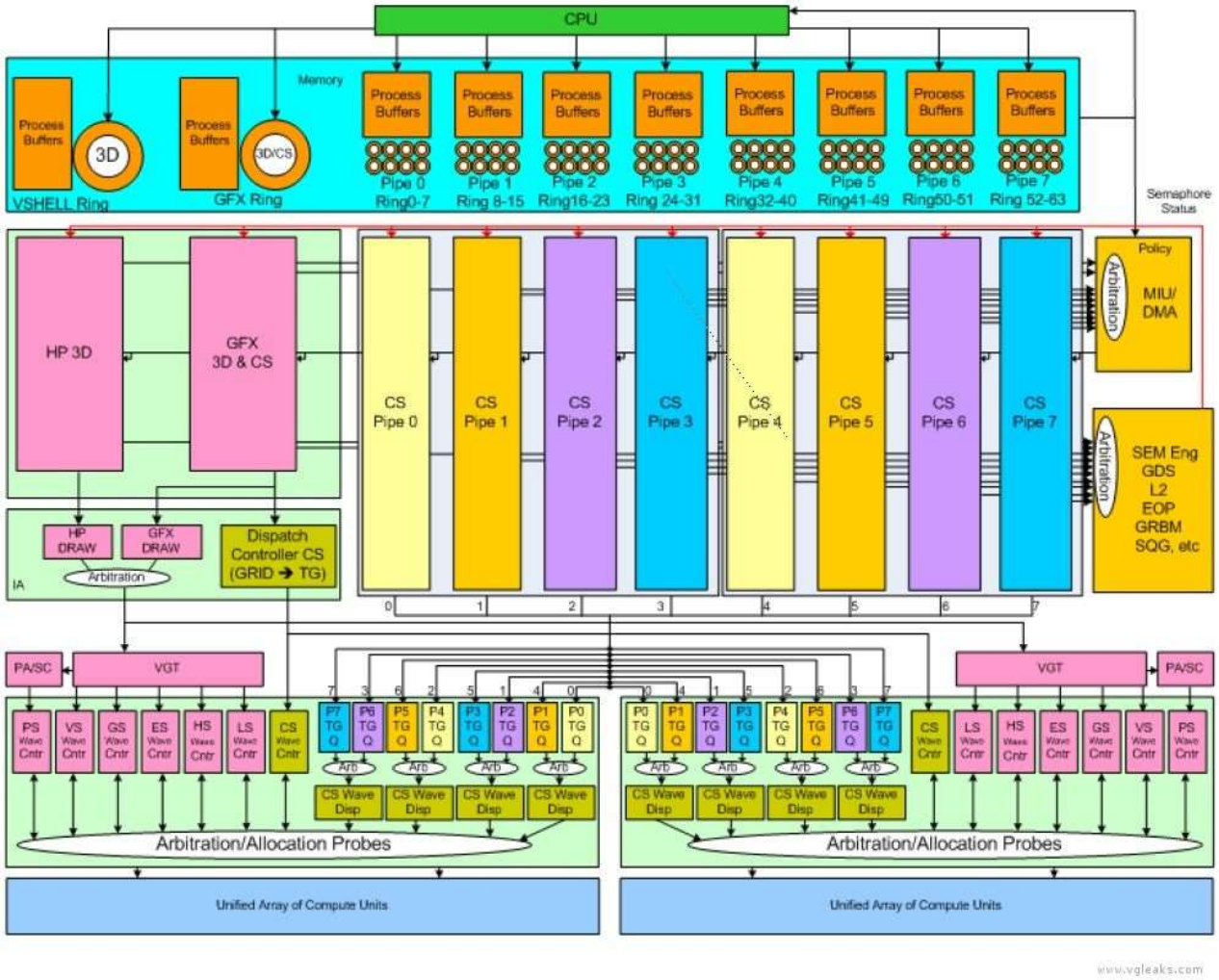


What's a Parallel Computer?



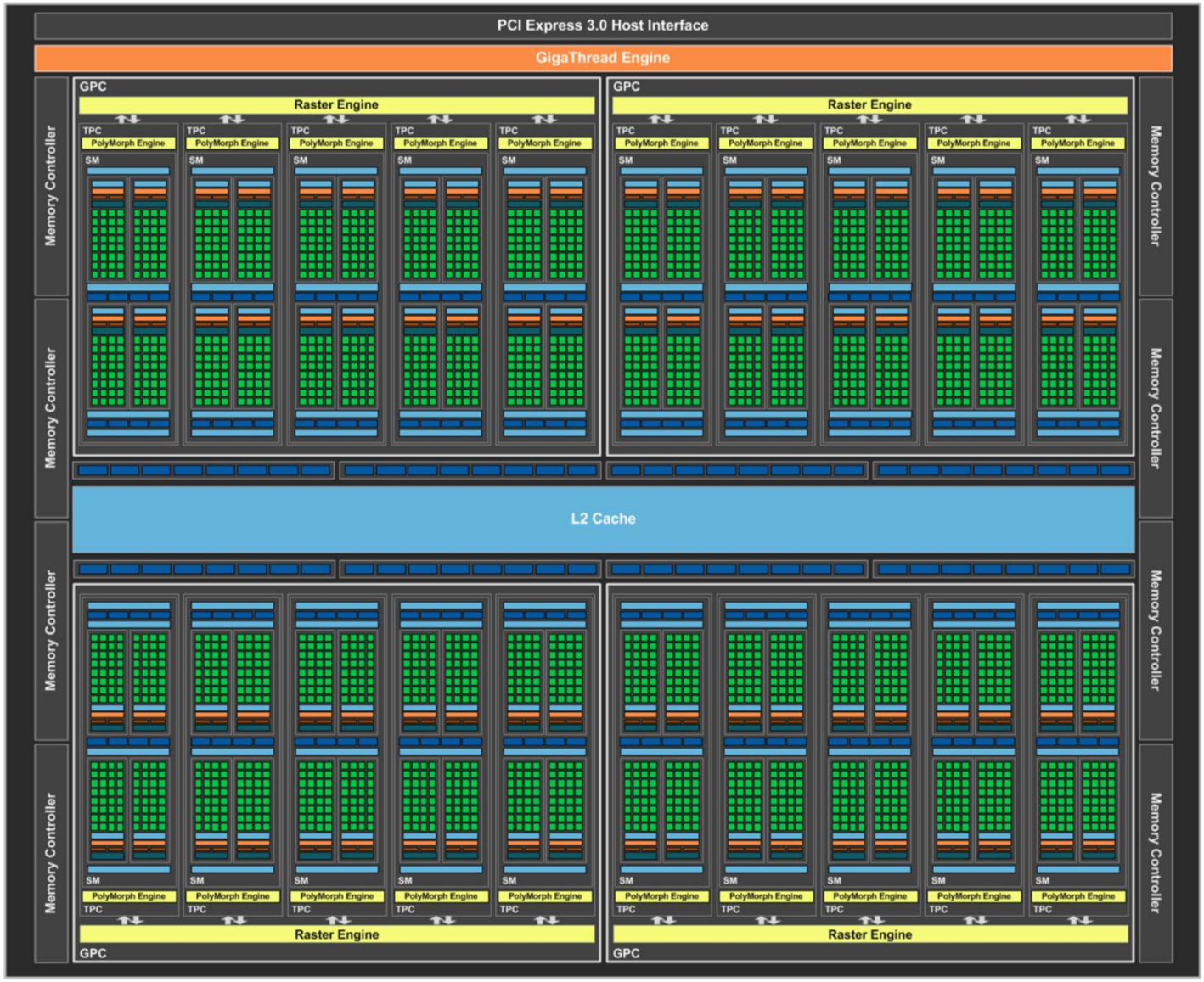
What's a Parallel Computer?

What's a Parallel Computer?

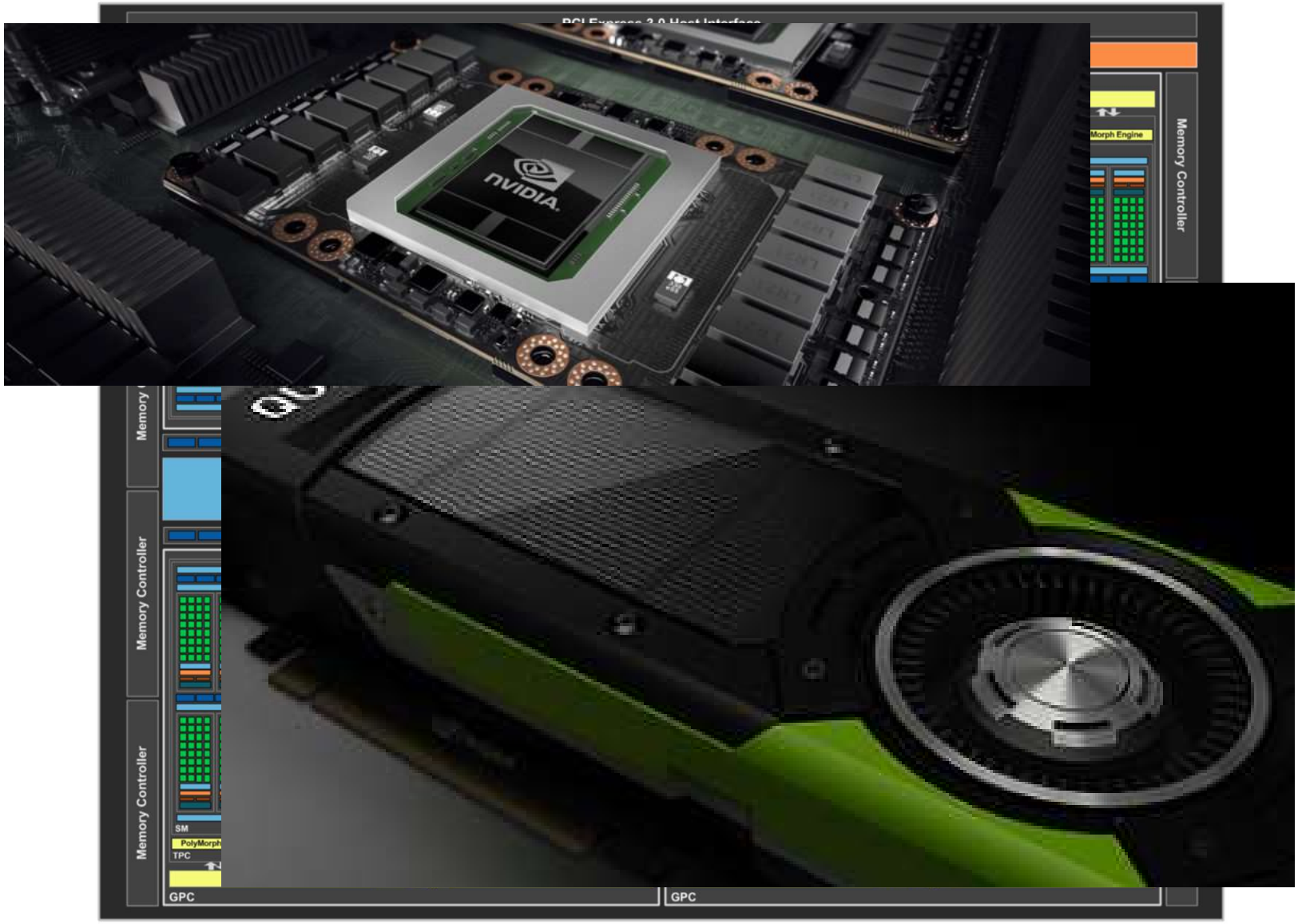


What's a Parallel Computer?

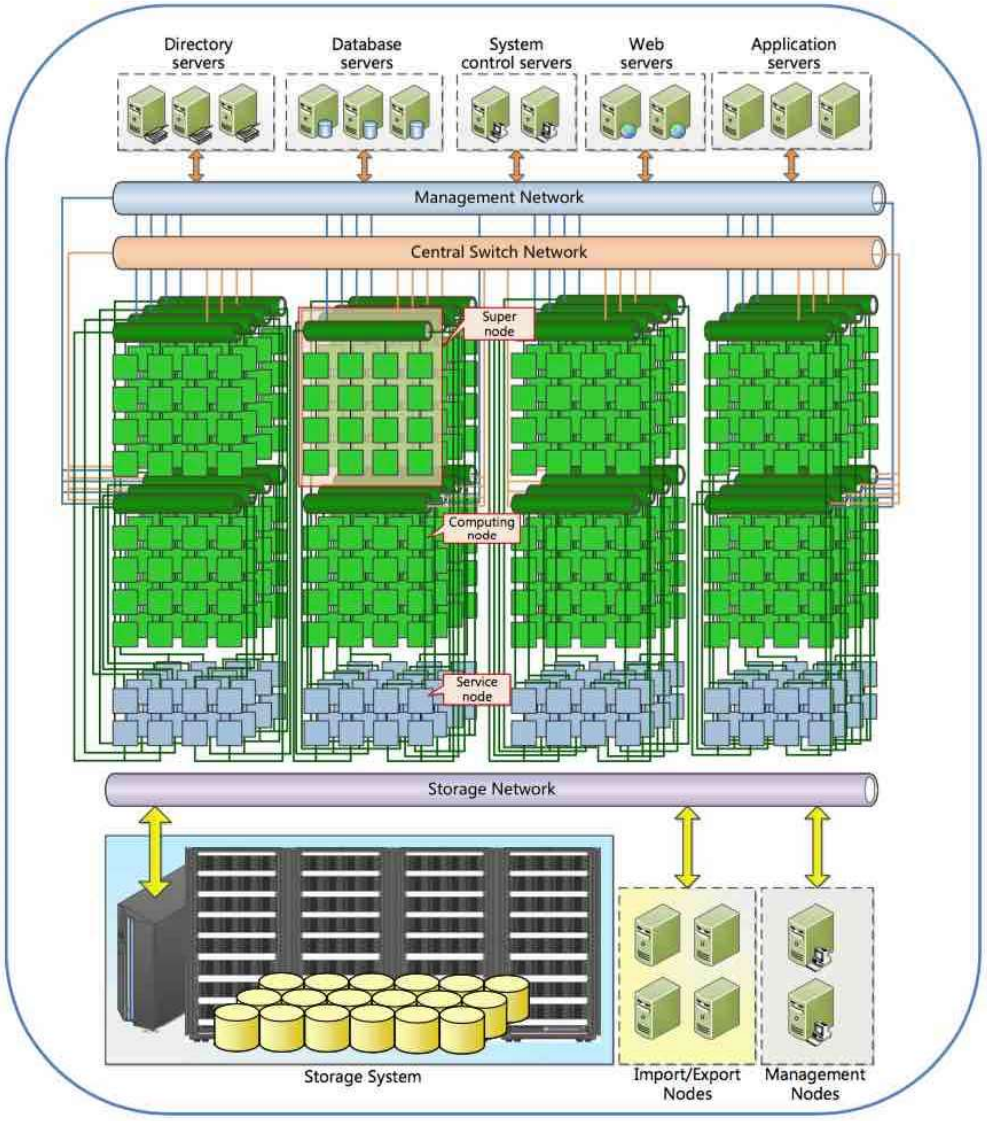
What's a Parallel Computer?



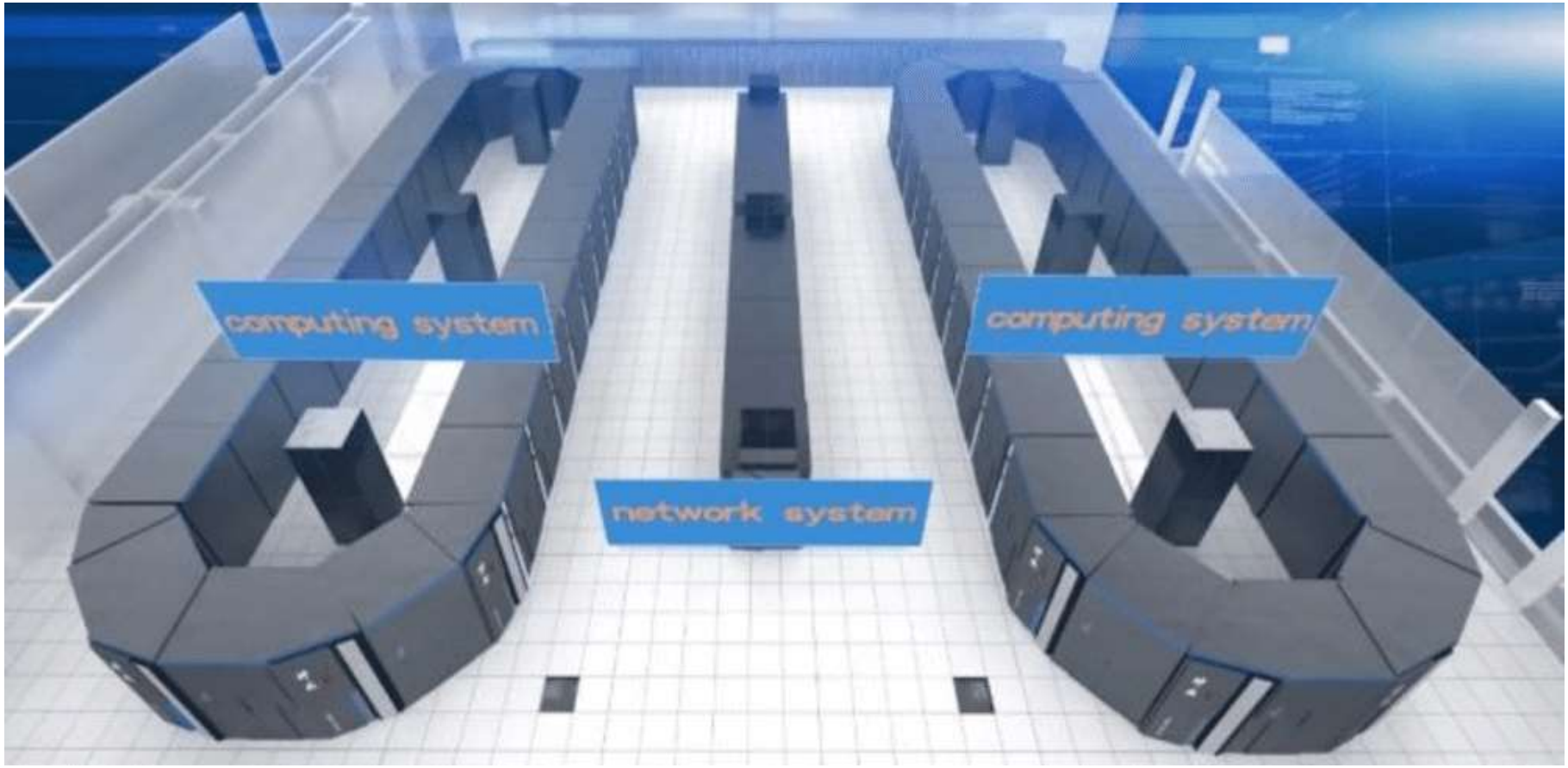
What's a Parallel Computer?



Whats a Parallel Computer?



Whats a Parallel Computer?



Conclusions

- Every computer is a parallel computer
- Parallel computers need independent work to run their many cores (or other resources) efficiently

- Themes of this class:
  - Identify available parallelism in programming tasks
  - Encode parallelism in parallel programs
  - Understand parallel hardware and how to optimize parallel performance