Concurrent and Distributed Systems

Talk One

Introduction to C&D Systems
and
Course Overview

Maxim Mozgovoy

What Is This Course About

Concurrent, parallel, and distributed computing.

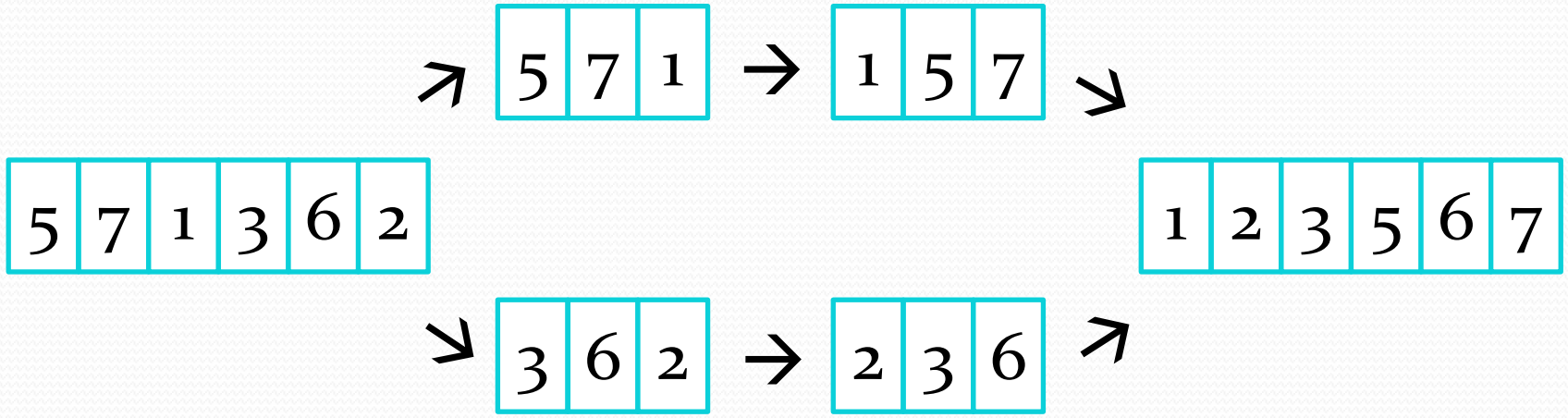
These concepts are related but still distinct:

- Parallel computing: a program divides a large problem into smaller subproblems to be solved simultaneously by multi-processor or multi-core hardware.
- Concurrent computing: a program is designed as a composition of independent processes, so it can use multiple processors, but still works on a single-processor machine.
- Distributed computing: a program consists of components located on networked computers that communicate by passing messages.

Examples: Parallel Computing

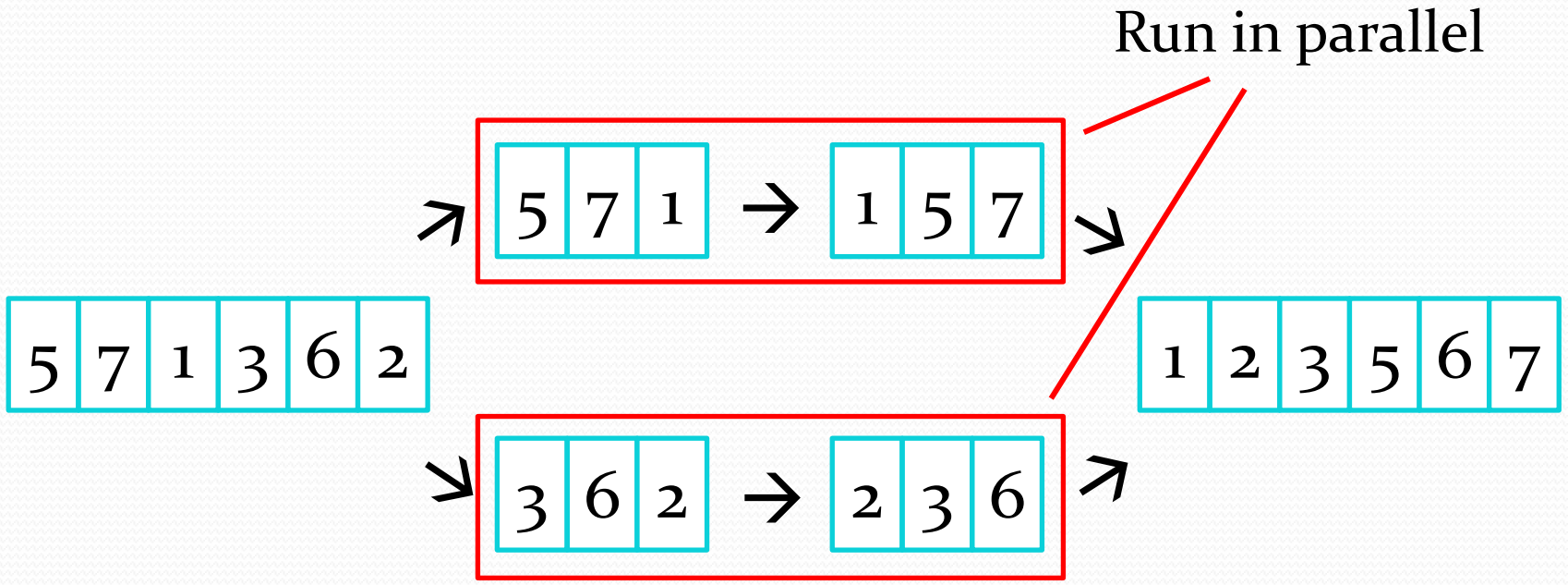
Parallel array sorting:

divide into subproblems and solve them



Examples: Parallel Computing

Parallel array sorting:



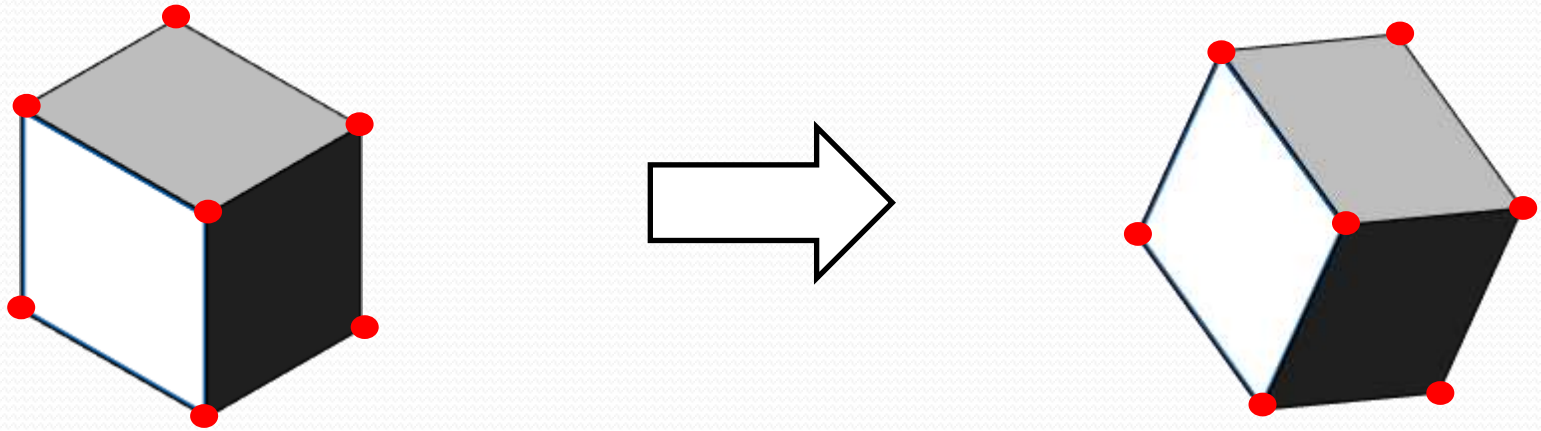
Challenges:

(1) Not always easy to divide the problem into subtasks;

(2) Not all parts of the algorithm can be run in parallel.

Examples: Parallel Computing

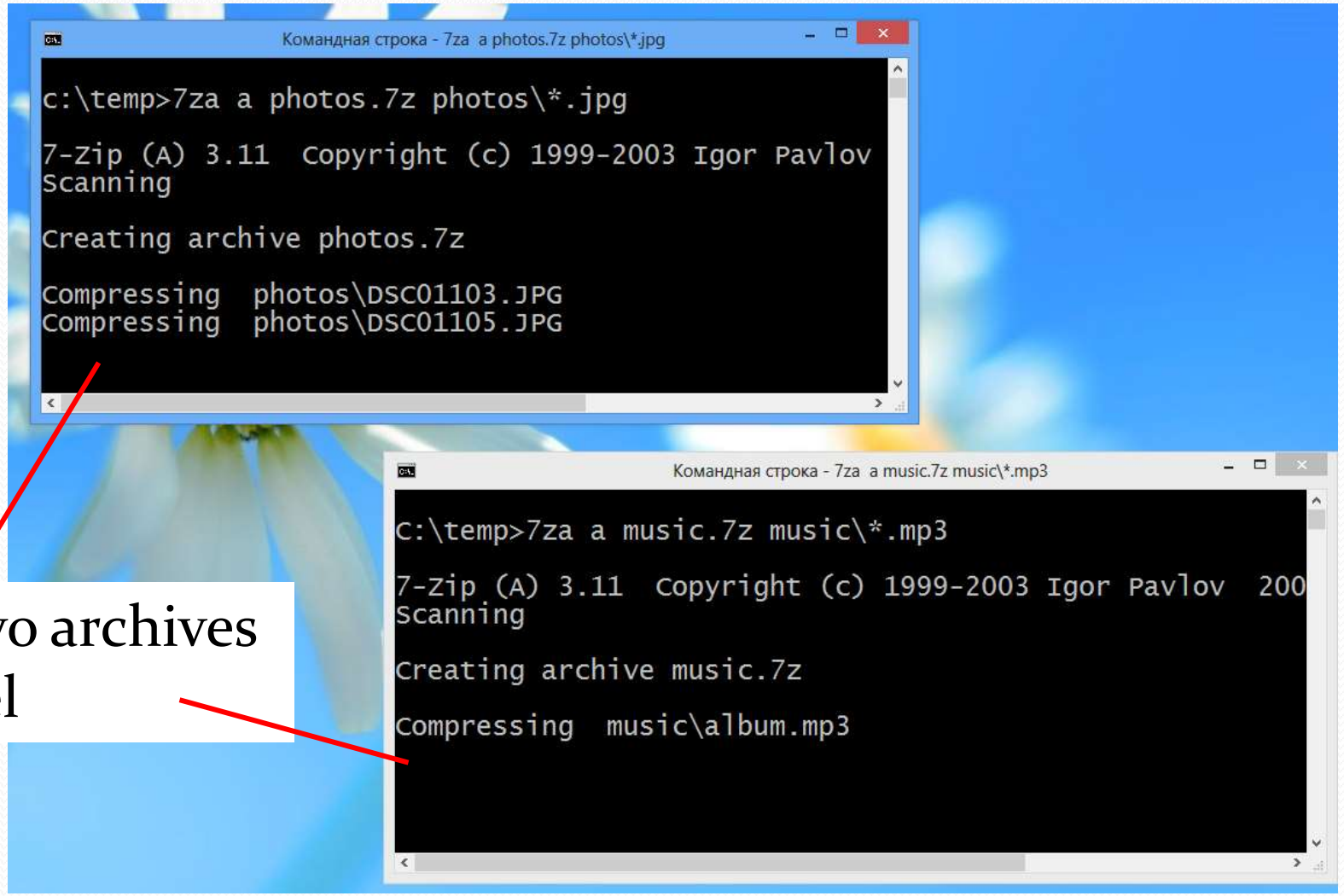
3D Graphic Manipulations:



Eight vertices have to be recalculated independently
(so we can do that in parallel very easily)

Examples: Concurrent Computing

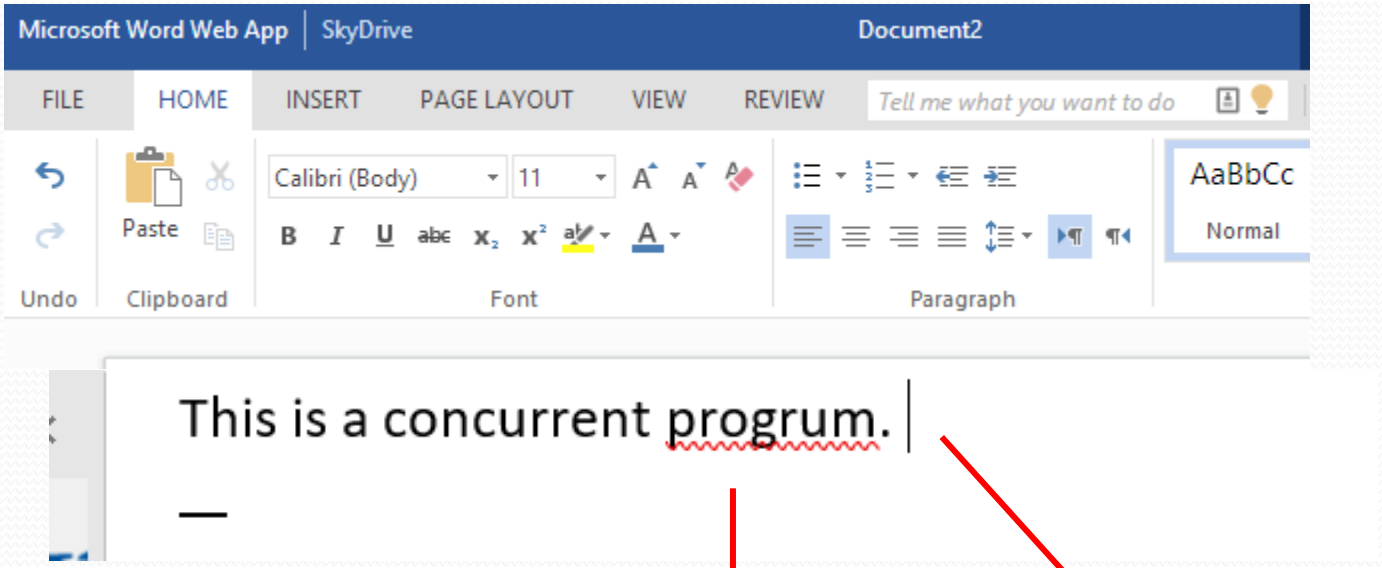
Multitasking Operating System (Windows, Linux, etc.)



Create two archives in parallel

Examples: Concurrent Computing

Responsive UI in applications and browsers



Another process runs spell checking.

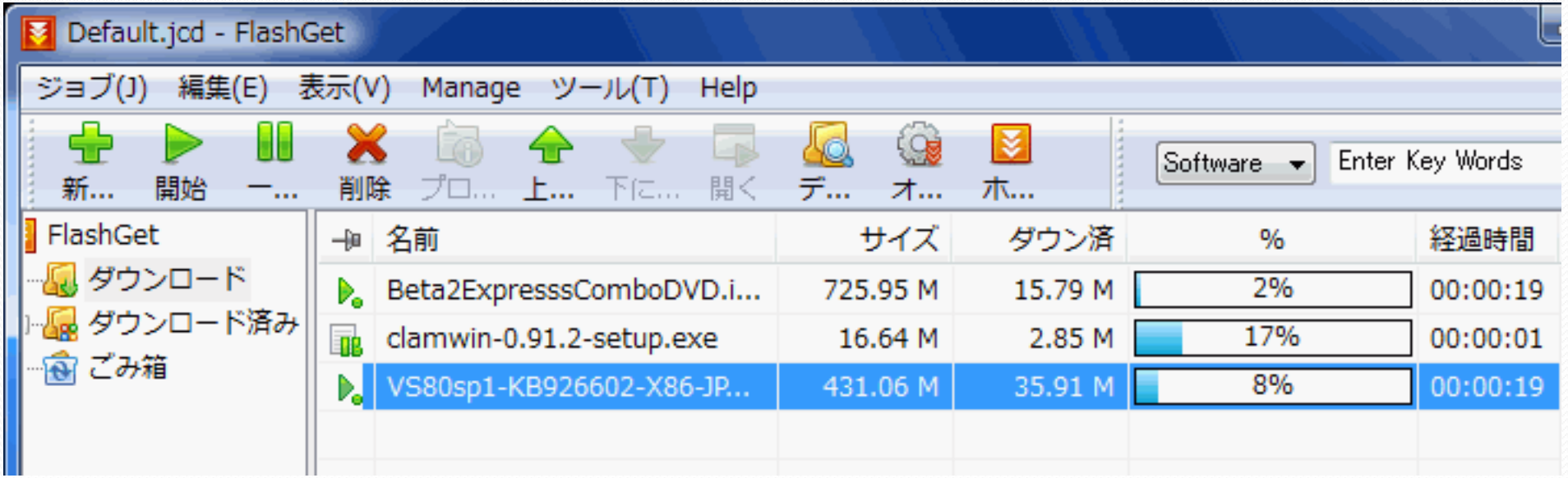
One process handles typing.

Examples: Concurrent Computing

Concurrent program may run faster even on single processor.

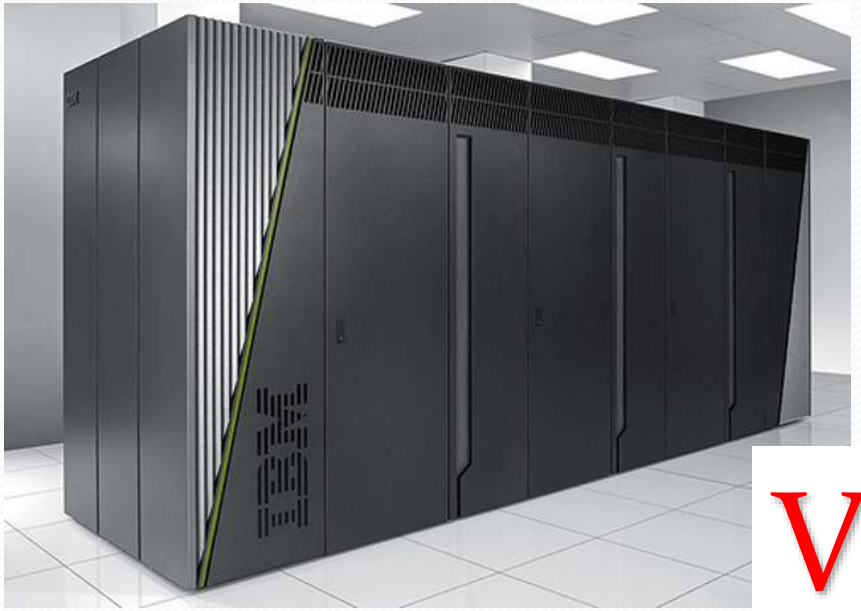
Problem: download several files from different web servers. Note that: (1) one CPU is fast enough to handle many parallel downloads; (2) some servers provide low download speeds.

Solution: parallel download! (helps because it is *I/O-bound*)



Examples: Distributed Computing

A cheaper substitute for hi-end parallel machines:



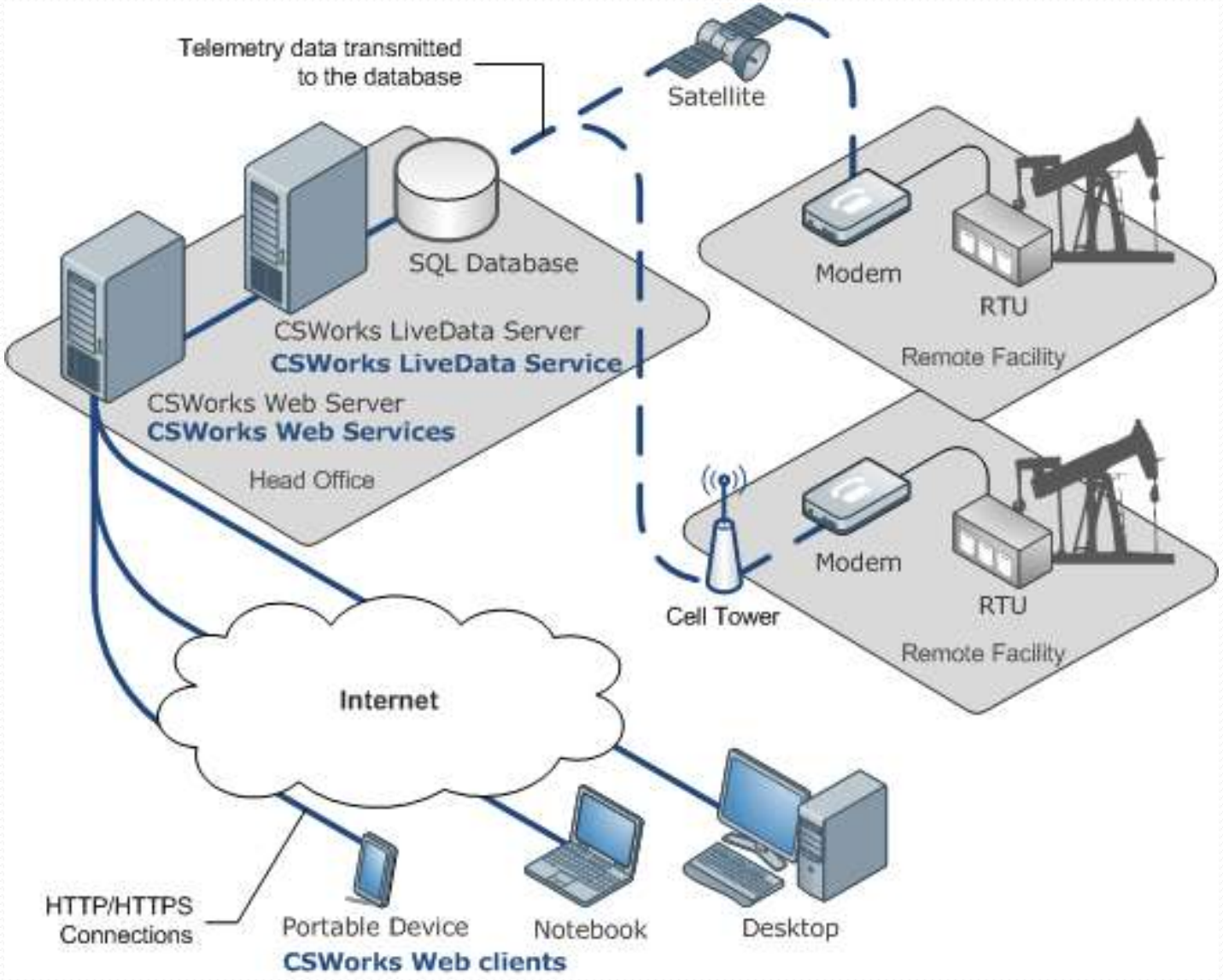
Supercomputer

VS

Networked computational cluster

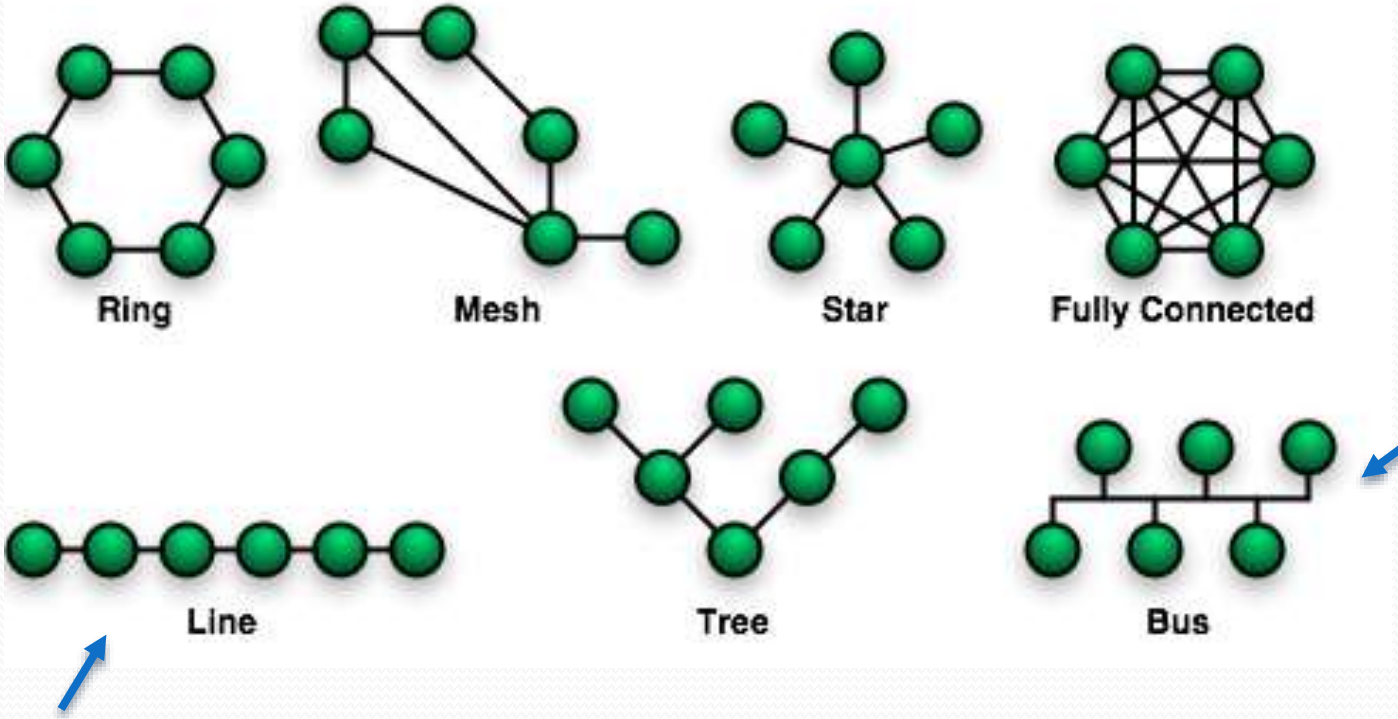
Examples: Distributed Computing

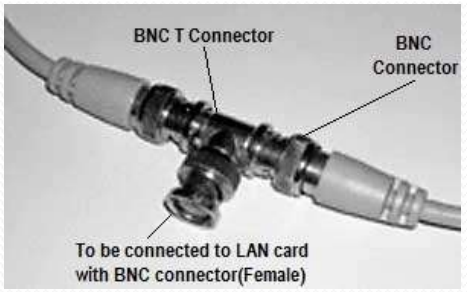
Organize programs and resources (also email, file sharing):



Network Topologies

For distributed systems, a network topology is important (different topologies are better for different tasks)





Coaxial table

Daisy chain (device to device connection)

For Internet applications, there is no way to control the topology (but possible for local Wi-Fi).

Difficulties

Synchronization

Parallel jobs should be coordinated to get correct results.

Builder A:
Build walls

Builder B:
Build roof

RESULT

Difficulties

Efficient work organization

All available hardware should do useful work most of time

A poorly designed concurrent system

Difficulties

Efficient work organization

However, ideal efficiency is very rarely achieved

× 1 year =

× ½ year =

(maybe)

365

× 1 day =

(very unlikely!)

Difficulties

In general, this is known as *Amdahl's law*:

T: total running time of the whole serial (non-parallel) program

B: total running time of a part that cannot be parallelized

F: fraction that must be executed serially: F = B / T

N: number of processors

$$Speedup \leq \frac{1}{F + \frac{1 - F}{N}}$$

Ex: suppose you have to do 10% of the computation *not* in parallel, and you have 5 processors. Then your speedup is at most 1 / (0.1 + (1 – 0.1) / 5) = 3.57 (times).

With 100 processors it is 1 / (0.1 + (1 – 0.1) / 100) = 9.1 (times), etc.

With $N = \infty$, Amdahl's formula converges to 1 / *F*, so this is the theoretical limit.

Difficulties

Network-related issues in distributed systems:

- Worse reliability;
- Additional design decisions in case of manually-built clusters;
- Lower communication speed;
- Scalability challenges.

Why You Should Study It

- ...-1990: C&D systems are not for ordinary people (banks, companies, universities, military)
- 1990-2000: Everyone has a multitasking desktop!
- 2000-2010: Everyone has access to a network!
- 2010-...: Everyone has a multiprocessor machine!

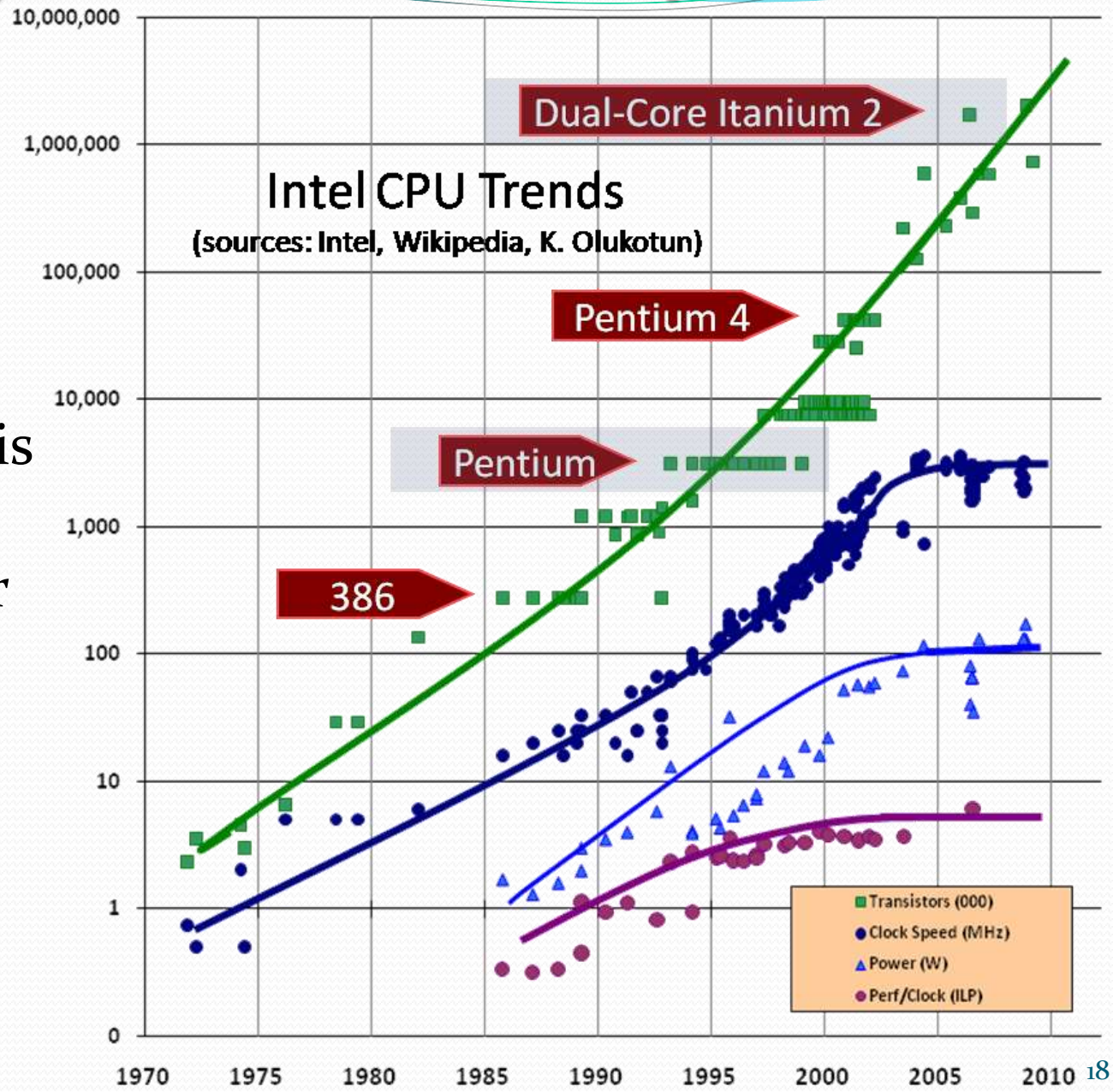
The chances that you will deal with C&D systems are very high.

Why You Should Study It

"The Free Lunch is Over" – *H. Sutter*

If the performance is not enough, you cannot just wait for better equipment (as it was possible before early 2000s).

You have to use concurrency!



Course Outline

During the course, we will consider the following topics:

- Basics of concurrent programming.
- Shared-memory and message-passing models.
- Formal verification by model checking.
- Distributed programming in computational clusters.
- Various architectures of distributed programs.
- Client-server programming.
- Distributed objects architecture.
- (And more…)

- You'll also have to write code!
  This course is mostly about practical programming.

Prerequisites

- Basic algorithms and data structures.

  (how to sort an array, what is binary tree…)
- Java programming.

  (again, quite basic level)

  Why Java: it works similarly on all major platforms and has many tools for concurrent & distributed programming.

Study Process

- This year I will continue experiments with flipped learning: most lectures will be available online!
- Watching lecture videos will be your homework, and in the classroom we will solve exercises.
- All information is available in our course Moodle.

- Mon, per. 3-4 (M10): discuss exercise solutions and future plans.
- Mon, per. 5-6 (std5): solve exercises.
- Thu, per. 5-6 (std5): solve exercises.

- Homework: watch video lecture & do the quiz.

Grading Scheme

- Exercises (40% of the final score).
  Exercises should be done before the next class!
- Midterm and final exams (40% of the final score).
- Lecture quizzes (20% of the final score).

- Scoring around 50% of points in each category
  should be sufficient to pass the course.

Note: the course is difficult because concurrent programming is difficult. There is nothing we can do about it.

Exercises show what kind of problems you will encounter in real-life concurrent programming.

Cooperation & LLM Policies

- Our goals are:
  - Learn various aspects of concurrent and distributed systems.
  - Ensure fair grading.

- "Learning" and "grading" are separate processes:
  - When learning, you can consult books, friends, LLMs.
  - Note that LLMs hallucinate a lot, so always double check them.
  - Exercises are, however, strictly individual.
  - Your solution must be made by you, not your friend or LLM.
  - When consulting friends or LLMs, note that only you are responsible for the solutions you submit, and you must perfectly understand everything there.
  - Remember you won't have any external help on exams.