Distributed Systems Design
COMP 6231

Introduction

Lecture 1

Essam Mansour


INTRODUCTIONS

Course Staff

Instructor: Dr. Essam Mansour

- Email: essam.mansour@concordia.ca
- Instructor’s Website: emansour.com
- Office hours: by appointment

You should arrange for a meeting via email

Teaching Assistant:

- Omij Mangukiya omij.mangukiya@mail.concordia.ca
- Waleed Afandi waleed.afandi@mail.concordia.ca

Essam

Omij

Waleed

Teaching Style

- I like interaction in class
- I like to ask questions
- I like to be asked questions

- I like to know (and memorize) your names ☺

- I like to give practical assignments and projects

- I like to learn …

Course Textbook

Textbook:

Distributed Systems, ***3rd ed***, by Tannenbaum and Van Steen, Prentice Hall *2018*

- All lectures will be prepared from this book.

- The book is ***available at the moodle***.
- plus some other material, we may need.

Course Outline

- Introduction (today Part 1)
  - What, why, why not?
  - Basics
- Distributed Architectures
- Interprocess Communication
  - RPCs, RMI, message- and stream-oriented communication
- Processes and their scheduling
  - Thread/process scheduling, code/process migration, virtualization
- Naming and location management
  - Entities, addresses, access points

Course Outline

- Canonical problems and solutions
  - Mutual exclusion, leader election, clock synchronization, …
- Resource sharing, replication and consistency
  - DFS, consistency issues, caching and replication
- Fault-tolerance
- Security in distributed Systems
- Distributed middleware
- Advanced topics: web, cloud computing, green computing, big data, multimedia, and mobile systems

Course Grading

Type	#	Weight
Project	1	30%
Exams	2	45%
Assignments	5	25%

Table 1: Breakdown of the main activities involved in the course.

Course Grading

Project: **30%** of your final score.

- Each team consists of 3 to 4 students.
- Each team will ***learn*** one of the following systems:
  - Pregel, GraphLab, PowerGraph, Cassandra, Couchbase, MongoDB, or Elasticsearch. You can propose a parallel system to develop.
  - ***Discuss the concepts of distributed systems*** in the chosen system.
  - **Use a real dataset of at least one GB, the larger the better,** and
  - ***Demo*** the capabilities of the chosen system.

- The deliverables of the project are:

Demo	**15%** of the final grade
Presentation	**10%** of the final grade
Report	**5%** of the final grade

Course Grading

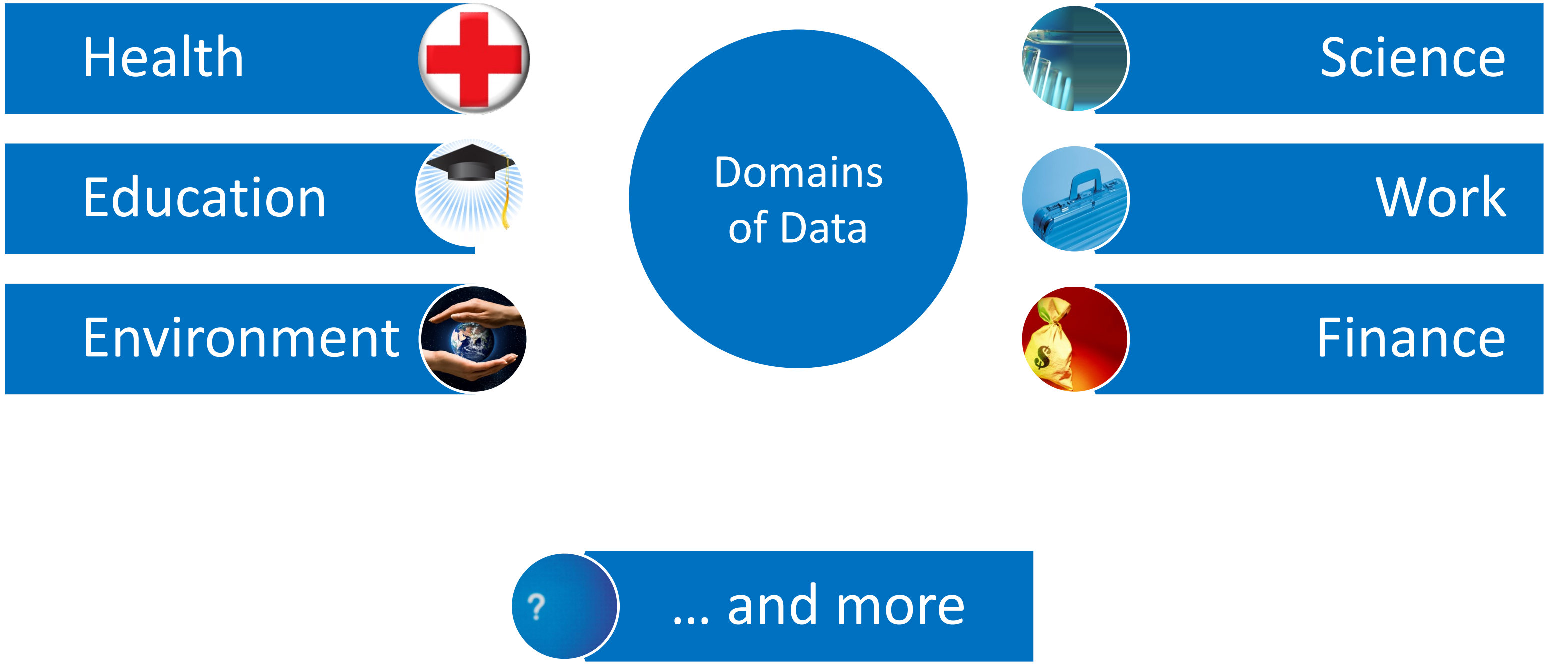
Assignments:

- 25% towards your final score
- 4 programming assignments

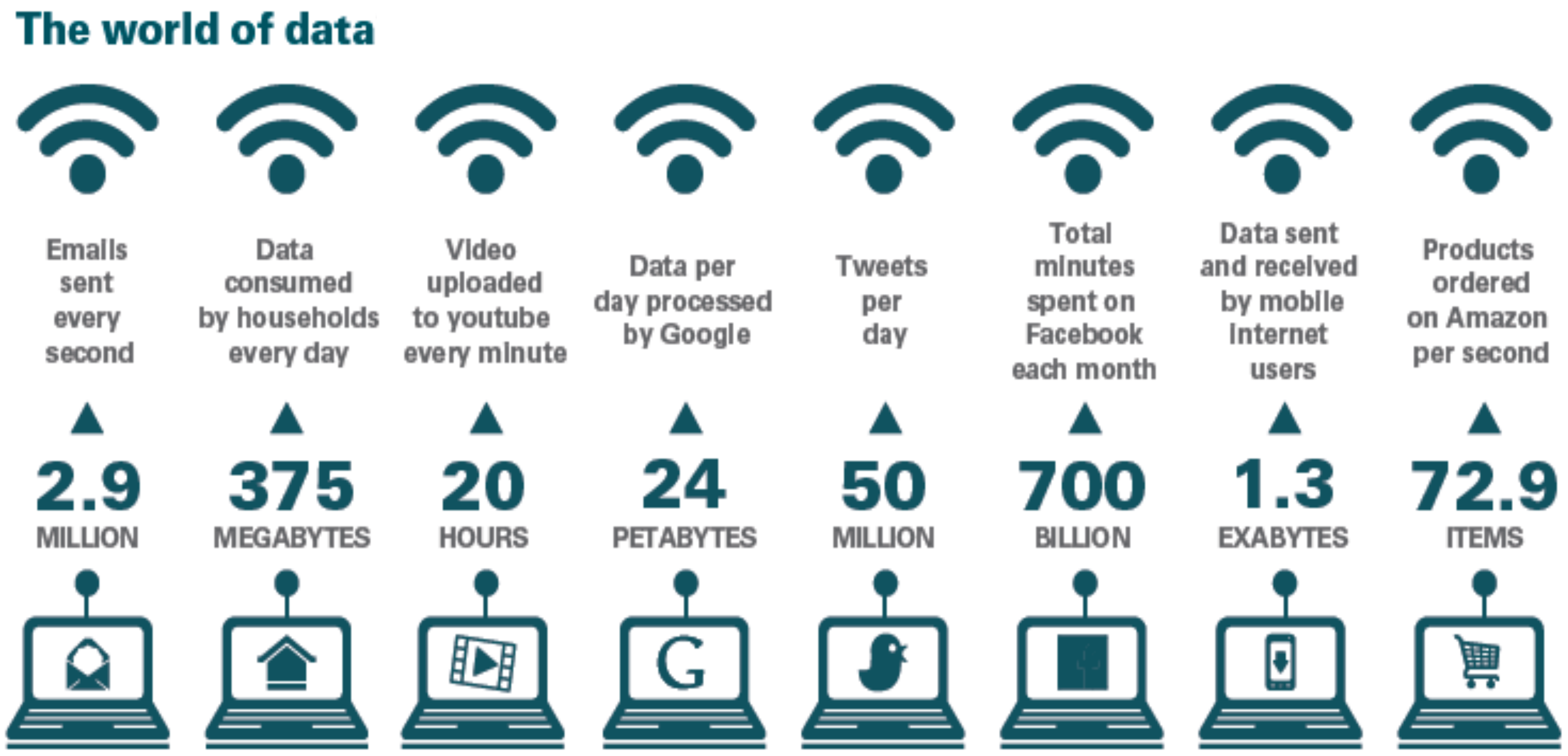
ID	Topic	Weight	Release Date	Due Date
1	Socket	5%	Fri Sept 22	Sun Oct 08
2	*Multithreading & MPI*	*9%*	*Fri Oct 06*	*Sun Oct 29*
3	Docker	5%	Fri Oct 27	Sun Nov 12
4	SPARK Map-Reduce	6%	Fri Nov 10	Sun Nov 26


… and now
It's Your Turn!

Data Becoming Critical to Our Lives

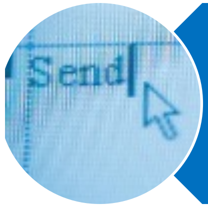


We Live in a World of Data…



What Do We Do With Data?


Store



Share


Access


Process


Encrypt


…. and more!

We want to do these seamlessly...

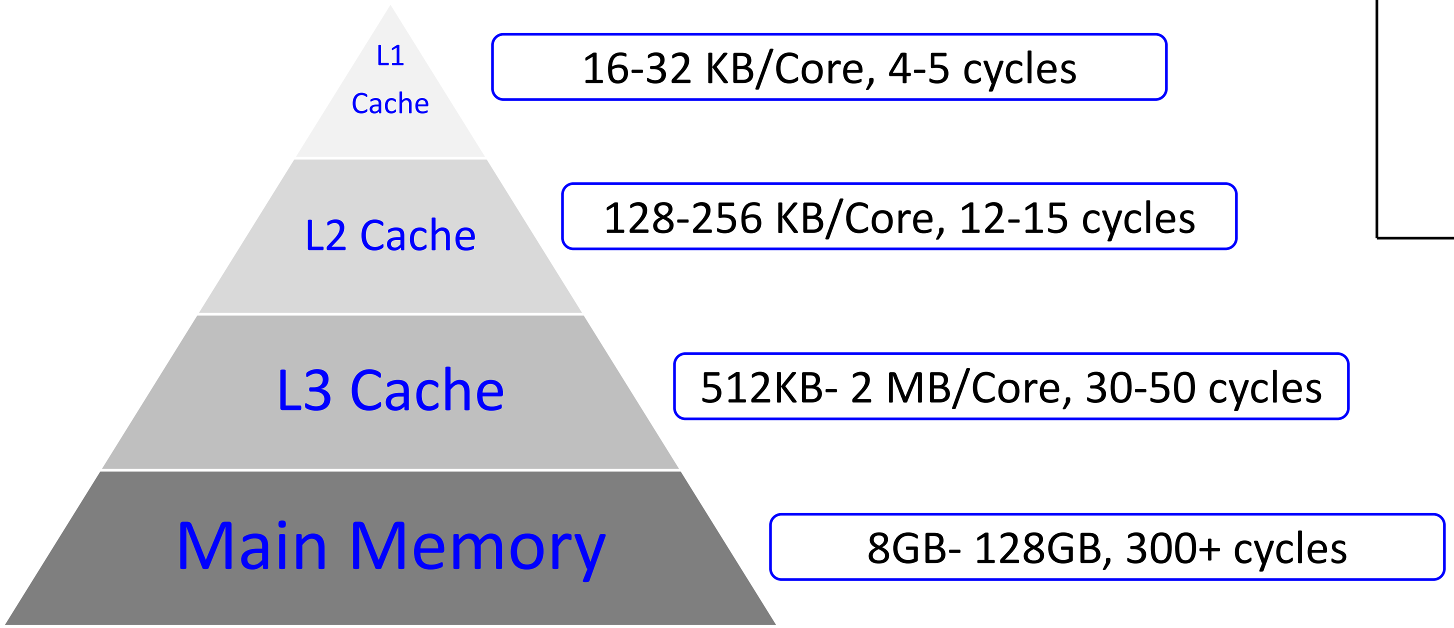
How to Store and Process Data at Scale?

- A system can be scaled:
  - Either *vertically* (or up) ✓
    - Can be achieved by hardware upgrades (e.g., faster CPU, more memory, and/or larger disk)
  - And/Or *horizontally* (or *out*)
    - Can be achieved by adding more machines

Vertical Scaling

- Caveat: Individual computers can still suffer from *limited resources* with respect to the scale of today's problems

1. Caches and Memory:

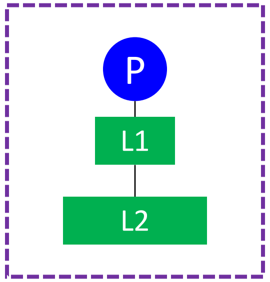


The image part with relationship ID rId8 was not found in the file.

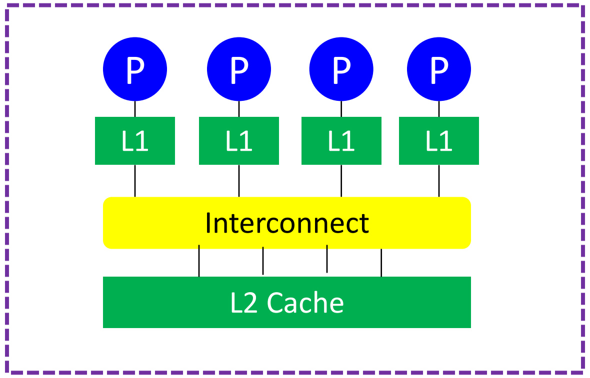
Vertical Scaling

- Caveat: Individual computers can still suffer from *limited resources* with respect to the scale of today's problems

2. Processors:
   - Moore's law still holds
   - Chip Multiprocessors (CMPs) are now available



A single Processor Chip

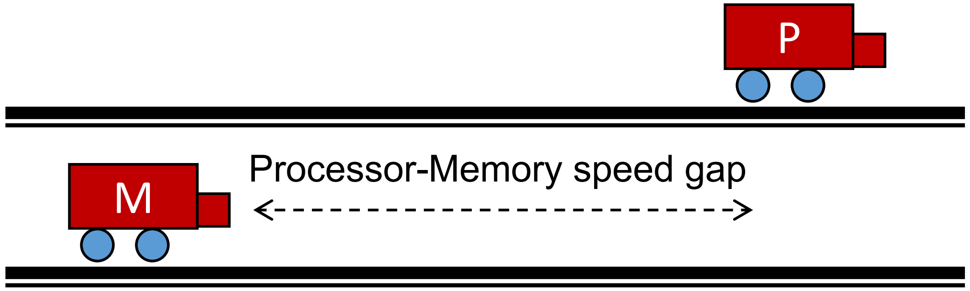


A CMP

Vertical Scaling

- Caveat: Individual computers can still suffer from *limited resources* with respect to the scale of today's problems

2. Processors:
   - But up until a few years ago, CPU speed grew at the rate of 55% annually, while the memory speed grew at the rate of only 7%



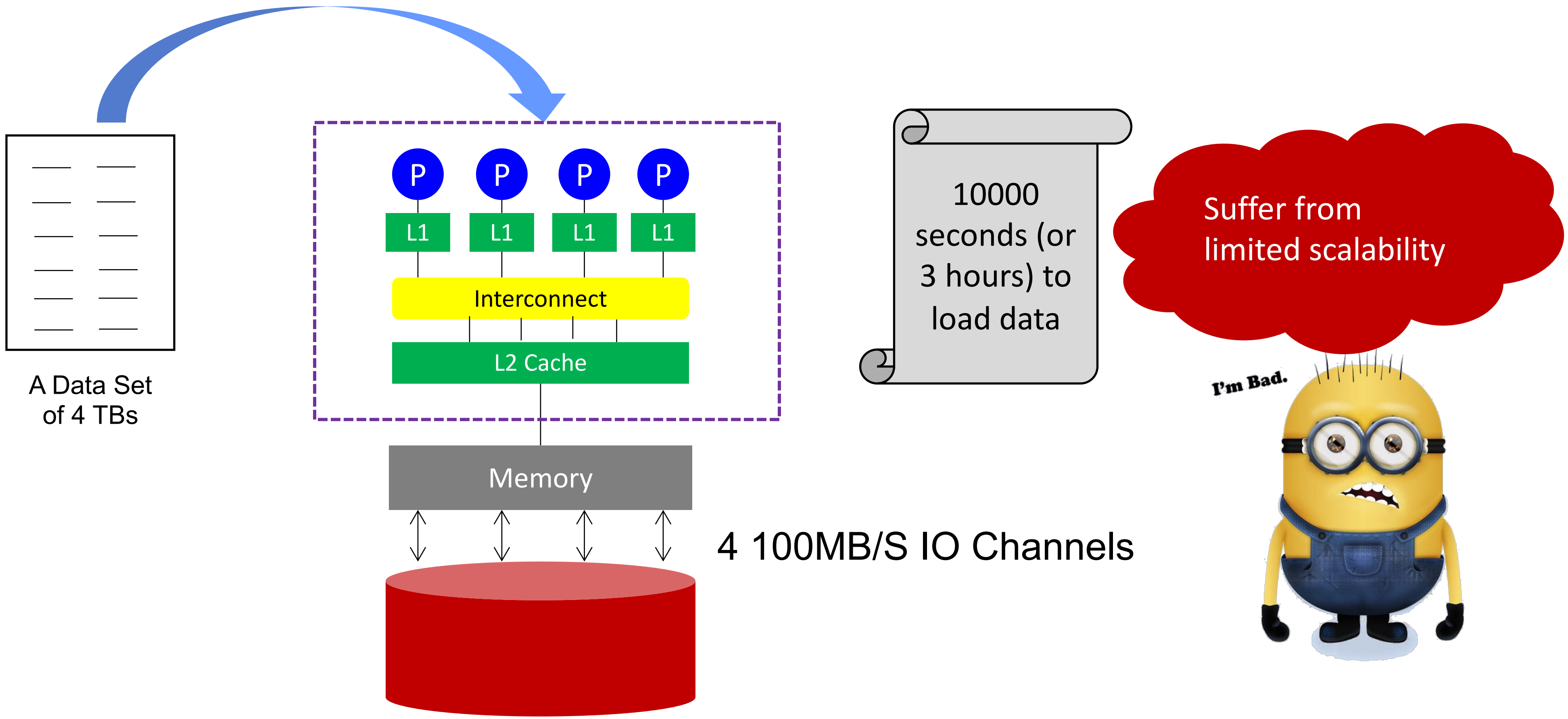
Vertical Scaling

- Caveat: Individual computers can still suffer from *limited resources* with respect to the scale of today's problems

2. Processors:
    - But up until a few years ago, CPU speed grew at the rate of 55% annually, while the memory speed grew at the rate of only 7%

    - Even if 100s or 1000s of cores are placed on a CMP, it is a challenge to deliver input data to these cores fast enough for processing

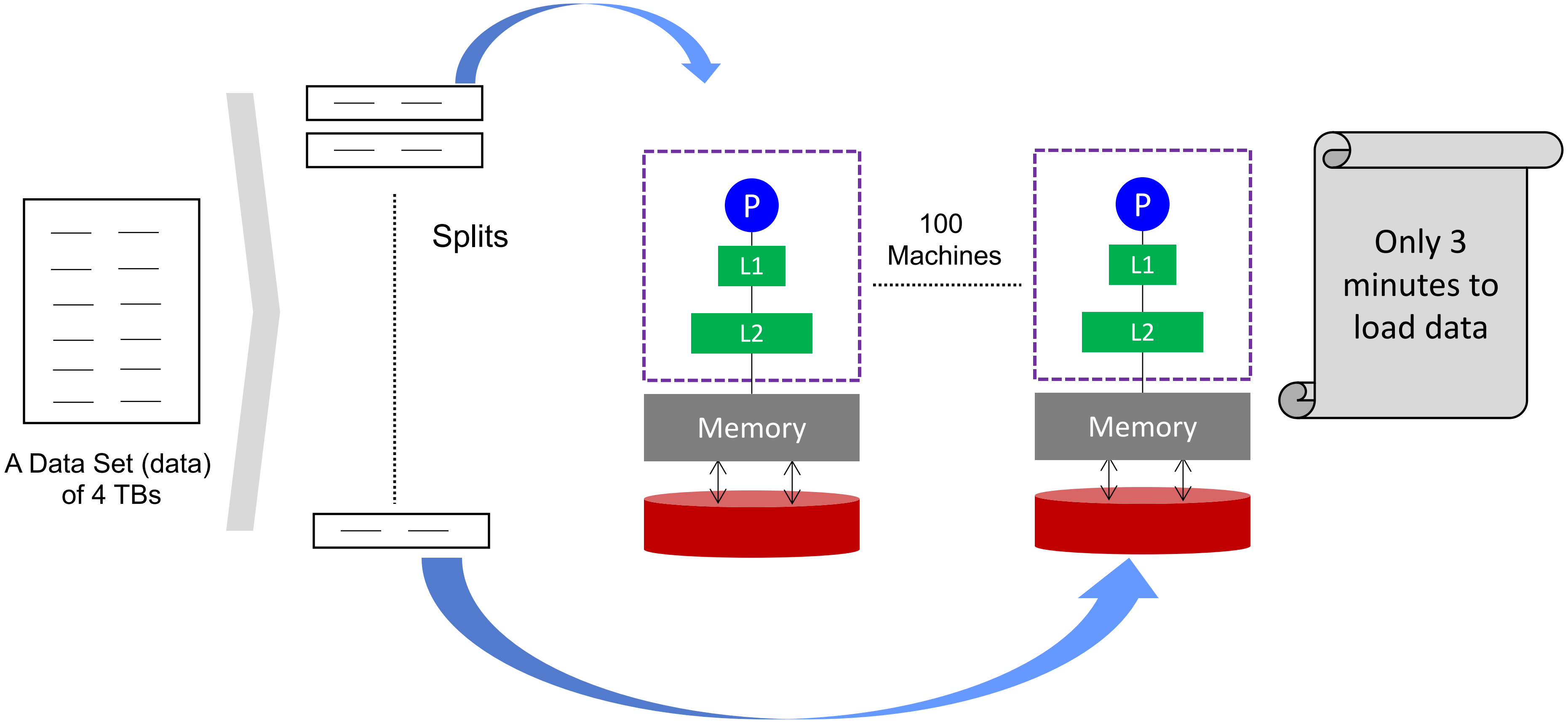
Vertical Scaling



How to Store and Process Data at Scale?

- A system can be scaled:
  - Either *vertically* (or up)
    - Can be achieved by hardware upgrades (e.g., faster CPU, more memory, and/or larger disk)
  - And/Or *horizontally* (or *out*) ✓
    - Can be achieved by adding more machines

Horizontal Scaling



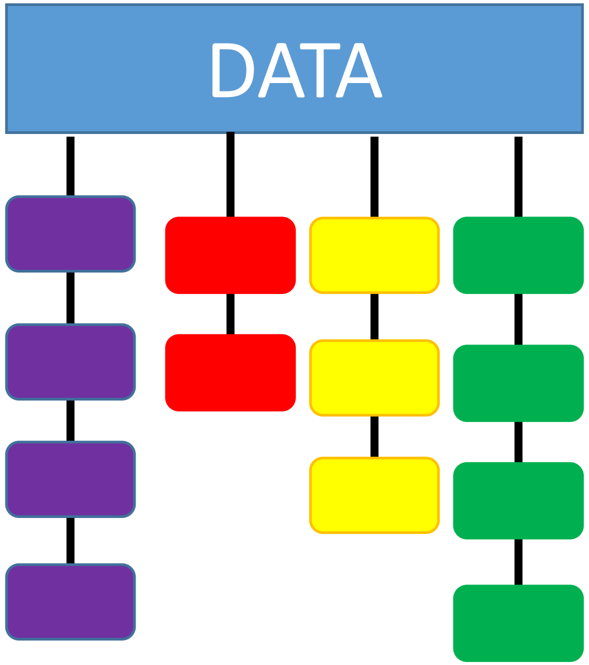
Requirements

- But, this necessitates:
  - A way to express the problem in terms of parallel processes and execute them on different machines (*Programming and Concurrency Models*)
  - A way to organize processes (*Architectures*)
  - A way for distributed processes to exchange information (*Communication Paradigms*)
  - A way to locate and share resources (*Naming Protocols*)

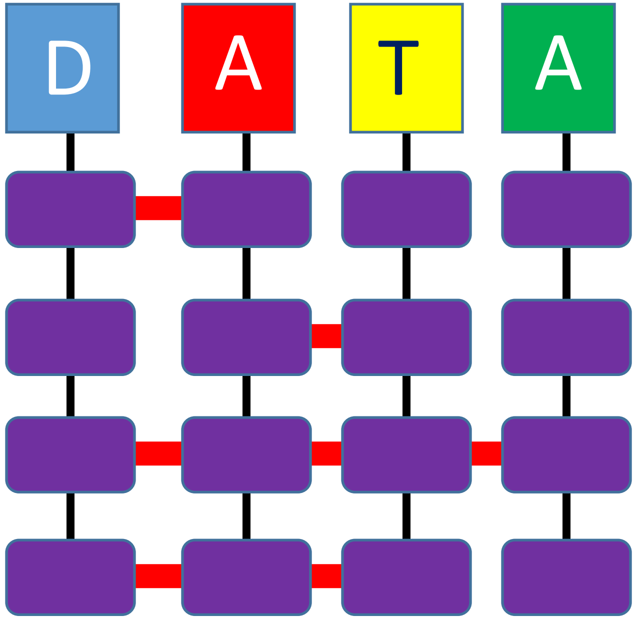
Requirements

- But, this necessitates:
  - A way for distributed processes to cooperate, synchronize with one another, and agree on shared values (*Synchronization*)
  - A way to reduce latency, enhance reliability, and improve performance (*Caching, Replication, and Consistency*)
  - A way to enhance load scalability, reduce diversity across heterogeneous systems, and provide a high degree of portability and flexibility (*Virtualization*)
  - A way to recover from partial failures (*Fault Tolerance*)

Degree of Parallelism



Task Parallelism



Data Parallelism

So, What is a Distributed System?

A distributed system is:

A collection of independent computers that appear to its users as a single coherent system

One in which components located at networked computers communicate and coordinate their actions only by passing messages

Features

- Distributed Systems imply *four* main features:

1. Geographical Separation
2. No Common Physical Clock
3. No Common Physical Memory
4. Autonomy and Heterogeneity

Parallel vs. Distributed Systems

- Distributed systems contrast with parallel systems, which entail:

1. Strong Coupling
2. A Common Physical Clock
3. A Shared Physical Memory
4. Homogeneity

I am not sure.
Can you verify that?

Another Definition of a Distributed System

A distributed system:

- Multiple connected CPUs working together
- A collection of independent computers that appears to its users as a single coherent system

Examples: **parallel machines**, **networked machines**

Distributed System Models

- Cluster computing systems / Data centers
  - LAN with **a cluster of servers** + storage
    - Linux, Mosix, ..
    - Used by distributed web servers, scientific applications, enterprise applications
- Grid computing systems
  - Cluster of machines connected over a WAN
  - SETI @ home
- WAN-based clusters / distributed data centers
  - Google, Amazon, Facebook …
- Virtualization and data center
- Cloud Computing

Emerging Models

- Distributed **Pervasive Systems**
  - "smaller" nodes with networking capabilities
    - Computing is "everywhere"

Computers

Mobile Devices

Consumer Electronics

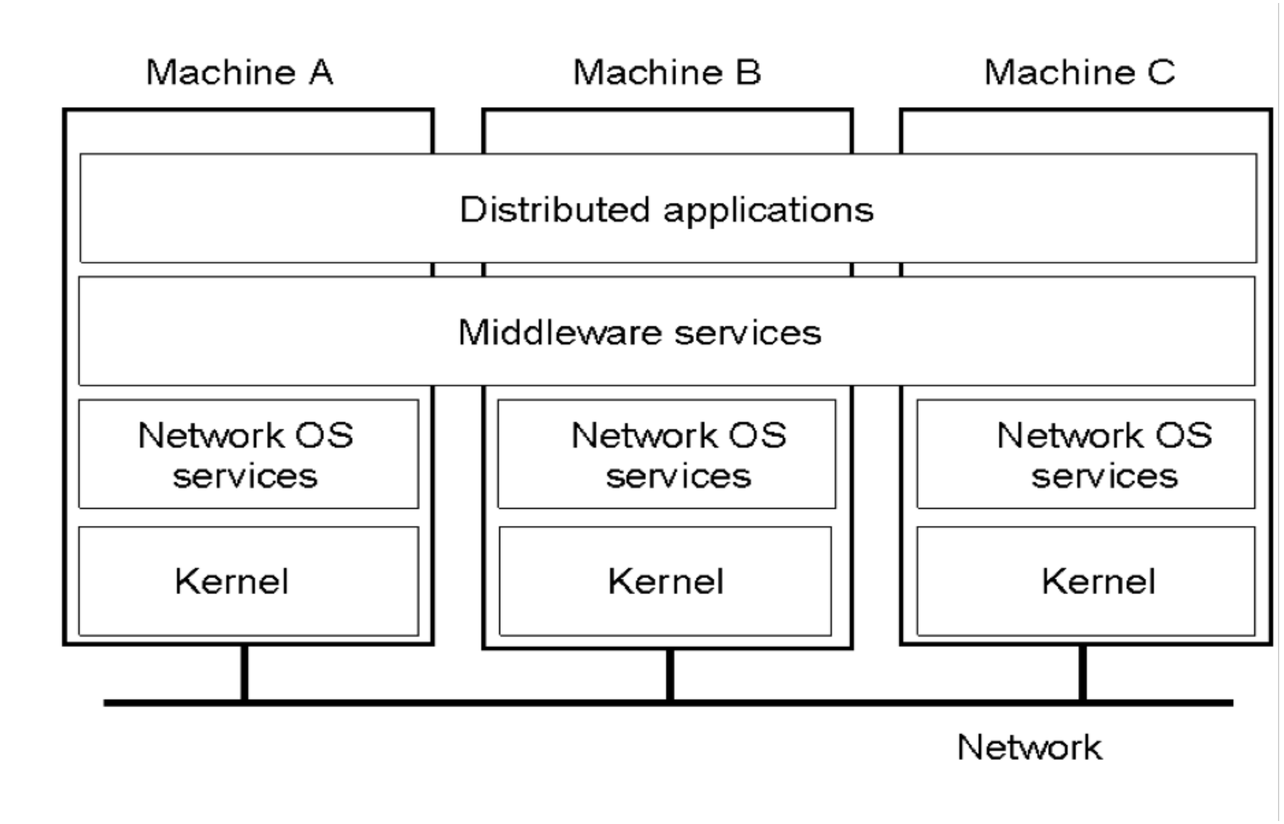
Personal Monitors and Sensors

…and even appliances

We also want to access, share and process our data from all of our devices, **anytime, anywhere**!

Middleware-based Systems

General structure of a distributed system as middleware.



Transparency in a Distributed System

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

Scalable Distributed System

Scalability Dimensions

- Size scalability:
    - A system can be scalable with respect to its size,
    - we can easily add more users and resources to the system without any noticeable loss of performance.

- Geographical scalability:
    - The users and resources may lie far apart,
    - The fact that communication delays may be significant is hardly noticed.

- Administrative Scalability:
    - Still be easily managed even if it spans many independent administrative organizations.

Scaling Techniques

- *Principles* for good decentralized algorithms
  - No machine has complete state
  - Make decision based on local information
  - A single failure does not bring down the system
  - No global clock

- *Techniques*
  - Asynchronous communication
  - Distribution
  - Caching and replication

Comparison between Systems

Item	Distributed OS		Network OS	Middleware-based OS
	Multiproc.	**Multicomp.**		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Depends on OS	Depends on OS	Open	Open

A To-Do List

- Read Chapters 1, Introduction

- Attend the lab

- Start working in your first assignment

Next Lecture

- Chapter 2: Architectures

Questions?