Introduction

ECE 454/751: Distributed Computing

Instructor: Dr. Wojciech Golab
wgolab@uwaterloo.ca

Slides are derived from M. Van Steen and A. S. Tanenbaum, Distributed Systems, 3rd Edition, Pearson, 2017.

What is a distributed system?

Textbook: A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.

Wikipedia: A distributed system is a software system in which components located on networked computers communicate and coordinate their actions by passing messages.

Leslie Lamport (2013 Turing Award recipient): A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

Maarten van Steen: Distributed systems are like 3D brain teasers: easy to disassemble; hard to put together.

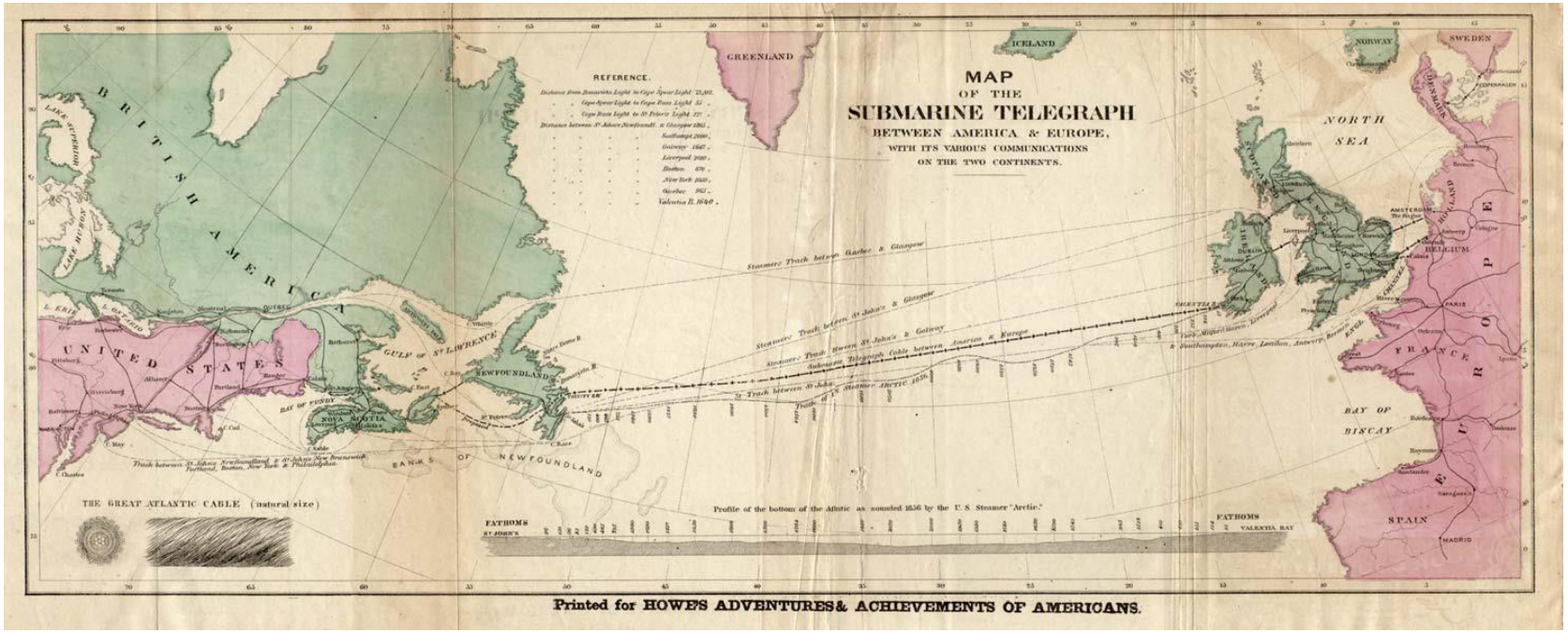
DNS

Why build distributed systems?

1. **Resource sharing saves money.**
   Example: office staff share a printer and file server.
2. **Integrating multiple systems into one can simplify business processes.**
   Example: payroll system talks to accounting system.
3. **A centralized system may not be powerful or dependable enough to solve a given problem.**
   Example: Google Web Search too big for one server.
4. **In some scenarios the users themselves are mobile and distributed around the world.**
   Example: social networking.

Anecdote: William Thomson (Lord Kelvin) vs. Edward Orange Wildman Whitehouse

大西洋电缆

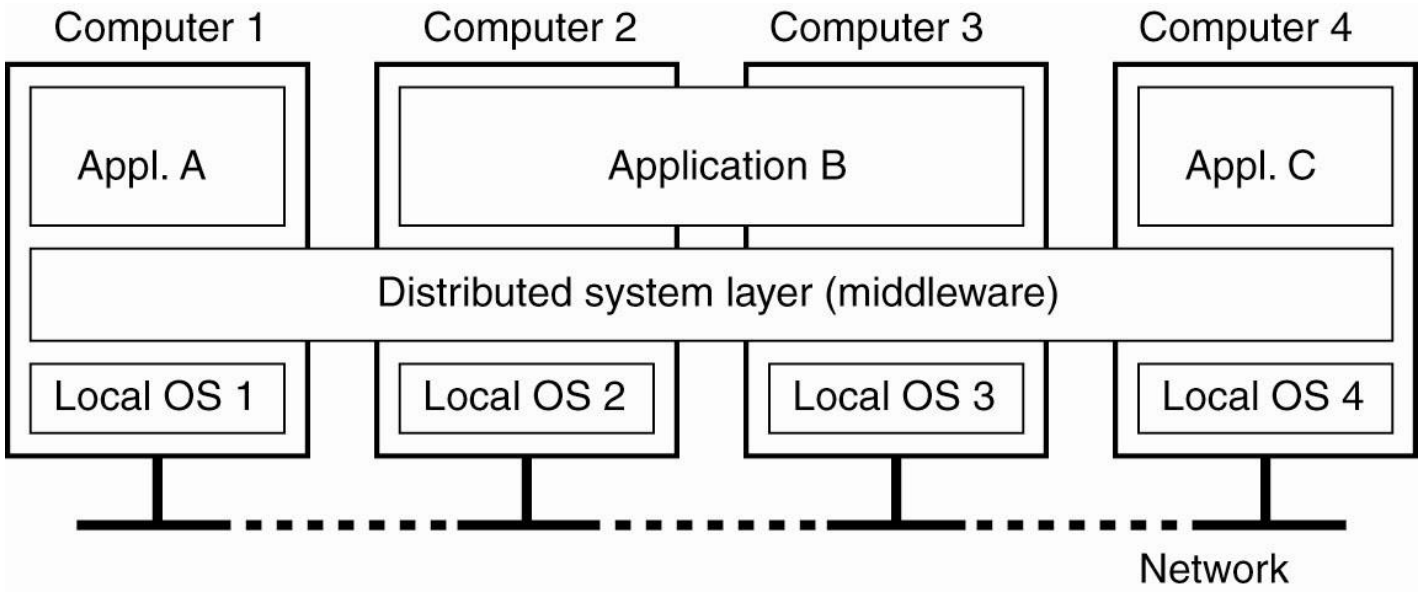


Sources: https://en.wikipedia.org/wiki/Transatlantic_telegraph_cable

https://atlantic-cable.com/Maps/index.htm (Bill Burns)

Middleware

In order to support heterogeneous computers and networks while offering a **single-system view**, distributed systems are often organized as middleware: a layer of software that separates applications from the underlying platforms.



Common middleware services

中间件有不同功能

- communication
  (e.g., add job to remote queue)
- transactions
  (e.g., access two independent services atomically)
- service composition
  (e.g., Google map enhanced with weather forecast)
- reliability
  (e.g., replicated state machine)

Goals of distributed systems

- supporting resource sharing
- making distribution transparent
- being open
- being scalable

Supporting resource sharing

Resources can include:

- peripheral devices (e.g., printers, video cameras)
- storage facilities (e.g., file server)
- enterprise data (e.g., contact info, payroll)
- Web pages (e.g., Web search)
- CPUs (e.g., supercomputer)

Making distribution transparent

Distributed systems, particularly middleware systems, attempt to provide **distribution transparency**. That is, they hide the fact that processes and resources are physically distributed.

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 is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Being open

An **open distributed system** offers components that can be easily used by or integrated into other systems.

Openness arises from the following properties:

- interoperability
- composability
- extensibility
- separation of policy from mechanism
  (e.g., configurable parameters in browsers and e-mail clients)

Being scalable

工作负载加大，性能保持住

Scalability is a system's ability to expand along three axes:

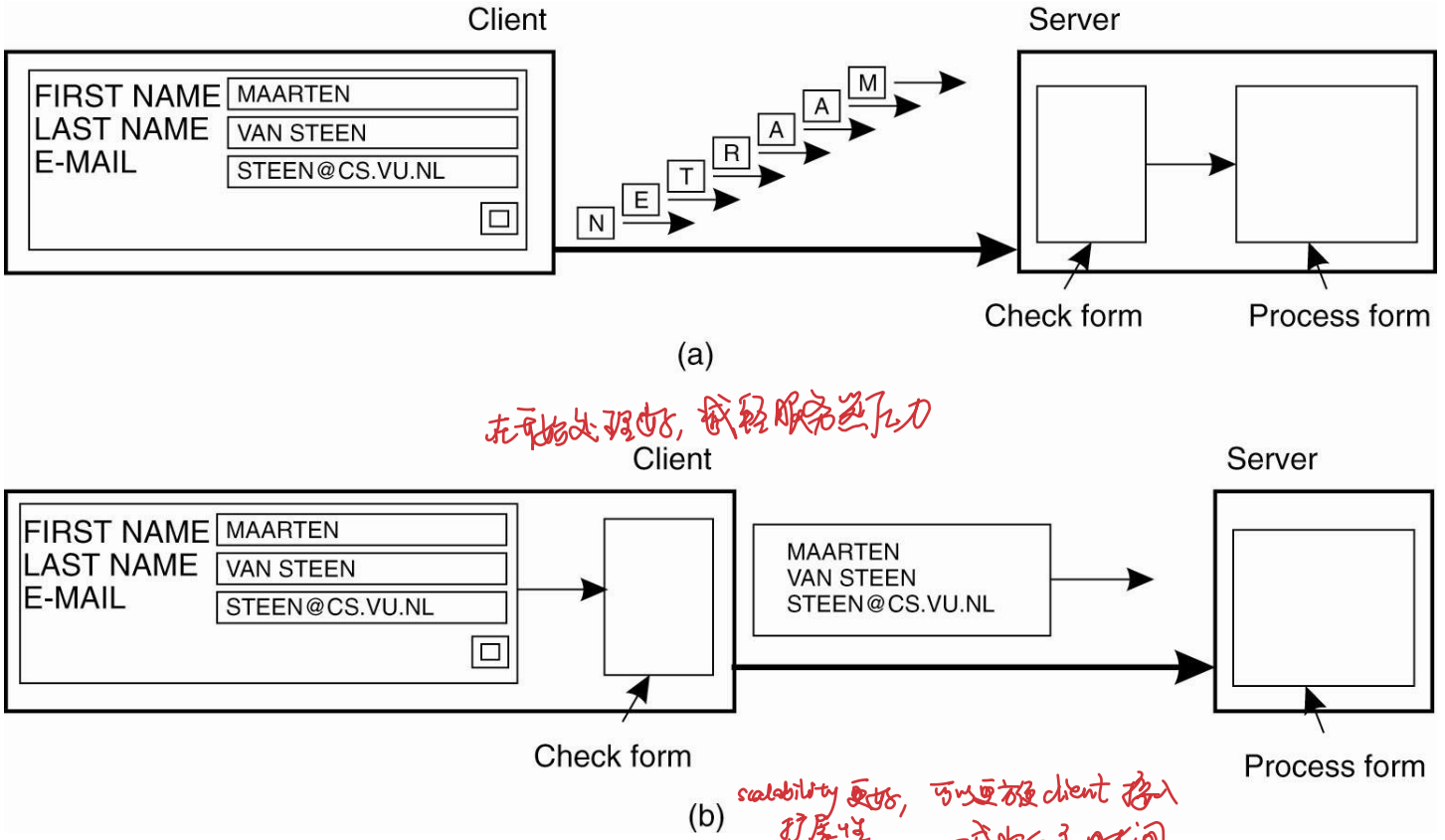
1. size (e.g., adding users and resources)
2. geography (e.g., users on different continents)
3. administration (e.g., multiple independent admins)

Simplifying design concepts tend to limit scalability:

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

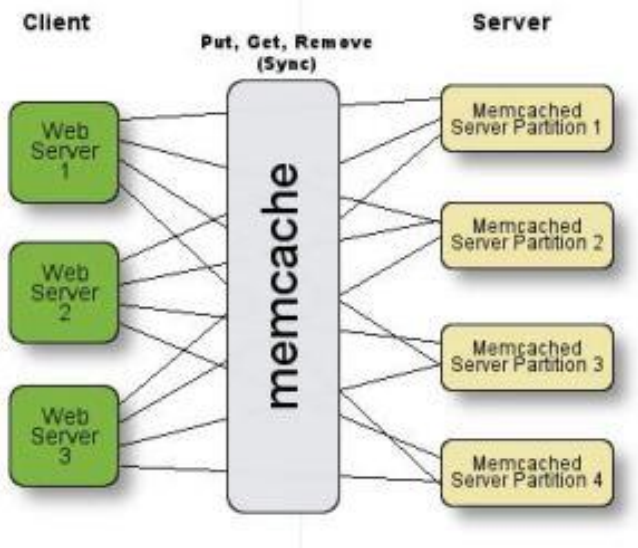
Scaling techniques: hiding communication latencies

Example: validating web form at server vs. at client.



Scaling techniques: replication

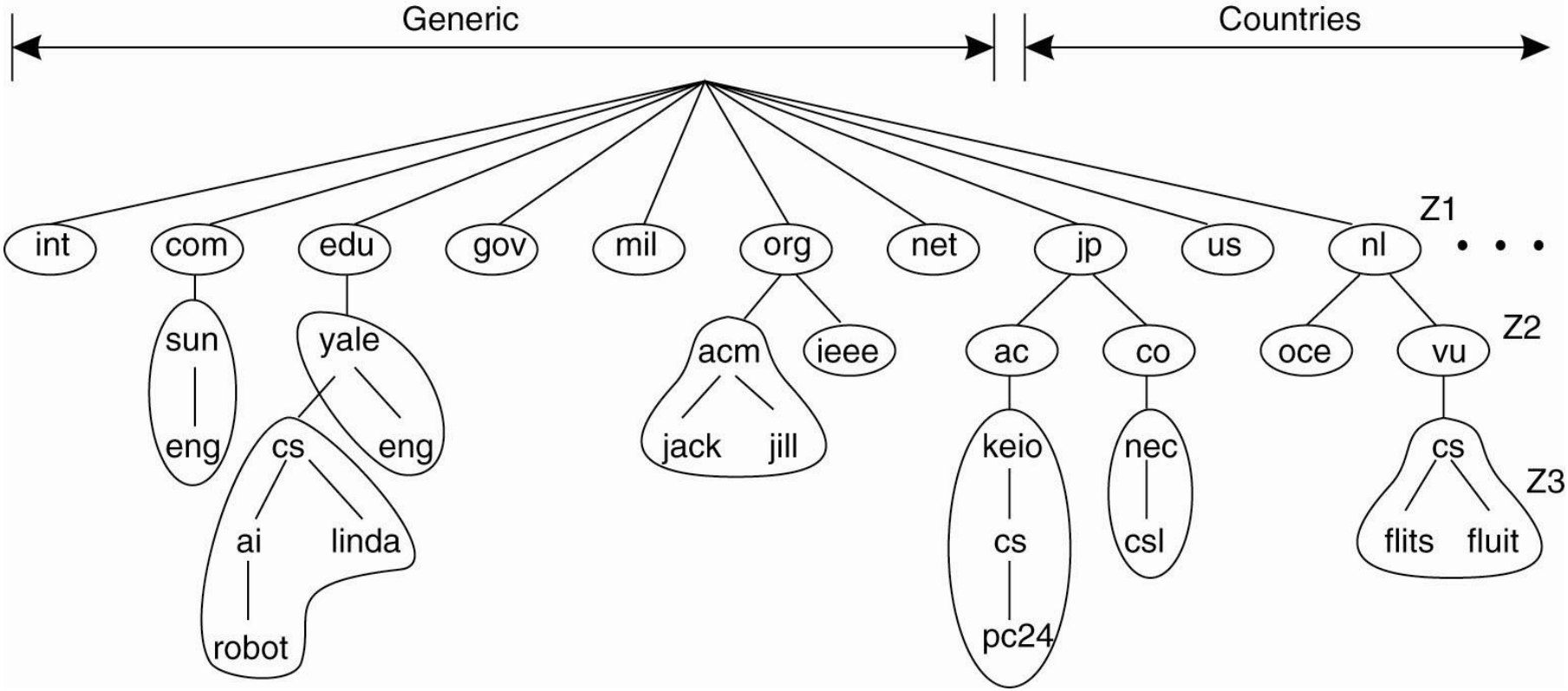
Example: using distributed memory cache to speed up web applications.



Source: https://linuxtechme.wordpress.com/2012/03/29/470/ (Renjith Raju)

Scaling techniques: partitioning

Example: original DNS name space was divided into zones.



Fallacies (pitfalls) of networked and distributed computing

1. The network is reliable.
2. The network is secure.
3. The network is homogeneous.
4. The topology does not change.
5. Latency is zero.
6. Bandwidth is infinite.
7. Transport cost is zero.
8. There is one administrator.

Credits: Bill Joy, Tom Lyon, Peter Deutsch, James Gosling.

Types of distributed systems

Distributed systems in the wild form a diverse ecosystem that includes the following species:

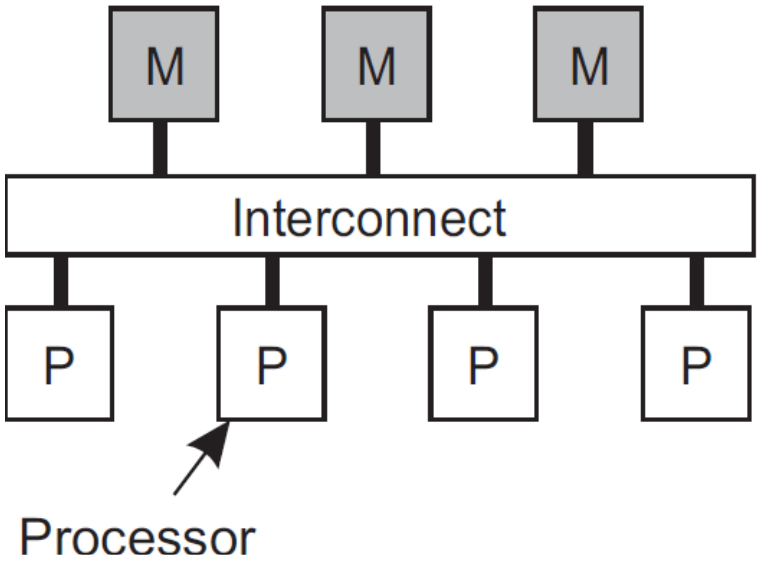
- Websites and Web services (e.g., D2L, YouTube)
- high performance computing (HPC)
- cluster computing (e.g., Hadoop, Spark)
- cloud and grid computing
- transaction processing
- enterprise application integration (EAI)
- distributed pervasive systems / Internet of things (IoT)
- sensor networks

High performance computing

两种不同模式.

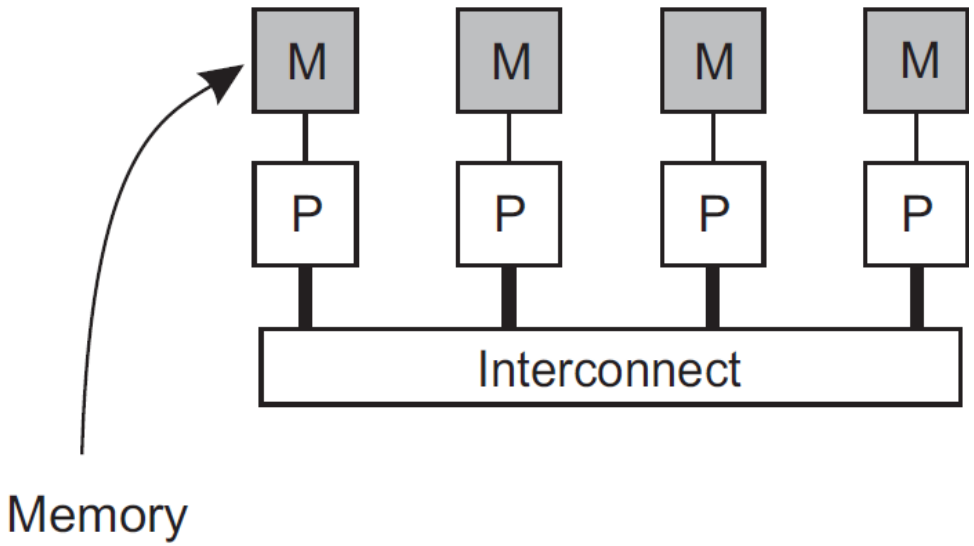
low level

Shared memory



higher level

Private memory



multiprocessor
(shared memory)

multicomputer
(message passing)

Shared memory vs. message passing paradigms

Shared memory paradigm:

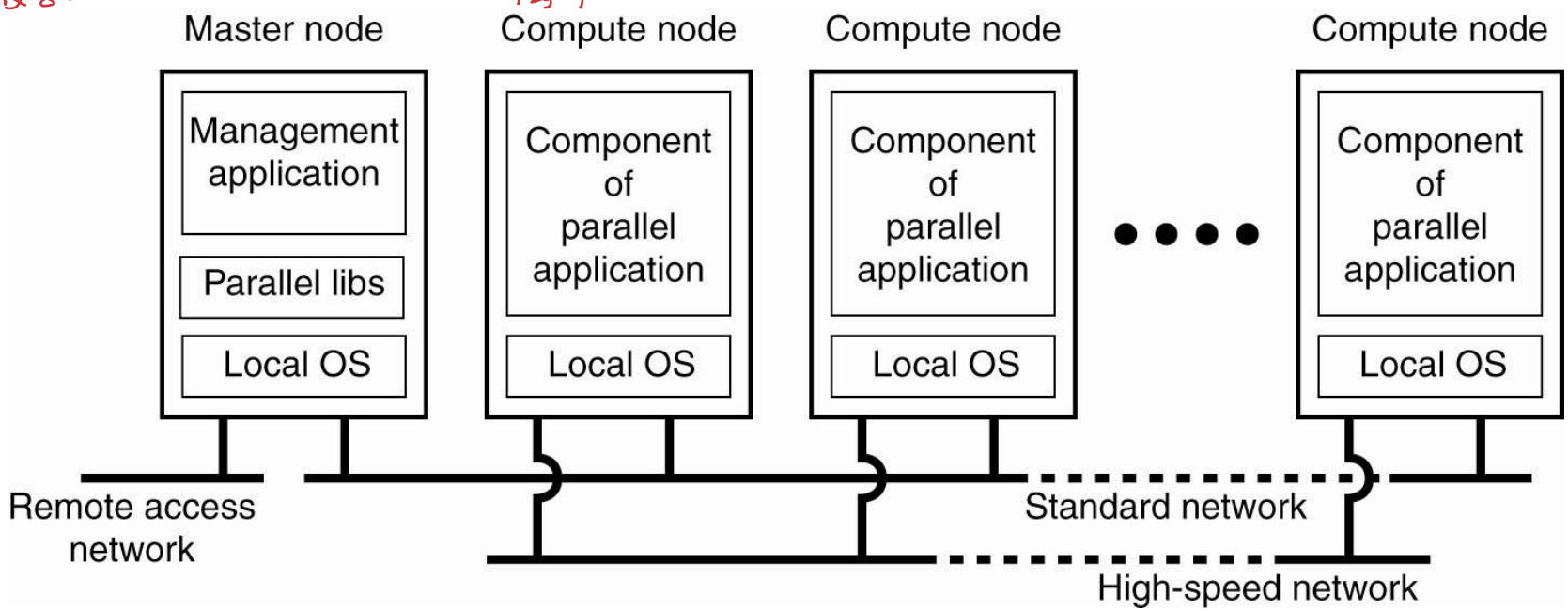
- threads communicate by accessing shared variables
  (easier to program but requires shared variable abstraction)
- used heavily for solving CPU-intensive problems
- practitioners tend to call this "parallel computing"

Message passing paradigm:

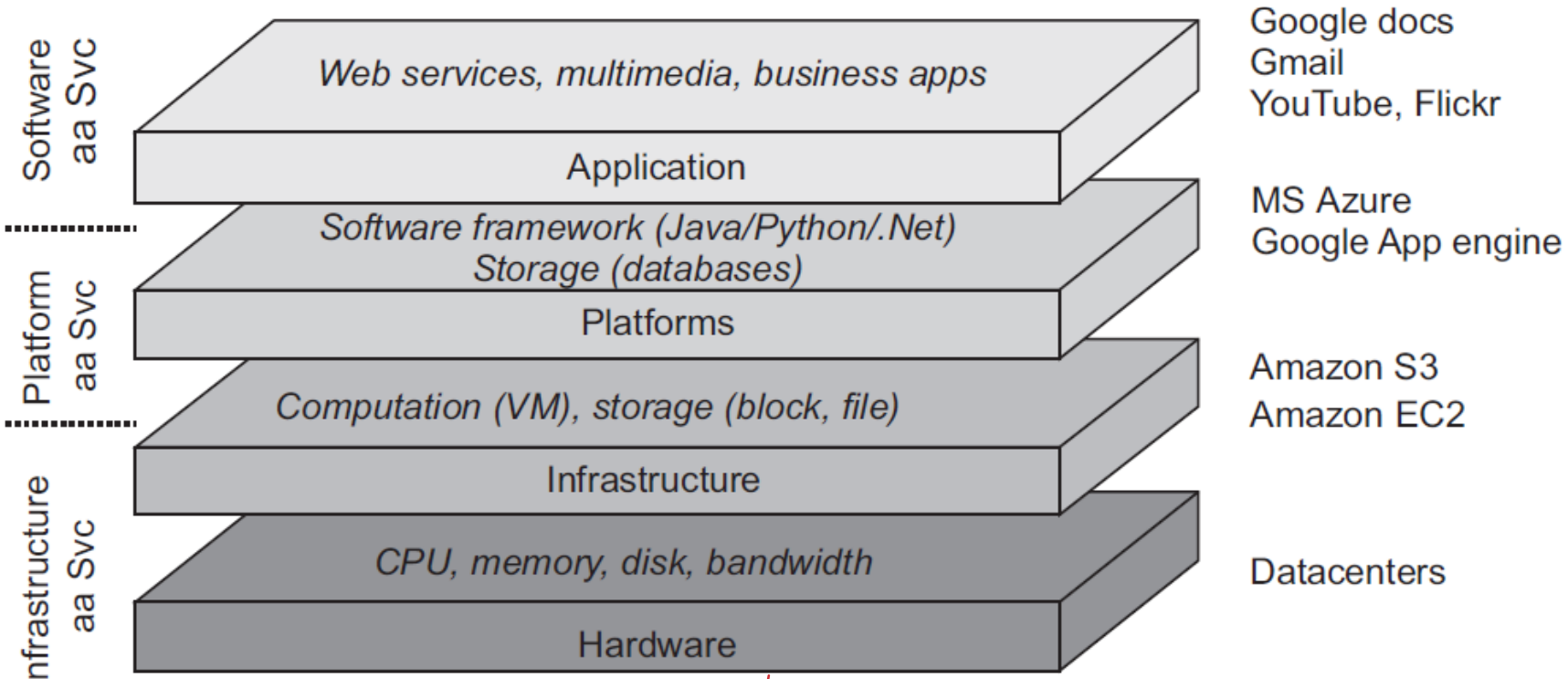
- processes communicate by sending and receiving messages over a network
  (more scalable but programmer deals with messages)
- used heavily for resource sharing and coordination
- practitioners tend to call this "distributed computing"

Cluster computing systems

Cluster computing frameworks distribute CPU or I/O-intensive jobs across multiple servers. (Think Hadoop.)



Cloud and grid computing



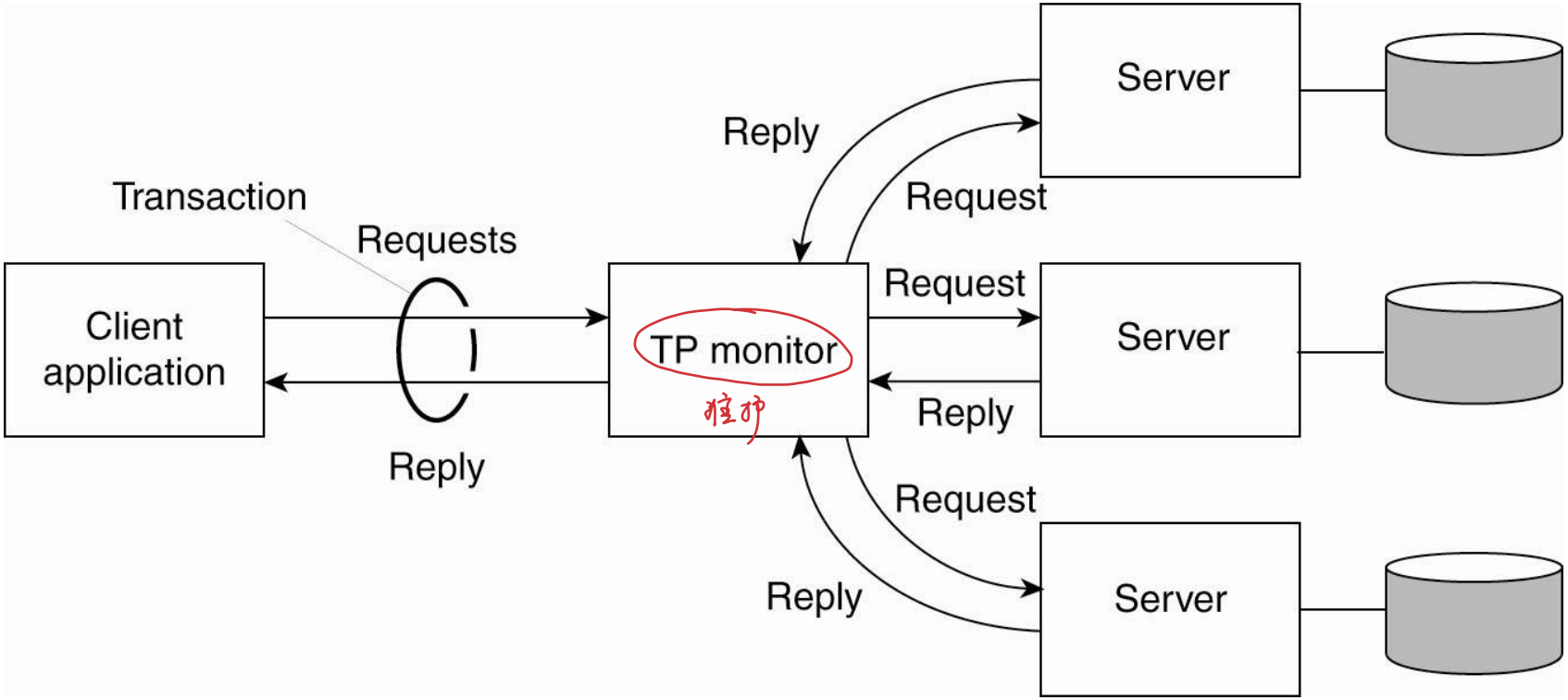
IaaS Amazon. Infrastructure as Service

PaaS google app Platform ...

SaaS gmail . google doc Software ...

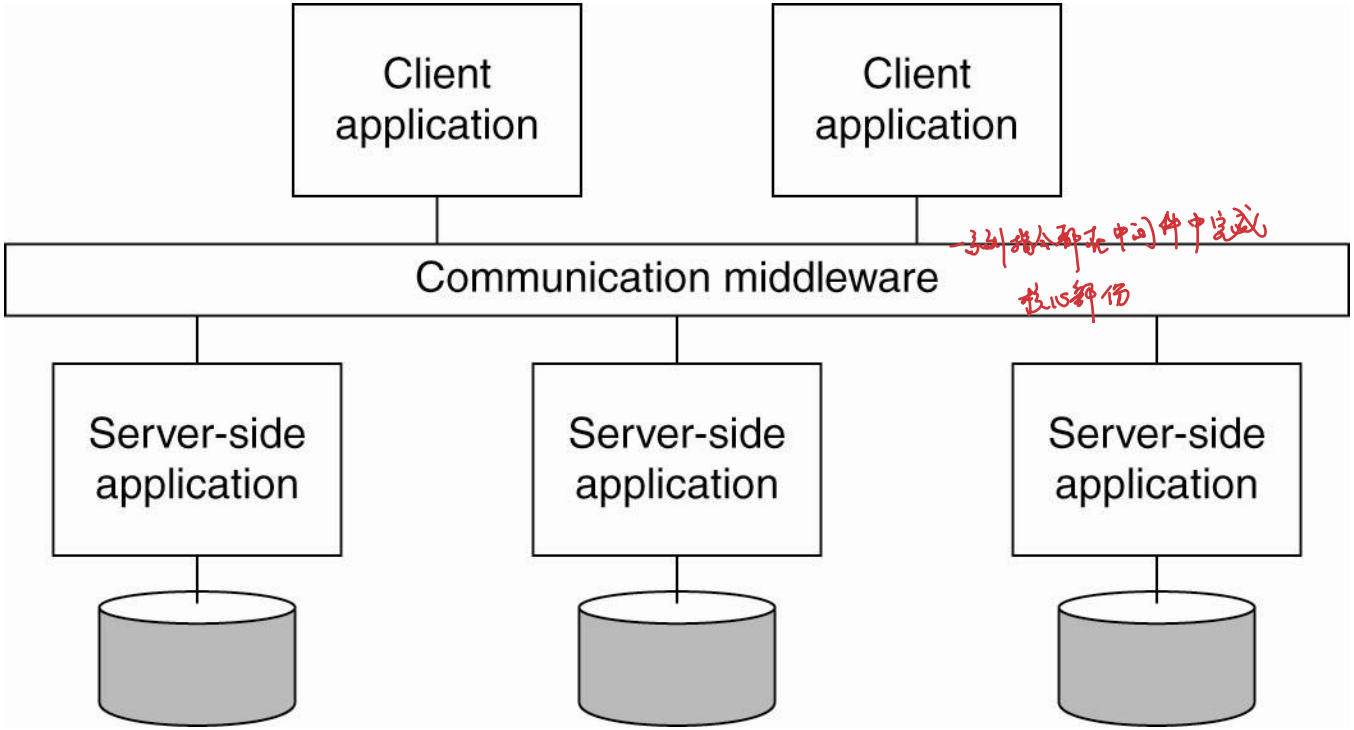
Transaction processing systems

Distributed transactions are coordinated by a **transaction processing (TP) monitor**.



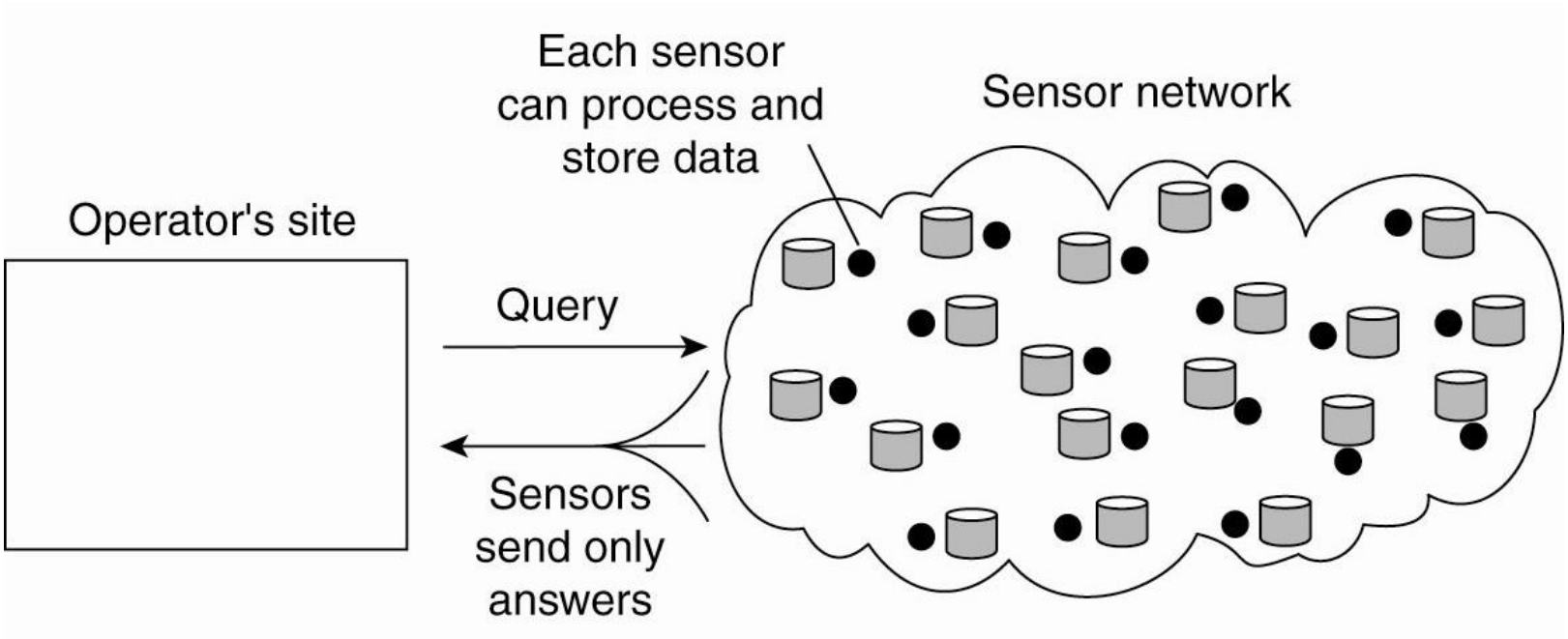
Enterprise application integration

Middleware is often used as a communication facilitator in enterprise application integration (EAI).



Sensor networks

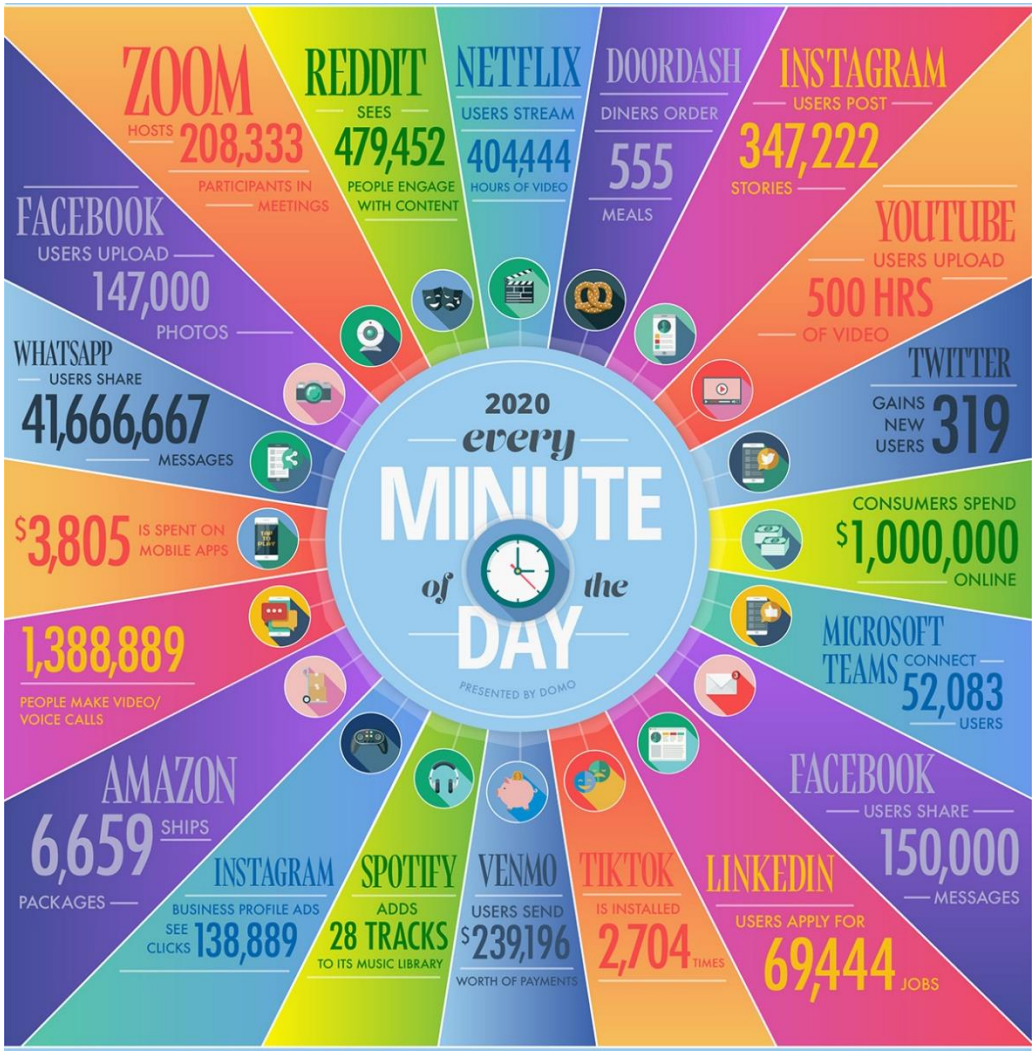
Sensor networks rely heavily on **in-network data processing** to reduce communication costs.



Example: activity tracking

"Person Wearing White Silicone Strap Black Smart Watch", used under CC0 license via Pixabay

Obligatory “60 seconds” slide



Source: https://www.visualcapitalist.com/every-minute-internet-2020/ (Aran Ali)