CS5223
Distributed Systems

Lecture 1: Introduction to Distributed Systems

Instructor: YU Haifeng

Outline

- What is a distributed system
- Why are distributed systems good
  - What are they not good for
- Common design goals of distributed systems and the transparency ambition

What is a Distributed System?

- *Steen & Tanenbaum*:
  - "A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system."
- *Leslie Lamport*:
  - "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."
- Our (vague) definition: Collection of computers linked by a computer network to achieve a certain goal

Examples of Distributed Systems

- Airline reservation system
- Bank automated teller machine network
- Search engines
- Blockchains
- Datacenters
- Mobile computing
- Sensor networks

Motivation: Why need them

- Resource sharing:
  - Remote printers, remote storage, blogs, music files, movies, etc
  - “Sharing” by definition reduce cost
  - Killer example: WWW, file sharing, E-commerce
- Coordination and communication:
  - Email, video conferencing, collaborative editing, remote surgery
  - Killer example: Games
- Cost-effectiveness and performance:
  - Cheaper to get a cluster with 100 commodity PCs than a super-computer
  - Can have comparable or even better computing power
  - Killer example: Datacenters

Motivation: Why need them

- Fault-tolerance:
  - Backup servers in web applications, banking systems
  - Killer example: Google, DNS
- Incremental growth:
  - Gradually grow your system to accommodate additional load
  - Avoid discarding existing machines
  - Killer example: Google, DNS

Distributed Systems: What made them possible

- Historically, computers are expensive
  - Cannot afford too many
  - PC changes the picture
- Historically, networking is not mature
  - LAN and Internet changes the picture
- Today: Almost every computer is connected to network, and often hundreds of thousands of PCs are involved in a certain task

Disadvantages of Distributed Systems: Complexity

- MUCH more complicated to design, to maintain, and often to use than a non-distributed system
- Designing a distributed system (often) requires radically different approaches from a non-distributed system
  - Computational complexity vs. message complexity
- Maintaining a distributed system probably needs a MS-degree instead of a Bachelor's
  - Install a shared file server vs. installing windows on a single PC
- Distributed systems are harder to use
  - To play a distributed game, one needs to select a "nearby server"
  - And what about migrating your state from one server to the other?

Distributed Systems: Common Design Goals

- Fault tolerance: system should continue functioning, when faced with various types of failures
  - Regardless of whether fault-tolerance was the motivation
  - E.g., motivation for game is only about coordination
- Mobility: users can log in from different sites and see the same environment
- Scalability: system should be easily adaptable to increased job/program size (service requirements)
  - Regardless of whether incremental growth was the motivation
- Heterogeneity: users can run applications over a heterogeneous collection of computers (big-endian vs small-endian)

Distributed Systems: Common Design Goals

- Openness: System should be able to be extended and re-implemented in various ways
  - Similar as a class definition in Java (declaring public methods and fields)
  - One can extend a Java class by inherit it
  - One can re-implement it
  - Also called “Separating Policy from Mechanism”
  - Openness = modular design in a centralize system

- Security: Obvious requirement for E-commerce and banking systems
  - Even games have security requirements

Distributed Systems: The Transparency Ambition

- Transparency: To the end user, a distributed system should look like conventional non-distributed systems, no distinction between local and remote resources
  - Hides all the “dirty things” behind the scene

- Transparency was emphasized a lot in early days of distributed systems
  - “Transparency” is part of Steen&Tenenbaum’s definition of distributed system
  - In early days, transparency was sometime the ONLY design target

Key question: What is good and bad about transparency?

History of Transparency and Why Is it Good

- In early days, there were no distributed systems
  - We knew about non-distributed systems and were comfortable with them
  - Then distributed systems come -- by human nature, we understand them by drawing connection with non-distributed systems
- Same applies to students learning about distributed systems first time
- Why it is good: Simplify the understanding and allow more people to use it.

If a distributed system S satisfies transparency, then anyone who knows how to use a non-distributed system can use S.

Why Is Transparency Bad

- In some cases we simply cannot achieve it…
- In some cases achieving transparency will sacrifice other things we care

- Example:
  - Impossible to obtain the same response time in a distributed game as in a non-distributed game – unless you want to argue against the speed of light
  - Impossible to completely hide the failure and recovery of a resource – user may experience a long delay and don't know whether the credit card has been charged
- Example
  - Impossible to completely hide replication in a distributed game – your view of the objects may lag behind

It is all about tradeoffs

If transparency is achievable, then it is all about tradeoff

Ease of use *vs.* Performance, Scalability, Security, etc.