Distributed Systems Cheatsheet

Edoardo Riggio

October 23, 2022

Distributed Systems - S.A. 2022
Software and Data Engineering
Università della Svizzera Italiana, Lugano

Contents

1 Introduction

With the advent in the mid 1980's of 16-, 32-, and 64-bit CPUs – as well as the invention of high-speed computer networks, made it possible for the creation of large numbers of geographically-dispersed networks of computers. These are known as **distributed systems**.

1.1 Definition

A distributed system is a collection of independent computers that appear to the user as a single coherent system.

Each computing element of these massive systems are able to behave independently from one another. A computing element is often referred to as a **node**. This node can either be a hardware device or a software process.

1.2 Consequences

Some of the main negative consequences that arise when dealing with distributed systems are the following:

- **Concurrency**
  This happens when several processes try to read and write on a shared storage service.
- **Absence of a global clock**
  Each node will have its own notion of time. This means that there is no common reference of time between the nodes.
- **Failure independency** The failure of a node can make another node unusable.

1.3 Challenges

Some of the challenges that arise when dealing with distributed systems are outlined in the following sections.

1.3.1 Openness

The services are offered according to standard rules. These rules describe both the syntax and semantics of such services. The standard rules of distributed systems are called **COBRA**(Common Object Request Broker Architecture).

1.3.2 Scalability

Scalability can be with respect to the **system size** – which would mean adding more users to the system; to the **geography** – which would deal with users lying far apart from one another; and to **administration** – which would deal

with the complexity to manage an increasing system.

Some scalability techniques are the following:

- **Hiding communication latencies**
  This is important for **geographical scalability**. Asynchronous communications can be used to reduce the waiting time of the users. This can be achieved with the use of batch processing and parallel applications.
- **Distribution**
  Components are split into parts and spread across the system. An example of this would be the DNS, where we have a tree of domains divided into non-overlapping zones. Furthermore, the name in a zone is handled by a single name service.
- **Replication**
  This can be use to increase both **availability** and **performance**, as well as reduce **latency**. Moreover, caching can be used as a form of replication, but is typically done on-demand.

1.3.3 Transparency

Transparency is the ability of a system to hide some of its characteristics or errors to the user. There exist several different forms of transparency:

- **Access transparency**
  Hide the differences in data representation and machine architecture.
- **Location transparency**
  Users cannot tell where a resource is physically located.
- **Relocation transparency**
  Even if the entire service was moved from one data center to the other, the user wouldn't be able to tell.
- **Migration transparency**
  Moving processes and resources initiated by users, without affecting any ongoing communication and operation.
- **Replication transparency**
  Hide the existence of multiple replicas of one resource.
- **Concurrency transparency**
  Each user is not going to notice if another user is making use of the same resource.
- **Failure transparency**
  The user or application does not notice that some piece of the system fails to work properly. The system is then able to automatically recover from the failure.

1.4 Types of Distributed Systems

There are three main types of distributed systems: **distributed computing systems**, **distributed information systems**, and **distributed pervasive systems**.

1.4.1 Distributed Computing Systems

These systems aim at high-performance computing tasks. These systems can be part of two subtypes:

- **Cluster Computing**
  All the resourced are located in a local-area network, and are using the same OS. In addition, they also have a common administrative domain.
- **Grid Computing**
  This is a "federation" of computer systems. Such systems may have different administrative domains, different hardware, software... Such systems are used to make collaboration between organisations feasible.

1.4.2 Distributed Information Systems

These systems are based on transactions. These transactions are used to make systems communicate between themselves. Transactions follow the **ACID** properties:

- **Availability**
  To the user, the transaction happens indivisibly.
- **Consistency**
  The transaction does not violate system invariants.
- **Isolation**
  Concurrent transactions do not interfere with each other.
- **Durability**
  Once a transaction commits, the changes are permanent.

The application components of each node communicate directly with each other.

1.4.3 Distributed Pervasive Systems

These systems are composed by mobile and embedded computing devices. This means that pervasive systems need to have the following characteristics:

- Embrace contextual changes
- Encourage ad-hoc composition
- Recognise sharing as the default

Some examples of such architectures are home systems, electronic health care systems, and sensor networks.