Coordinating Distributed Systems

Theory and practice

Daniele Venzano

Eurecom

Introduction

Outline

- **What is a distributed system?**
- **The consensus problem**
  - A few examples of distributed consensus
  - CAP theorem
  - Eventually consistent Vs Strongly consistent
  - Fault tolerance: possible faults in distributed systems
- **Consensus protocols**
  - Raft from A to Z
  - Paxos overview
  - Other protocols
- **Implementations - Zookeeper**
  - History
  - Architecture
  - Data model
  - Higher-level primitives

What is a distributed system?

- A set of processes seeking to achieve some common goal by communicating with each other

What is a distributed system?

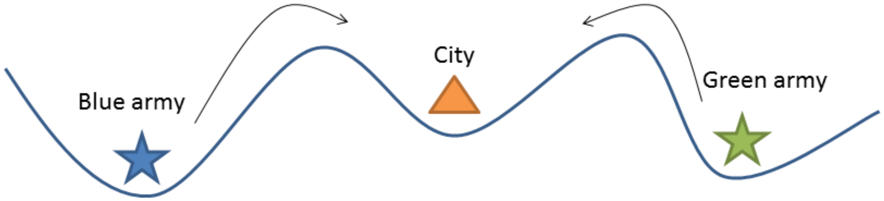
- Like software matryoskas
  - Multi-threaded process
  - Multi-process on a single server
  - **Multiple processes in a set of servers in the same datacenter**
  - **Multiple processes in a set of geographically distributed servers**
- Why?
  - Inherent distribution (sensors, peer-to-peer, publish-subscribe, ...)
  - Engineering choice (fault tolerance, replication, performance, ...)
- These processes need to coordinate to reach a common goal:
  - data aggregation
  - synchronization
  - transactions
  - ...

The consensus problem

Wedding consensus

- The priest follows a well known protocol to reach a consensus:
    1. Priest: Alice, will you marry Bob ?
    2. Alice: yes
    3. Priest: Bob, will you marry Alice ?
    4. Bob: yes
    5. Priest: You are now husband and wife
- In distributed systems this becomes:
    1. Coordinator: Alice, can you commit key X with value 5 ?
    2. Alice: yes, I can
    3. Coordinator: Bob, can you commit key X with value 5 ?
    4. Bob: yes, I can
    5. Coordinator: Ok, both of you record that X has now a value of 5
- What if Bob flees from the church?

The two generals



- Two generals want to attack a city
- They can only use unreliable messengers to communicate
- They need to attack at the same time to succeed

An infinite number of messages is needed for each general to be sure the other agrees on the time of the attack.

Consensus examples

Processes need to reach a common goal:

- aggregate functions: sensors calculating average temperature
- synchronization: agree on a value, elect a leader
- reliable broadcast: a message sent to a group is received by all or none
- atomic commit: ensure that processes reach a common decision whether to commit or abort a transaction

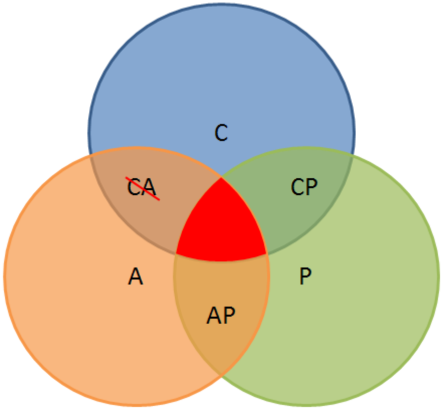
Properties of a distributed system

- Linearizability (Consistency): a given set of operations is *linearizable* if it appears to the rest of the system to occur instantaneously
  - Writes are linearizable operations if every read receives the most recent write or an error
- Availability: a system is *available* if every request to a non-failing node always receives a response, eventually
  - Reading stale data is ok, though
- Partition tolerance
  - The system continues to function properly even if the network loses or delays an arbitrary number of messages

The CAP theorem links these three properties.

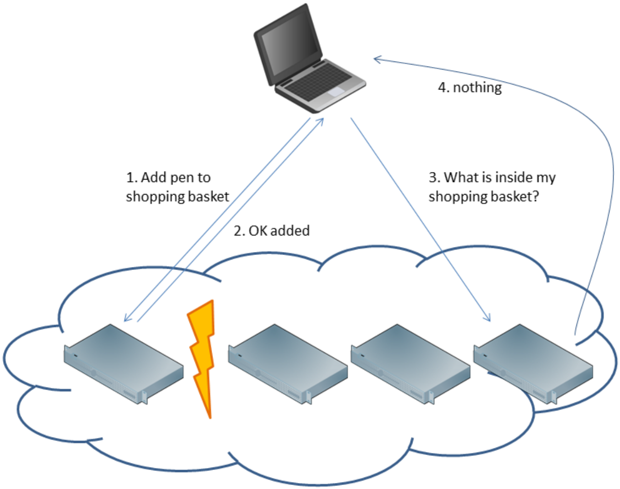
CAP theorem - 1

The theorem says: between C A P, you can choose only two



(... but you need the P ...)

CAP theorem - 2 - Why not all three?


1. Add pen to shopping basket
2. OK added
3. What is inside my shopping basket?
4. nothing

If there is a network partition either C or A will break

CAP theorem - 3 - P

- Network partitions occur outside anyone's control in real life
- Cannot sacrifice the Partition-Tolerance property
- In the event of a network partition either A or C is maintained: it is the choice of the designer
- Practical distributed systems are CP or AP
- Some can be configured to shift between CP and AP (tunable consistency)

Note: a network partition can also be a very slow link

CAP theorem - 4 - Summary

- First stated by Eric Brewer (Berkeley) at the PODC 2000 keynote
- Formally proved by Gilbert and Lynch, 2002[1]

- The CAP theorem formally states the trade-offs among different distributed systems properties
- In practice network Partitions occur, the designer can choose one of Consistency or Availability
- The choice heavily depends on what your application/business logic is

CAP theorem - 5 - Summary

CP-oriented systems:

- BigTable, Hbase, MongoDB, Redis, MemCacheDB, Scalaris, Paxos, Zookeeper[1]

AP-oriented systems:

- Amazon Dynamo, CouchDB, Cassandra[1], SimpleDB, Riak, Voldemort

In-depth articles on the misuse of the CAP theorem in describing real systems:

- `https://codahale.com/you-cant-sacrifice-partition-tolerance/`
- `https://martin.kleppmann.com/2015/05/11/please-stop-calling-databases-cp-or-ap.html`

[1]CA or CP tunable

Consensus protocols

Zookeeper

History

Architecture

Higher-level primitives

References I

[1] Seth Gilbert and Nancy Lynch.
Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.
SIGACT News, 33(2):51–59, June 2002.