Introduction, Misc Concepts

13 November 2017

Ignacio Coterillo
Computer Engineer, CERN

About me (1/4)

- 2000 - 2005: Communications Engineer in the University of Cantabria

```
Specialization in Radiocomunications
```

- 2005 - 2006: Master Thesis in the University of Gent, Belgium

```
Electrocardiogram classification using Wavelet Domain features
```

- 2006 - 2011: Institute of Physics of Cantabria (CSIC/UC)

```
Grid Computing and e-Science
```

- 2011 - 20XX: CERN

```
CERN's IT Department, Databases Group
```

About me: Master Thesis (2/4)

First contact with "big" data and "data science" methods, working with Raw ECG data in Time-Voltage measurements with some metadata in additional files.

Over 360 hours of ECGs (e.g. WAV files) from different sources (kind of sensors)

ETL: Pre-processing, filtering, enrichment and re-organizing

Analysis (Wavelets decomposition)

Machine learning for classification

About me: IFCA (3/4)

- The DORII Project: Deployment of Remote Instrumentation Infrastructure

Geographically distributed sensor system, including data taking, transport, storage and replication.

- A prototype for a mobile sensor tracking system with integrated pattern analysis

IoT like project, targetting the tracking of mobile phones at a geographical level, coping with issues of high cardinality of the problem, high data throughput, big volume.

About me: CERN (4/4)

As part of the CERN IT Databases group, my main focus has been on the Database on Demand service, starting first as backend developer, later as architect and main administrator and currently as project leader and service manager.

Currently hosting around 600 databases (**MySQL**, **PostgreSQL** and **InfluxDB**) for all CERN departments and most user communities (IT Services, Administration, Experiment support, etc.)

A biased recent Big Data Time Line in (mostly Google) papers

- 2003: Google File System (https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf)
- 2004: Google Map-Reduce (https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf)
- 2006: Google BigTable (https://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf)
- 2007: Amazon Dynamo (http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf)

2007 - 2008: **NoSQL**, SQL is dead, RDBMS are bad and also dead: MongoDB, Redis, Riak, etc. grow popular

2009: Yahoo open sources **Hadoop**, inspired by GFS and the MR framework

- 2013: Google F1 (https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41344.pdf)

SQL is cool again

- 2016: Google Spanner (https://www.usenix.org/system/files/conference/osdi12/osdi12-final-16.pdf)

How big is the "Big" in Big Data?

Whatever, depending on the technology state of the art, or your budget

TeraSort (2008) (http://sortbenchmark.org/YahooHadoop.pdf)

When to use Big Data technologies?

When your data problem is not solvable with your standard tools and resources

"When your data doesn't fit in memory"

Not only size (The three V's)

Gartner (Original) (https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf)

Stonebraker (https://cacm.acm.org/blogs/blog-cacm/155468-what-does-big-data-mean/fulltext)

- Big Volumes
- Big Velocity (High write throughtput, RT Analytics)
- Big Variety: Large number of data sources/formats

The CAP theorem (1/2)

Brewer's Conjecture (https://github.com/papers-we-love/papers-we-love/blob/master/distributed_systems/brewers-conjecture.pdf)

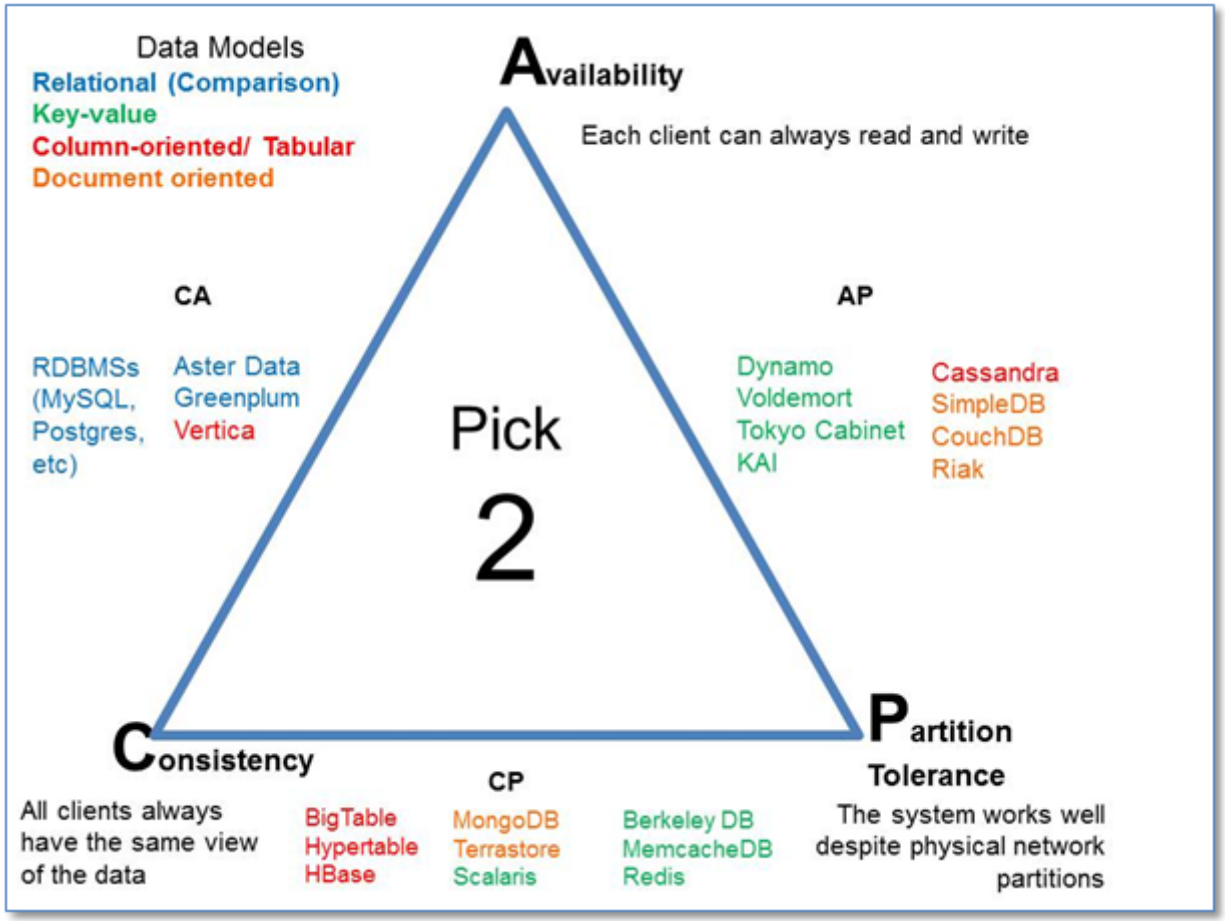
There are three desired characteristics for a distributed system

- Consistency
- Availability
- Partition tolerance

TL;DR

```
In the case of a network (P)artition (eventually bound to happen) a distributed
system can have either (C)onsistency or (A)vailability
```

The CAP theorem (2/2)



ACID vs BASE (1/3)

Atomic: Everything in a transaction succeeds or the entire transaction is rolled back.

Consistent: A transaction cannot leave the database in an inconsistent state.

Isolated: Transactions cannot interfere with each other.

Durable: Completed transactions persist, even when servers restart etc.

ACID vs BASE (2/3)

	Dirty reads	Non-Repeatable Reads	Phantom Records
READ UNCOMMITTED	Yes	Yes	Yes
READ COMMITTED	No	Yes	Yes
REPEATABLE READS	No	No	Yes
SERIALIZABLE	No	No	No

As a general rule, the more protection you have the worse concurrency performance you will get

ACID vs BASE (3/3)

Basic Availability: Tolerance to multiple failures, generally by using a highly distributed number of replicas. The lost of a single chunk of data shouldn't impact the whole database

Soft-state: ACID like requisites for the data are pushed to the application level and considered responsability of the application developers.

Eventual consistency: The system should be consistent at some point *in the future*

Scaling: Vertical vs Horizontal

- Vertical scaling

AKA: Buy bigger computers (More memory, more storage, more CPUs, faster IO, more bandwidth)

- Horizontal scaling

Partition the problem, ideally in a linear manner, so it can be treated with N smaller/cheaper (compared to the vertical scaling option) elements.

Scaling: Storage (1/2)

RAID (Redundant Array of Inexpensive Disks) is a data storage virtualization technology that combines multiple physical disk drive components into a single logical unit for the purposes of data redundancy, performance improvement, or both.

RAID (Wikipedia) (https://en.wikipedia.org/wiki/RAID)

- Software
- Hardware

Scaling: Storage (2/2)

Network Attached Storage: A *single* element providing access to storage over network (typically using the NFS protocol)

Storage Area Networks: A local network of multiple devices. Typically using specialized cabling (Fibre Channel) and exposing resources as block devices.

Scaling: Costs

- Open Source Licenses: Total cost of ownership?
- Commercial Licenses: Licensing models

```
- Price per node
- Price per cpu/core
```

Scaling: Self-Hosted vs "Cloud"

- Again, **TCO**
- Can your data actually be in the cloud? (GDPR, confidentiality, latency ...)
- How critical is your data platform to your organization/business?
- Even more important to prevent vendor lock-in than running on-premises
- Expenses can be elastic according to actual use
- Pricing can vary day to day

Polyglot systems

Fancy way of saying data is stored in different places

Good if the data is normalized, otherwise consistency needs to be enforced

Think monolith vs micro-services architectures for application development

Too many systems will increase complexity in unpredictable ways

Misc

Almost all *database* like systems have certain overlap with others in functionality and features.

There's always a trade off to be made: in the data model, in the solutions, the implementations, ...

Active databases normally require low level (Kernel) tuning: IO, MEM, TCP

> Latency numbers every programmer should know (https://gist.github.com/hellerbarde/2843375)

Thank you

Ignacio Coterillo
Computer Engineer, CERN
ignacio.coterillo.coz@cern.ch (mailto:ignacio.coterillo.coz@cern.ch)