Architectures for Big Data

Federico Bruzzone

27 settembre 2022

Indice

1 Course presentation

The course aims at describing **big data processing framewokds**, both in terms of **methodologies** and **techonologies**.

Part of the lesson will focus on **Apache spark** and **distributed patterns**.

"May I ask..." a brave student voice break the presentation.

It is not a spurious correlation

- What an Architecture is?
- Why so I need to know this stuff?
- What is this "Hadoop"? Do I reallt need to know what a Name Node is?
- I would like to put a jBoss inside a Docker to allow Kubernetes load balancing it! (No! This is too much even for a joke)

1.1 You are going to learn

- How to **distribute computation** over clusters using Map Reduce model
- How to write **Apache Spark** code
- How **Hadoop works** and why it works that way
- What a **software architecture** is
- How to design batch architectures to manage **data workflows**
- Several **design patterns** that could be used in a **distributed** environment
- The **limit of traditional SQL** with Big Data

1.2 Topics Overview

1. Enterprise Architectures
2. Design Patterns
3. Hadoop
4. Distributed Algorithms
5. Big Data and SQL
6. Big Data Document
7. Containers

1.3 Technologies Overview

1. Python
2. Apache Spark - Resilient Distributed Dataset
3. ELK Stack: Elastic Search, Logstash, Kibana
4. Docker

1.4 Workshops Overview

1. Workshop 1 - R. Tommasi (Marelli)
2. Workshop 2 - F. Palladino (artea.com)
3. Workshop 3 - D. Malchiodi (Unimi)
4. Workshop 4 - D. Malagodi (Google)

2 Architecture 101

Architectures:

- The art or practice of **designing** and **building** structure and especially habitable ones.
- A unifying or coherent **from** or **structure**

Foundation for the study of Software Architecture / L. Wolf, 1992

Software architecure principles can be **inherited** by appealing to several well-established architectural disciplines.

While the subject matter for the two is quite different, there are a number of intresting **architectural points** in building architecture that are suggestive for software architecture

- multimple **views**
- architectural **styles** item style and **materials** +

2.1 Multiple Views

2.1.1 Building Architecture

Building Architecture uses MULTIPLE VIEWS

A building architect works with the customer by means of a number of different views in which sone **particular aspect of the building** is emphasized.

For exmaple, there are elevations and floor plans that give the **exterior views** and "**top-down**" views, respectively.

The elevation views may be supplemented by **contextual drawings** or even scale models to provide the customer with the look of the building in its context.

2.1.2 Different Stakeholders

Each perspective is not just a matter of different level or detail.

It is linked with **different natures** and **accountability**.

- The **Owner** needs the building for a specific purpose. He/she does not know how, but hw/she knows perfectly **why**
- The **Architect** needs to project and formalize something that fit completely with owner's needs, to plan the **what**
- The **Builder** needs to design **how** the what will be built matching with natural laws and techological costraints

2.1.3 ~~Building~~ Software Architecture

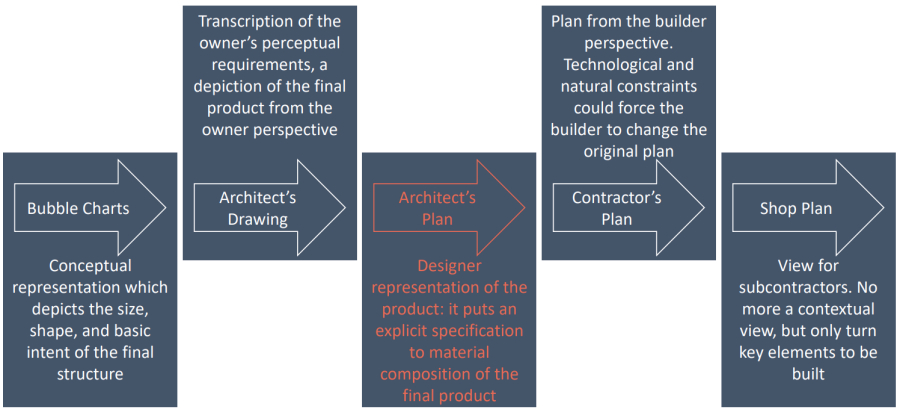
~~Building~~ Software Architecture uses MULTIPLE VIEWS

Different **type of users** will use Software Architecture: each of them will need a specific point of view.

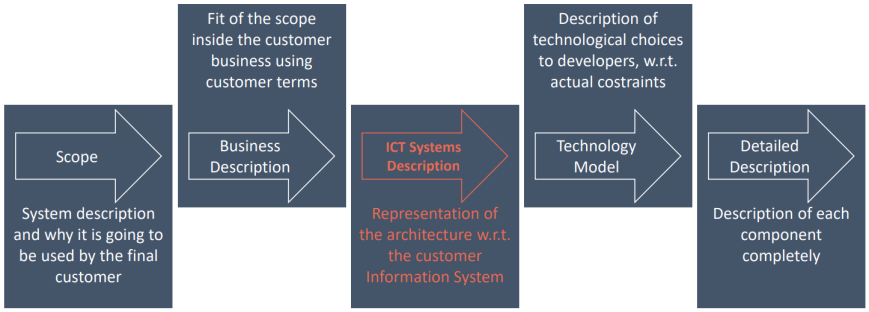
A **Full Stack** developer needs to know how to write code inside the Architecture while a **Data Scientist** where are data they need.

*Since the technology permits **destributing** large amounts of computing facilities in small packages to **remote location**, some kind of structure (or architecture) is imperative because **decentralization without structure is chaos**.*

2.1.4 Zachman Framework for Building



2.1.5 Zachman Framework for Information System



2.1.6 Different point of views

Each perspective is not just a matter of different level of detail.

It is linked with **different natures** and **accountability**.

- **Input-Process-Output**

  Product description in detail w.r.t. intended capabilities, appearance, and interactions with users

- **Entity-Relationship-Entity**

  «Stuff things is made of», description of data in each building blocks

- **Node-Line-Node**

  Flows between each componenet