# Jazz, a lightweight data processing framework

Open Source software released by

BBVA Data & Analytics

## 1. OSS in a Large Corporation

- Putting a corporation exactly upside down
- Wearing the running shoes
- Jazz @ BBVA

# It takes a lot of courage to do OSS in a large corporation

In a hypothetical pre-digital corporation that wants to implement a new product, the decision process could be:

- **Who** leads the project?
- **What** will we do?
- **How** can we do it?

## It takes a lot of courage to do OSS in a large corporation

In a hypothetical pre-digital corporation that wants to implement a new product, the decision process could be:

|  |  |
|---|---|
| • **Who** leads the project? | **Me**, It’s **my** budget. |
| • **What** will we do? | **Another** xyz in 6 months. (#metoo) |
| • **How** can we do it? | Get **someone who knows**. |

# In OSS, priorities are in reverse order

- **How?** We have identified an opportunity, are sick of the wrong tools, **need** to do it right once and for all.
- **What** is our low hanging fruit? What problem are we going to tackle **with our new superpower?** What market is ready for what we can offer?
- **Who?** Anyone who shares the vision, has the skills (actually any skills are welcome) and wants to be part of it **for the sake making it happen**.

4 years ago someone said about BBVA Data & Analytics “You are putting the bank upside down.”

Still today, I am not sure if that was meant as a compliment.

1.1 Putting a corporation exactly upside down

## “Wearing the running shoes”

**Athletics** is not about who looks cooler in sportswear or is wittier in front of a journalist, … its only about **measurable performance**.

- **Kaggle**
- **Computer Go** Olympiad
- OSS is **humbling**, it puts you **in the position you earned**
- Your **claims can be verified**

Corporations typically don't like that.

## 1.2 Wearing the running shoes

# Big Thanks to BBVA

Releasing Jazz as OSS shows a lot of **courage**, **vision** and **commitment.**

**Digital transformation** is **not just** about creating great apps, it is a fundamental change in the relationship with **people**.

## Now, we are on our own, with our strengths …

- We don't just **shoot at the moon**, we shoot at a moon in a distant galaxy.
- We have **decades of successful experience**, in AI to understand what works and why.
- **We deliver**.
- We stand **on the shoulders of giants**. **OSS**
- We are used to **wearing the running shoes**.
- We **enjoy** doing it.

## … and weaknesses

- We still have to finish the MVP.
- We have to **create a community**.
- We need success stories.
- Short version: We need help.

## 2. AI technology in 2018

- The perfect world in a data pipeline
- What makes the perfect world crumble?
- Implementing Alphago without C++
- So, what is the point?

## First, an apology for making this a little bit too technical

**The perfect world in a data pipeline**

• 99% of the running time in highly optimized thread-safe C++ (or go) code crunching numbers, committing persisted records, processing messages, etc.

• 1% time running business logic written in a productive, expressive, well engineered, scripting language (e.g., Python).

## What makes the **perfect world crumble**?

- Implementing **new ideas**: Scripting languages scratch the surface, with new ideas, we have to re-implement what is in the lower layers.
- Frameworks force **a way of thinking**.
- Persistence as a **different process**.
- APIs as a different process, APIs using scripting languages in the back-end.
- No native support for RL, binary formats, …

## Try to implement **Alphago** without writing in C++ (or any real language)

Alpha Go is **MCTS** + **DNN**

- Without **DNN**: Alphago is still a world class player, actually top world class in 9x9.
- Without **MCTS**: Alphago plays good moves at wrong moments doing enough mistakes to be weak amateurish.

## So, **what is the point?**

In short:

• Current frameworks are far from ideal for implementing radically different ideas.

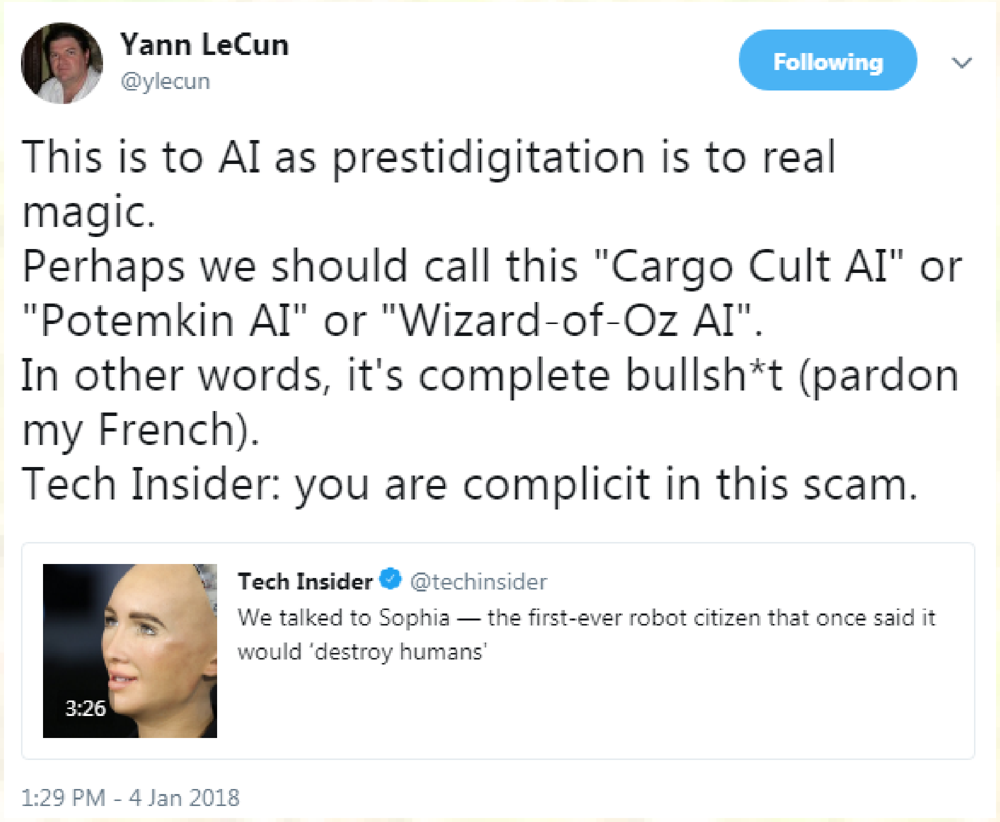
• There is **a huge room for improvement**.

## 3. AI criticism circa 2020

- Unfocused
- How are they unfocused?
- Focused

## We live an unprecedented AI hype

On one side, makers of false claims deserve derision …



Yann LeCun
@ylecun

Following

This is to AI as prestidigitation is to real magic.
Perhaps we should call this "Cargo Cult AI" or "Potemkin AI" or "Wizard-of-Oz AI".
In other words, it's complete bullsh*t (pardon my French).
Tech Insider: you are complicit in this scam.

Tech Insider @techinsider
We talked to Sophia — the first-ever robot citizen that once said it would 'destroy humans'

3:26

1:29 PM - 4 Jan 2018

## … on the other, “unfocused” criticism raises

“Unfocused” includes:

- Metaphysical impossibility
- Psychological arguments
- Neuro-scientific views
- Anthropo-biased Armageddon scenarios
- Quantum-entangled obsessions
- Hype-surfers trying to look brilliant

## How are they unfocused?

- **AI is in math and CS**, just like aerodynamics is in math and physics, **not in ornithology**.
- Biology is, at most, a source of inspiration. Btw. Cells are anything but simple, **compositionality** is. If evolution did it, it must have been **incrementally**.
- The **implications of the Church-Turing Thesis are profound**: All known computation paradigms are interchangeable, including our brain and a computer.
- There is no evidence that we need significantly **more computing power** than we have. Actually, estimates speak against it.

# And focused

**Judea Pearl**

✓ Theoretical Impediments to Machine Learning With Seven Sparks from the **Causal Revolution**

**François Chollet**

✓ *"For most problems where deep learning has enabled transformationally better solutions (vision, speech), **we've entered diminishing returns territory** in 2016-2017."*

**Geoff Hinton**

✓ *"Science progresses one funeral at a time. The future depends on some **graduate student who is deeply suspicious of everything I have said**."*

**Gary Marcus**

✓ Deep Learning: **A Critical Appraisal**

**Joscha Bach**

✓ Principles of **synthetic intelligence** - The necessary ingredients of thinking machines.

**Filip Piekniewski**

✓ *"Predicting the **A.I. winter** is like predicting a stock market crash - impossible to tell precisely when it happens, but almost certain that it will at some point."*

…

## 4. AI technology the Jazz way

- Everything is a block!
- What makes it so special?
- Python, R, front end
- The roadmap
- The most important thing in OSS

## In Jazz, everything is a block!

Blocks are (at the simplest)

- **Tensors**(*) with attributes
- **Files** with an instruction manual (**)
- **Web resources** including (urls, mime types, etc.) (**)

(*) A tensor is a vector of vectors of vectors … of **one type**.
(**) All are the same: **tensors with attributes**.

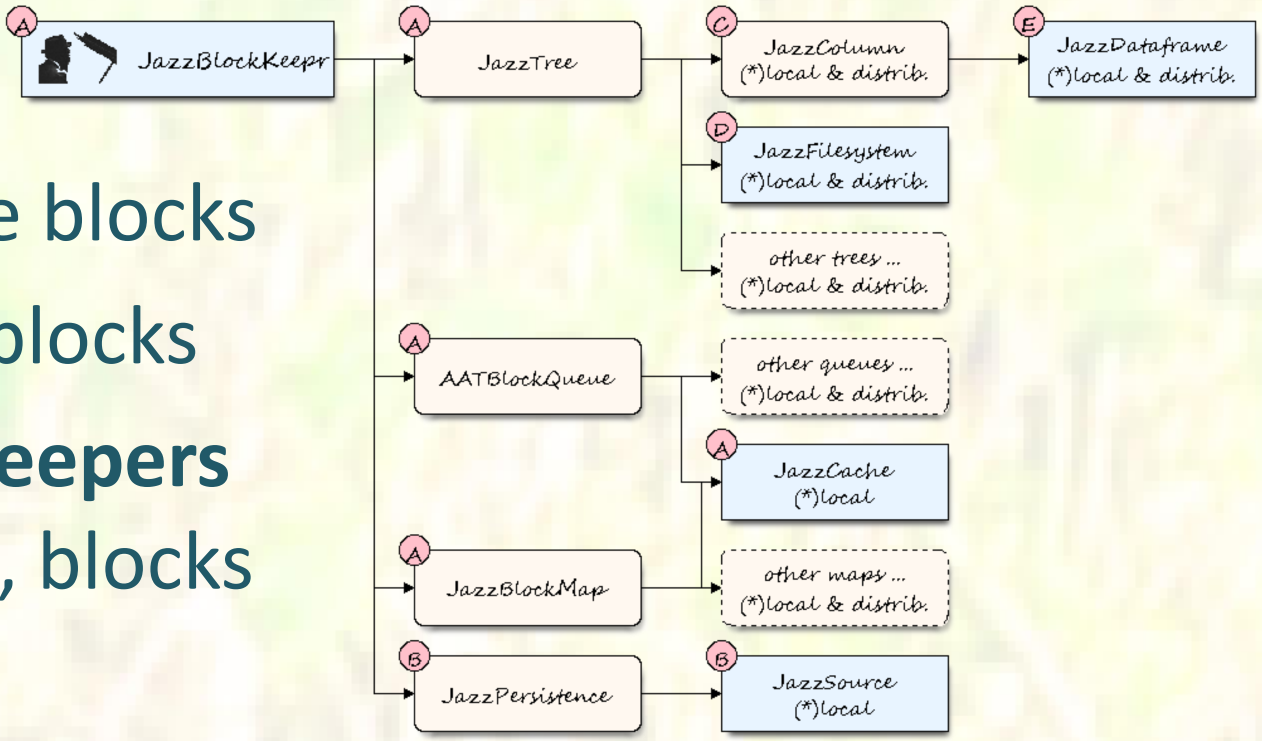
# In Jazz, everything is a block!

- **Blocks** are owned by **keepers**(*).
- **Bebop**(**) **functions** do operations on **blocks**.
- Bebop functions are **methods of classes**.

(*) Efficient C++ structures: queues, pipes, trees, priority queues, maps, memory mapped files, etc.

(**) **Bebop** is a thread safe, automatically paralleled language with support for **automatic programming**.

## In Jazz, everything is a block!



- **Functions** are blocks
- **Keepers** are blocks
- **Classes are keepers**

and, therefore, blocks

Okay, stop!
So far you are describing yet another data processing framework, what makes it so special?

# What makes it special?

**1.** Blocks are **volatile**, **persisted**, **distributed** or **ubiquitous** as defined. Abstraction from **extremely small** to **extremely large** is automatic. The same function works on a small vector or a distributed data-frame.

## What makes it special?

**2.** Extremely efficient persistence (a memory mapped file) in the same process allows **sampling without copying**. (No need of a DB.)

**3.** Extremely efficient **REST API** in the same process **with a thread pool**. (No need of an http server.)

## What makes it special?

**4. A full language** that is close to perfect performance (assuming enough tensor size), thread safe and auto distributing instead of just **some optimized data mangling primitives**. (No need of an interpreter.)

## What makes it special?

**5. Distributed data-frames where cells are tensors** (E.g., video or sound tracks, not links to files)

**6.** An **abstraction over the file system** that allows reading/writing binary and compressed files on demand.

# What makes it special?

**7. Tree Search** as a first class citizen

**8. Reinforcement learning** "out of the box"

**9.** Inference on **small samples** done efficiently

**10.** Support for **automatic programming** at bytecode level

## What makes it special?

**11. Deep Learning as a first class citizen**

Of course!

It’s just **not our only** (or even our first) **priority**.

**Jazz** can be as efficient doing **just DL** as a specialized platform.

## I am a Python guy, an R lady, a front end developer ...

You all have all the necessary skills to use Jazz.

- install.packages('rjazz')
- pip install pyjazz
- docker run kaalam/jazz_neat.latest

**4.3 Python, R, front end**

# As a user, when can I start?

**A brief history of Jazz**:

- **2015/2016(Q3)** A predecessor project code named *"b-shades"* was developed at BBVA D&A as a proprietary solution to efficiently access individual data in large sparse tables. It is an R package written in C++ using HBase and HDFS as a persistence.
- **2016 (Q4)** BBVA D&A decides to pivot *"b-shades"* to a technology independent solution in just one process written in C++, the new project is named **Jazz**.
- **2017 (Q3)** Organizational changes at BBVA D&A delegate the development of data technology to a new corporation. At the time, **Jazz** was already fully functional, the fastest data block storage by large, the only one **with native support for R objects** and was used as an http server in internal solutions.
- **2017 (Q4)** After intense code review considering **the high quality of the code, the documentation and the software engineering involved**, **Jazz was released as OSS under an Apache 2.0 license** on the 2017-12-11 17:42 GMT+01:00. Both the server and the R package. Jazz is still today the project with the best software engineering metrics at BBVA D&A by large.
- **2018 (Q1)** The released R client was, again after a code review, accepted as an official R package in cran (The Comprehensive R Archive Network) as the package **rjazz**.
- **2018 (Q1)** A meritocratic society of volunteers started refactoring Jazz into the product described in this presentation. Over 1000 working hours have been spent in Jazz development in the first 5 months.

# As a user, when can I start?

**FUTURE**:

- **2018 (Q4)** The first MVP of Jazz, as described in this presentation, will be released.
- The version number will be **0.3.1**
- It will include everything as described, but will still lack large libraries of useful primitives to make it "the platform of choice" for average users. It will take time to build enough momentum to make it into the finals **"wearing the running shoes"**.
- But again, we do this **for the sake of making it happen**.

And remember, the product is **the second most important thing** in any Open Source Software Project

**The most important thing** in any Open Source Software Project is, of course …

## The community

4.5 The most important thing in OSS

## We need volunteers!

4.5 The most important thing in OSS

# Thank you!

Release only: **https://github.com/bbvadata/Jazz**

Development: **https://github.com/kaalam/Jazz**

Programming doc: **https://kaalam.github.io/develop**

Jazz reference: **https://kaalam.github.io/jazz_reference**

**kaalam.ai**

**@kaalam_ai**