Hadoop the worng way

Iker Martinez de Apellaniz
Data Engineer at Schibsted

Hi!!

```
/** @author imartinez*/
Person me = new SoftwareEngineer();
me.setName("Iker Mtz de Apellaniz Anzuola");
me.setTwiter("@mitxino77");
me.setChildNumber(2);
me.setLocations({
  "St. Cugat, Barcelona",
  "Kanpezu, Euskadi",
  "*, World"
});

me.addExperience({
  "Data Engineer at Schibsted",
  "Big Data Team Lead at GFT",
  "Research & Development at Emagister",
  "Other Development Stuff"
});

me.addMainSkills({
  "Hadoop",
  "Search",
  "Agile",
  "Java"
});

return me;
```

“Errare humanum est”

–Séneca

“One who makes no mistakes makes nothing”

–Giacomo Casanova

“We learn from failure, not from success!”

–Bram Stoker, Dracula

“Mistakes are proof that you are trying”

“Mistakes are meant for learning, not repeating!”

Everything

There are lots of stones on the path

- Ignorance on what you are doing
- That moment before coffee
- That brilliant Idea
- We don’t have time for that
- We need to think about the future

Type of errors you can do/suffer and how to (try to) avoid them

- New technologies require time to learn and configure properly
- You won’t see your own mistakes
- Check and ask feedback on your design
- Don’t sacrifice critical features before going life
- Do a MVP/PoC previous to a gigantic App

#1: Configuration Issues

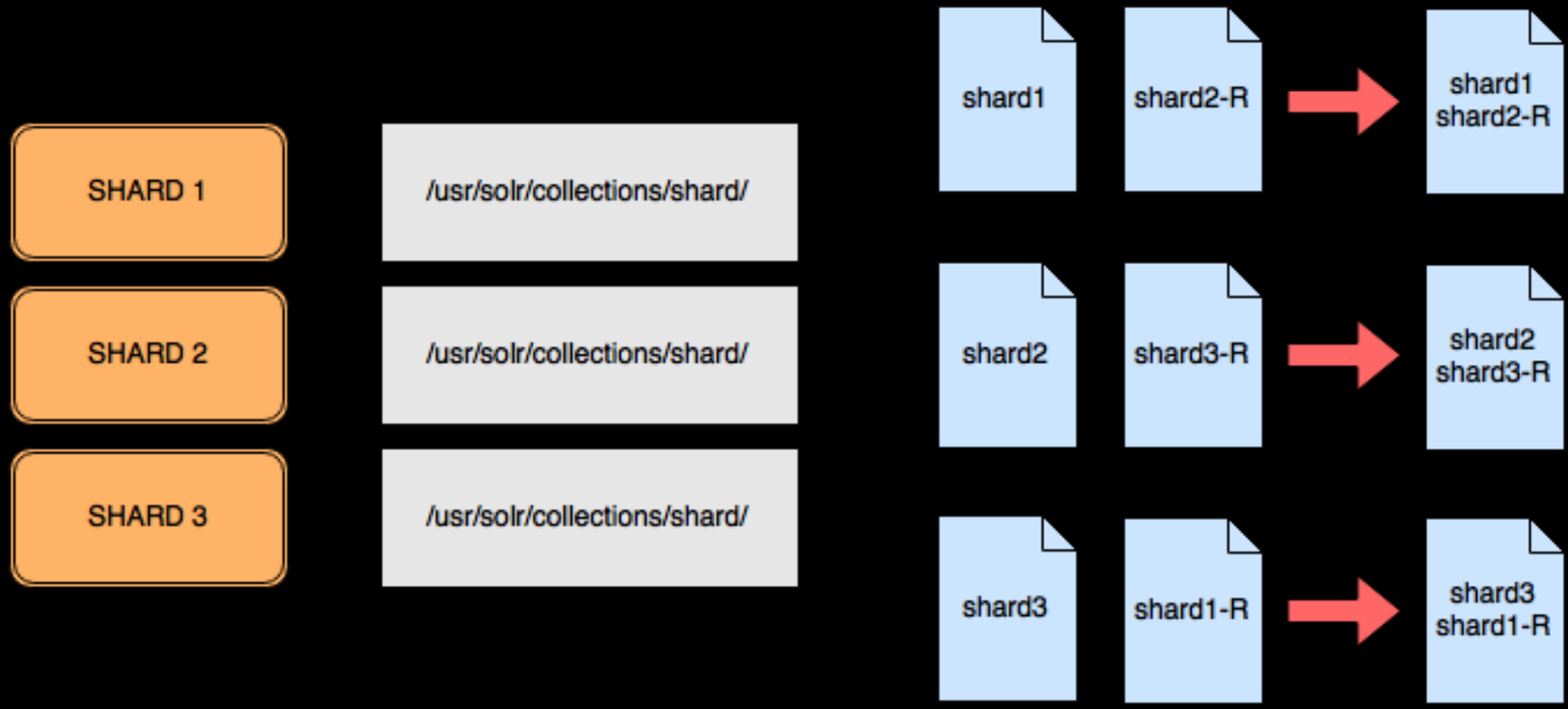
- **Symptom:** Misconfiguration led to Data duplication and crazy behaviours
- **Found:** We had less pages than numResults/ pageSize on search results
- **Problem:** Incorrect configuration on shards of Solr Cloud

Search result demo

```
{
 "response":{"numFound":5787345824,"start":0,"end":20,"docs":[
    {
     “id":"MA147LL/A", "name":"Apple 60 GB iPod”, "price":399.0,
    },
    {
     “id":"JKA14YU/A", "name":"Nexus 6 32 GB", "price":639.0,
    },
    ...]
}
```

```
{
 "response":{"numFound":5787345732,"start":20,"end":40,"docs":[
    {
     “id”:”XXX", "name":"....","price":###.0,
    },
    ...]
}
```

```
{
 "response":{"numFound":3482149317,"start":29000,"end":29020,"docs":[
    {
     "id":"YYY", "name":",,,,,", "price":###.0,
    },
    ...
```



Two different shard’s data stored on one file

cli configuration

- Configure command line to attack proper Cluster
- Configure consistently, different tools to same cluster
- Take care on adding new things, like installing a simple node cluster locally

#2: N block file 1 mapper

- **Symptom:** Humongous file takes only one mapper no matter which configuration I try
- **Found:** changing block size or numMapper params doesn't change behaviour
- **Problem:** trying to read a gzip file on a parallel way

OH F**K!!

- Gzip Files have to be read completely before uncompression
- That was obvious once you know it
- You need to change mindset to HDFS world
  - Prepare your data for Hadoop
  - Use lzo compresion (for example)
  - Know the effects of the params

#3: Refactor of code makes process slower

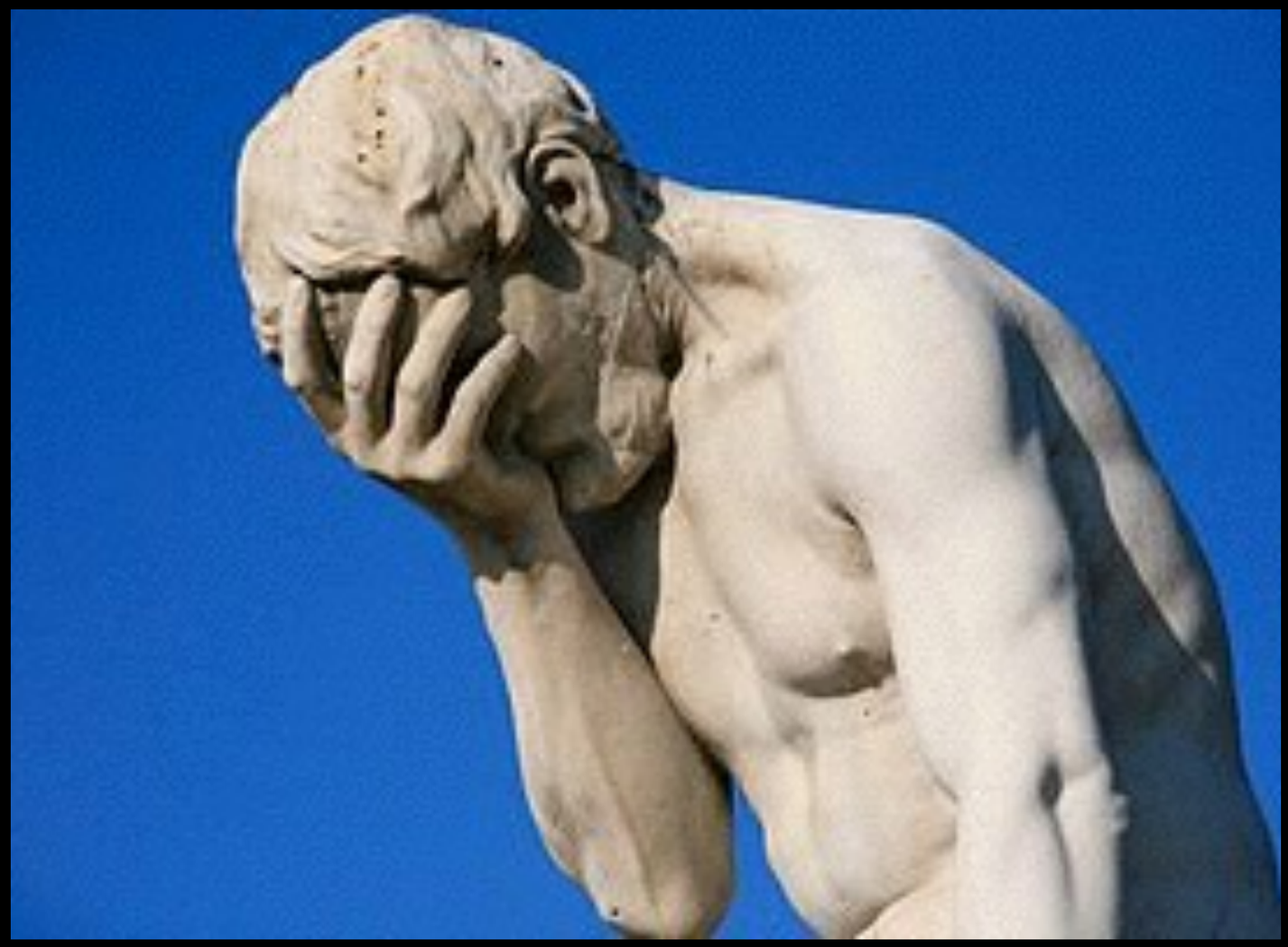
- **Symptom:** After refactoring code, process takes longer and writes more bytes
- **Found:** reviewing logs we found one line repeated #M times
- **Problem:** add “simple” log traces on a method

```
private void applyTaxes(Document balance, float taxPercentage) {
    Float grossAmount = Float.valueOf(balance.get(GROSS_AMOUNT));
    Float netAmount = grossAmount * (1 - taxPercentage);
    balance.add(new FloatDocValuesField(NET_AMOUNT, netAmount));
    logger.info("tax: " + taxPercentage + " / Gross: " + grossAmount + " / Net: " + netAmount);
}
```

Good intention while coding doesn’t mean doing good

40M times

```
tax: 0.12 / Gross: 1000 / Net: 880
tax: 0.12 / Gross: 500 / Net: 440
```



#4: Think about model changes

- **Symptom:** Exception after releasing a new code
- **Found:** A big red box on the console tells you something is wrong
- **Problem:** Refactoring code misplaced filed on the read/write fields functions breaking backward compatibility

```java
public class MyWritable implements Writable {
    // Some data
    private int fieldID;
    private boolean validField;
    private long fieldTimestamp;
    private String fieldName;

    public void write(DataOutput out) throws IOException {
        out.writeInt(fieldID);
        out.writeBoolean(validField);
        out.writeLong(fieldTimestamp);
        out.writeUTF(fieldName);
    }

    public void readFields(DataInput in) throws IOException {
        fieldID = in.readInt();
        validField = in.readBoolean();
        fieldTimestamp = in.readLong();
        fieldName = in.readUTF();
    }

}
```

```java
public class MyWritable implements Writable {
    // Some data
    private int fieldID;
    private String fieldName;
    private long fieldTimestamp;
    private boolean validField;

    public void write(DataOutput out) throws IOException {
        out.writeInt(fieldID);
        out.writeUTF(fieldName);
        out.writeLong(fieldTimestamp);
        out.writeBoolean(validField);
    }

    public void readFields(DataInput in) throws IOException {
        fieldID = in.readInt();
        fieldName = in.readUTF();
        fieldTimestamp = in.readLong();
        validField = in.readBoolean();
    }

}
```

Better field organisation, but how yesterday's data was stored?

Find Versioning alternatives

- First Field is a version field: *complicated to maintain*
- Create your own Protocol for serialization: *don't even try*
- Use Avro: *Solution for new attributes, changes on order, changes on platform, …**

* https://martin.kleppmann.com/2012/12/05/schema-evolution-in-avro-protocol-buffers-thrift.html

#5: Big Data everywhere

- **Symptom:** Bootstraping an App gets more time than executing it
- **Found:** Log reviewing
- **Problem:** Do you really need hadoop to process a 30 MBytes file??


MY PASSPORT
WD
My Passport
portable hard drive. disque dur portable.
1 TB To


SAY BIG DATA
ONE MORE TIME
memegenerator.ne

#6: Redoing things from scratch

- **Symptom:** Estimation on scaling the app to be distributed requires redoing everything
- **Found:** Product Owner's face is getting red
- **Problem:** Wrong design, business logic is embebed on the framework.

Best Practices

- Clean Code
- SOLID
- Software Craftmanship
- Delta Architecture
- Microservices
- …

#7: Over Design

- **Simptom:** Inability to move forward due to dependencies and technical debt
- **Found:** Every day you have a pain in the ass seen unused classes
- **Problem:** You added this very useful functionality on your code that you have never used.

Use Common Sense

- Think on the use you are going to give to this data/app
  - Make of the world a better place by parsing my apache logs
  - Understand your fellow Data Science colleagues' needs
  - Apply Machine Learning
  - Make Data Available for reporting or Visualisation

Take away #1

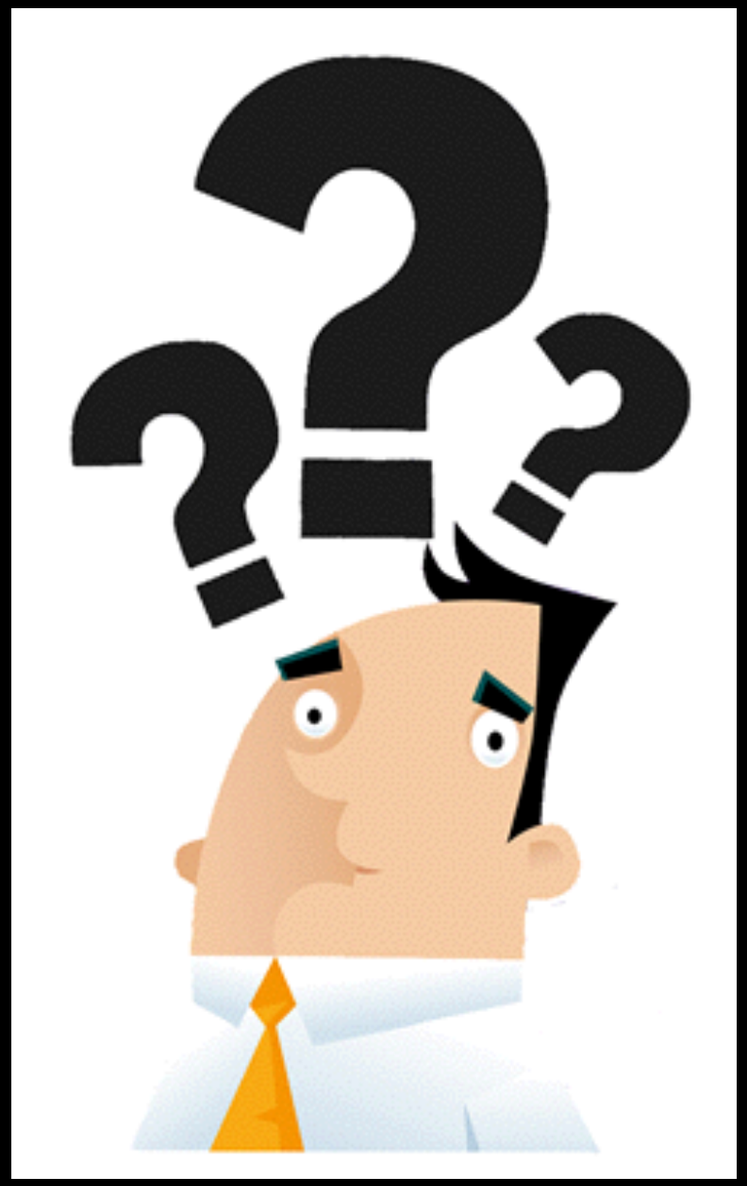
- Forget About Hadoop

Keep your business logic separated from the framework you use

Take away # 2

- Keep Hadoop in mind

Remember that you are processing distributed and at scale when implementing



Qeustions??

We are hiring

Data Engineers
DevOps
Mobile Developers
Senior Software Engineers

Feedback is welcome

iker@schibsted.es
@mitxino77