Python for Distributed Systems

PyData Madrid. April 8th 2016

Guillem Borrell i Nogueras

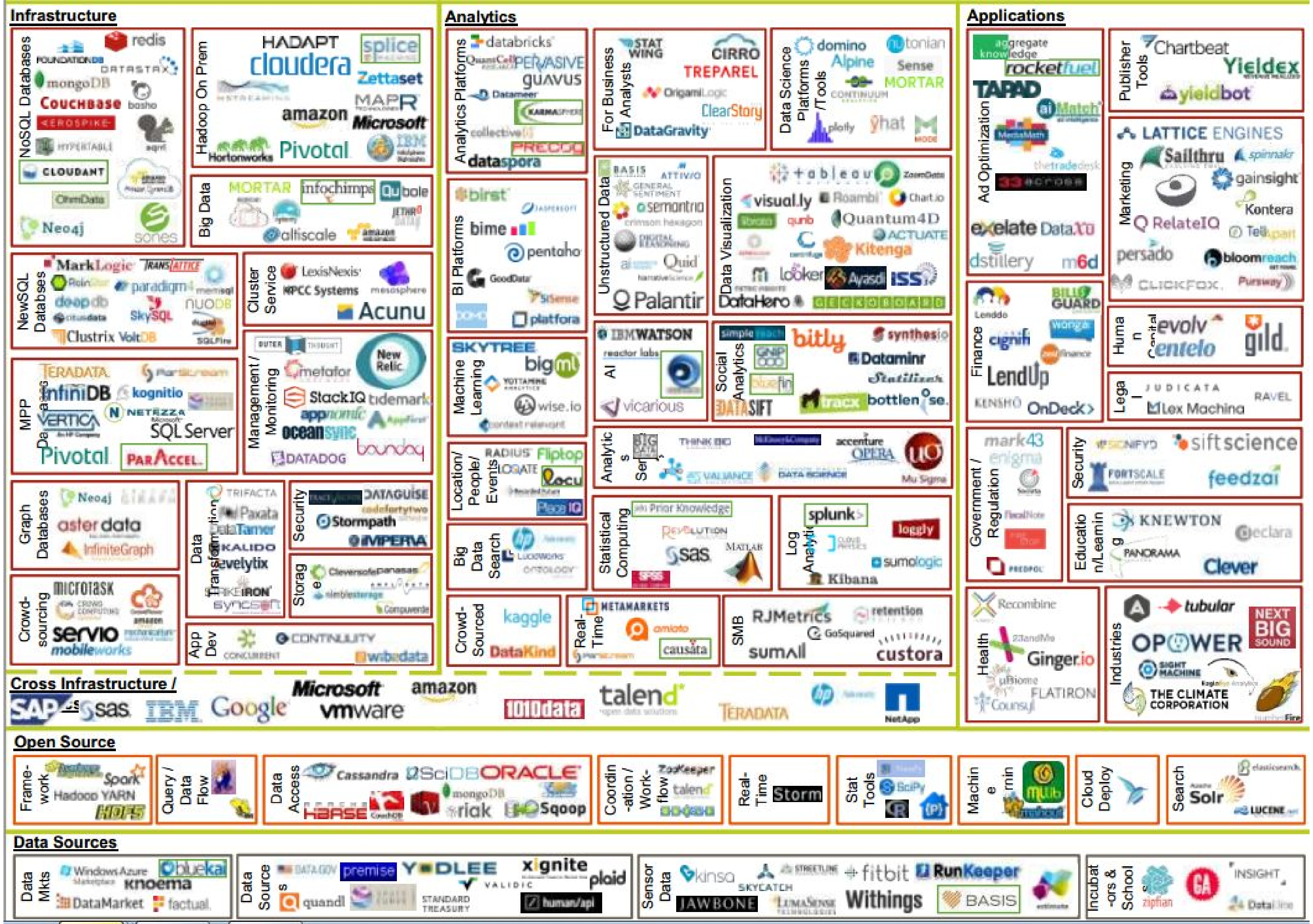
https://github.com/guillemborrell/pydata-madrid

Me

- Aerospace Engineer. PhD.
- I worked with supercomputers.
- I dealt with lots of data.
- In some cases I actually built the supercomputers I used.
- I build high-performance distributed tools for a living.
- @guillemborrell
- http://guillemborrell.es

High performance distributed systems


ZX Spectrum
BLUE
EDIT
1
DEF FN
RED
CAPS LOCK
2
FN
MAGENTA
TRUE VIDEO
3
LINE
GREEN
INV. VIDEO
4
OPEN #
CYAN
5
CLOSE #
YELLOW
6
MOVE
WHITE
7
ERASE
8
POINT
GRAPHICS
9
CAT
BLACK
DELETE
0
FORMAT
SIN
Q
PLOT
ASN
COS
W
DRAW
ACS
TAN
E
REM
ATN
INT
R
RUN
VERIFY
RND
T
RAND
MERGE
STR $
Y
RETURN
CHR $
U
IF
CODE
I
INPUT
IN
PEEK
O
POKE
OUT
TAB
P
PRINT
READ
A
NEW
RESTORE
S
SAVE
DATA
D
DIM
SGN
F
FOR
ABS
G
GOTO
SQR
H
GOSUB
CIRCLE
VAL
J
LOAD
VAL $
LEN
K
LIST
SCREEN $
USR
L
LET
ATTR
ENTER
CAPS SHIFT
LN
Z
COPY
BEEP
EXP
X
CLEAR
INK
L PRINT
C
CONT
PAPER
L LIST
V
CLS
FLASH
BIN
B
BORDER
BRIGHT
IN KEY $
N
NEXT
OVER
PI
M
PAUSE
INVERSE
SYMBOL SHIFT
BREAK
SPACE

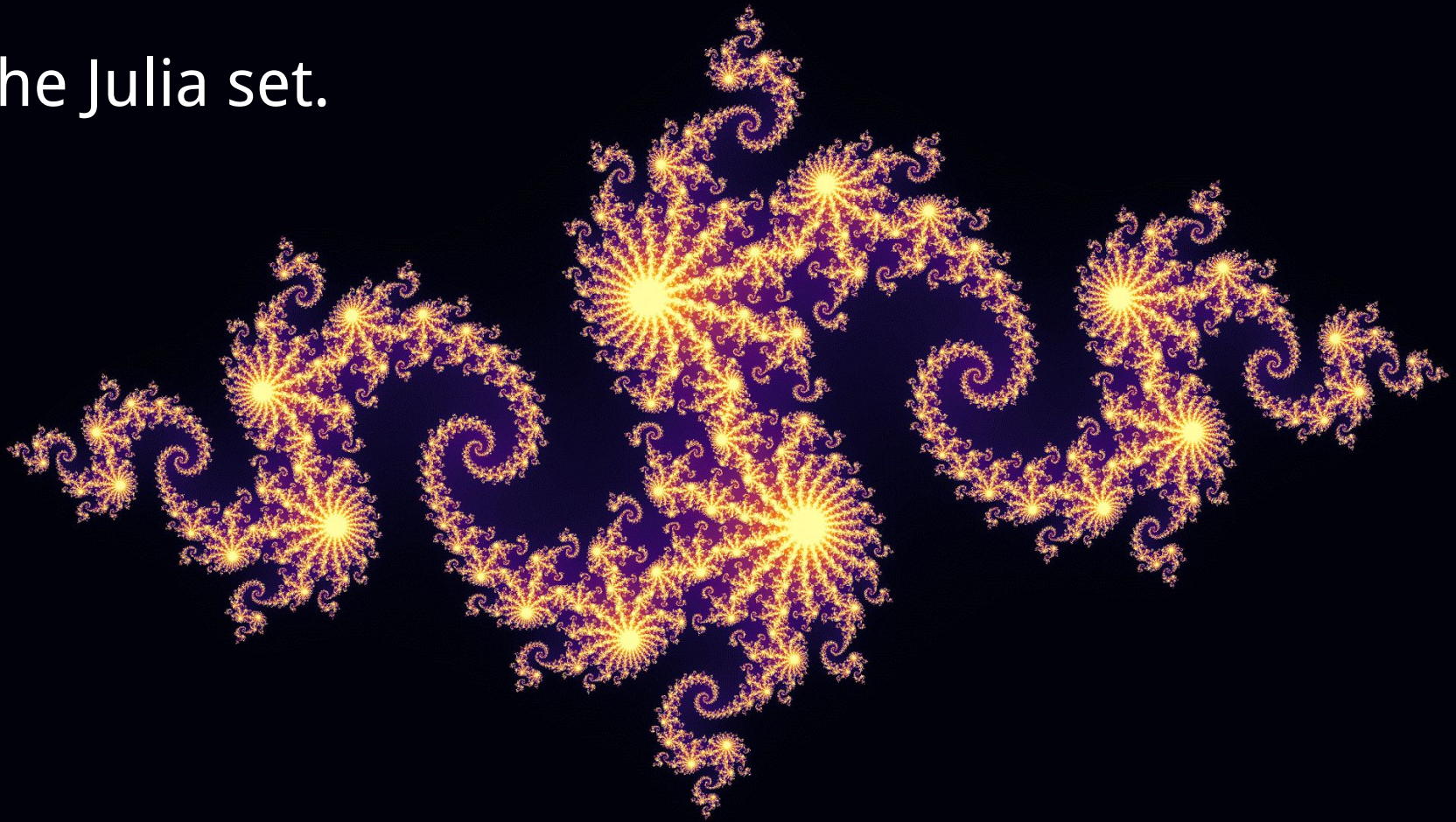


Infrastructure
NoSQL Databases
Hadoop On Prem
Big Data
NewSQL Databases
Cluster Service
MPP Databases
Management / Monitoring
Graph Databases
Data Transformation
Security
Crowd-sourcing
Storage
App Dev
Analytics
Analytics Platforms
For Business Analysts
Data Science Platforms /Tools
BI Platforms
Unstructured Data
Data Visualization
Machine Learning
AI
Social Analytics
Location/ People/ Events
Analytics Services
Big Data Search
Statistical Computing
Log Analytics
Crowd-Sourced
Real-Time
SMB
Applications
Ad Optimization
Publisher Tools
Marketing
Finance
Human Capital
Legal
Government / Regulation
Security
Education/Learning
Health
Industries
Cross Infrastructure / Analytics
Open Source
Framework
Query / Data Flow
Data Access
Coordination / Workflow
Real-Time
Stat Tools
Machine Learning
Cloud Deploy
Search
Data Sources
Data Mkts
Data Sources
Sensor Data
Incubators & Schools

Let’s start from ground up.
And let’s be ambitious.

The Unicorn idea.

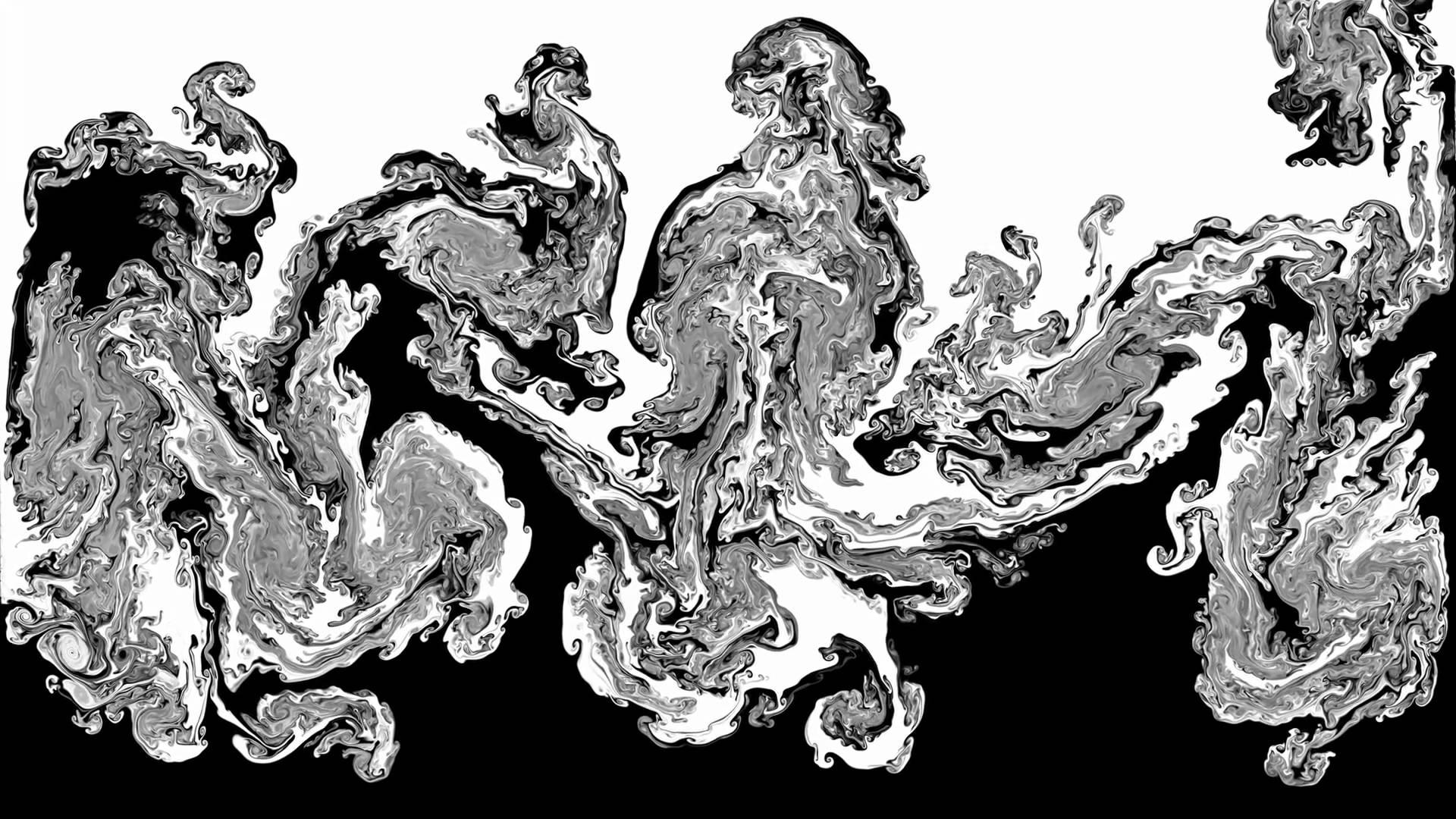
The Julia set.



Pierre Fatou
1878 - 1929

Gaston Julia
1893 - 1978

$$z_{n+1} = z_n^2 + c$$



Let’s make a Julia set.

Now we have an application

- Something that is useful, but takes a good amount of resources. Like CPU/IO intensive stuff.
- But applications are for cellphones and pirates!
- Services = $, €, ¥…
- We need a name, an elevator pitch, and a REST API to hire a frontend developer.
- We will make something that works in the Angel round.

Fractal.ly

Find beauty in complexity…
Web scale!

The REST API

Is it scalable?
Is it responsive?
Is it efficient?

Let’s make it a little more efficient.

Now we visit a business angel.
And we got money to rent servers!

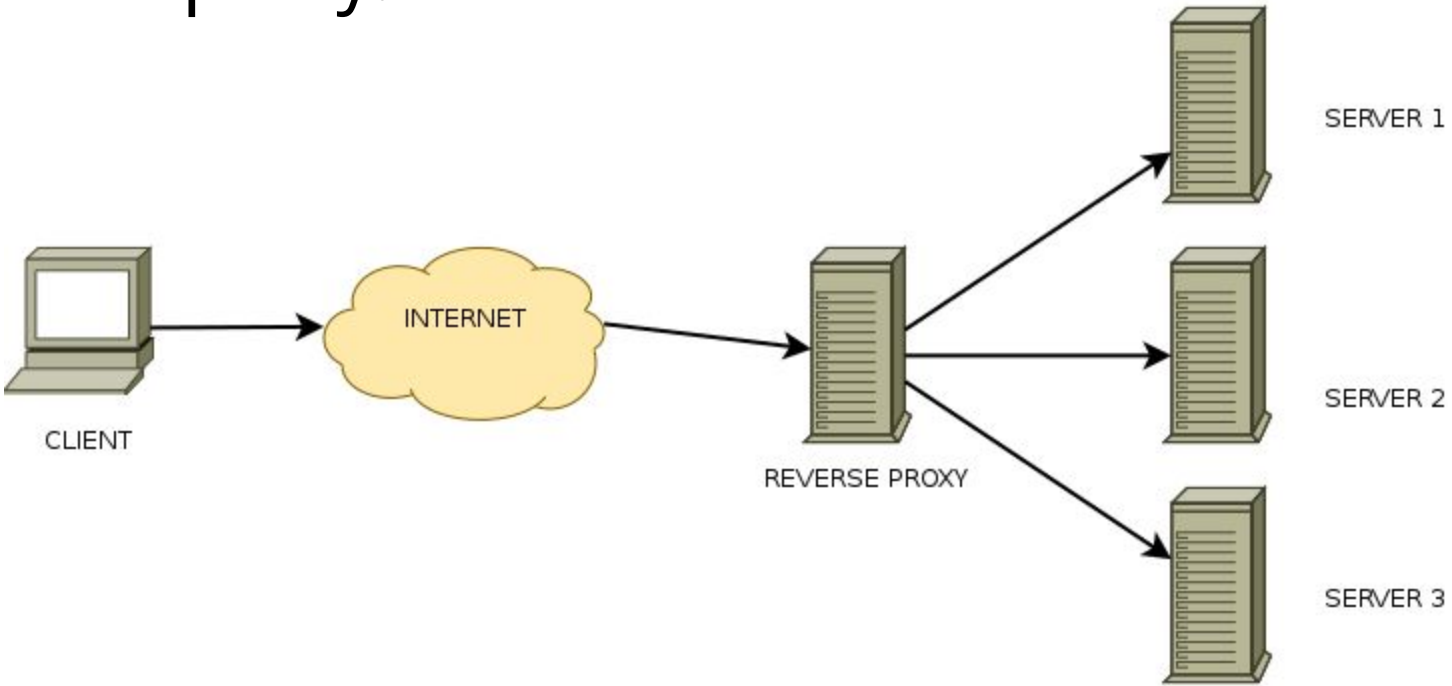
Let’s make it parallel.

Parallelization is about domain decomposition

- Split the problem in separated portions.
- The portions should share the minimum amount of information. (CAP)
- Exploit low-hanging fruits.

It is fine to deal with each request in parallel. We have to be pragmatic, it’s not about publishing a CS paper.

A reverse proxy.



Is it scalable?
Is it responsive?
Is it efficient?

We have our first users. They complain because the application is not responsive enough.

We hire a frontend developer.
And we make the application asynchronous.

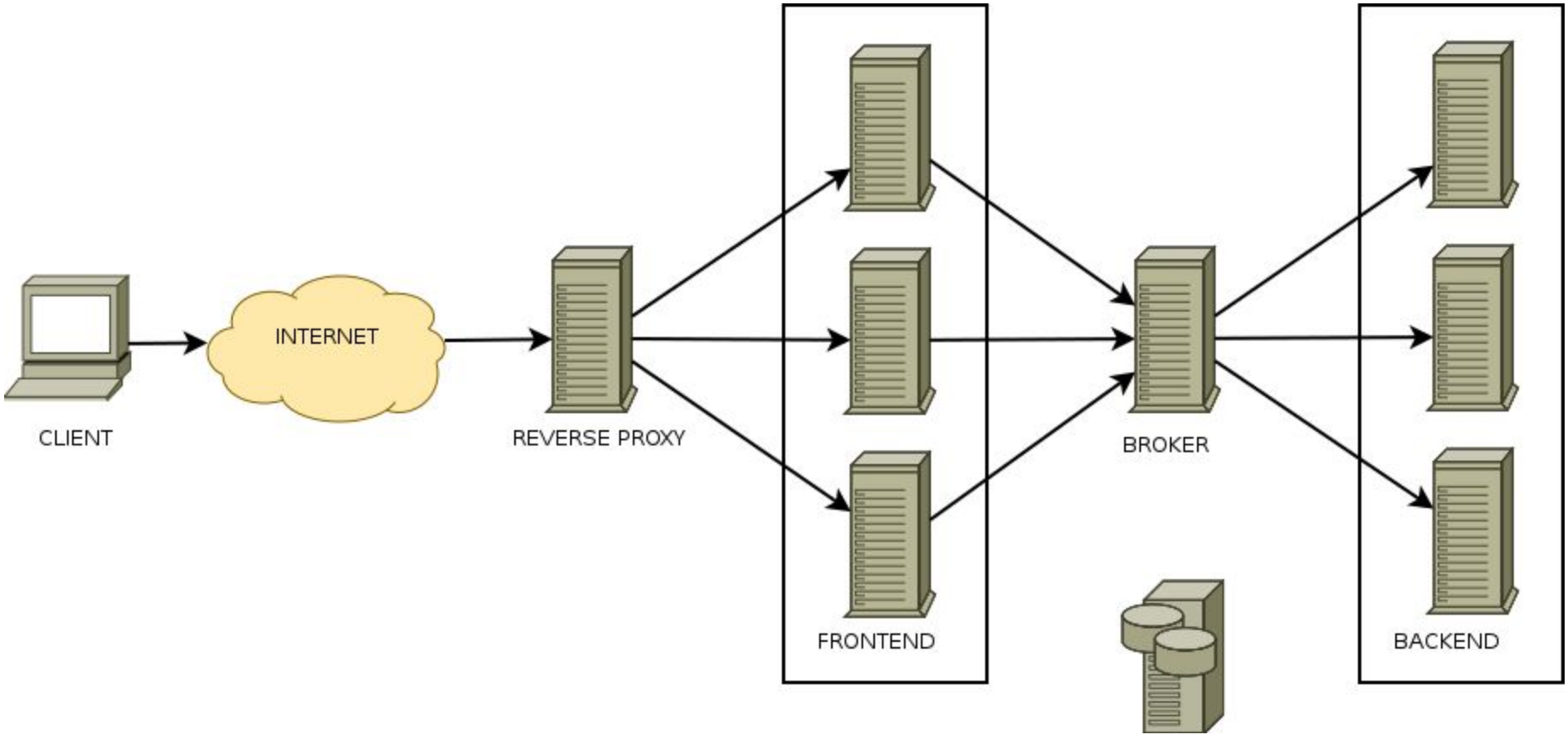
Polling is a good way to deal with asynchronicity.

But the application is not as responsive as we expected


GIL
HYATT
REGENCY
SANTA CLARA

The GIL is there because the alternatives are quite frightening.

Split I/O intensive process from CPU intensive processes.


CLIENT
INTERNET
REVERSE PROXY
FRONTEND
BROKER
BACKEND

We need message passing.

A message queue

A serialization format

Is it scalable?
Is it responsive?
Is it efficient?


julia

We now have a parallel, distributed, multi-threaded, multi-language, responsive and efficient service to obtain julia sets WEB SCALE!


amc

Questions?