How to serve 2500 requests per second

SAPO text Ads

Miguel Mascarenhas Filipe @ codebits, November 2010

Who am I ?

Team lead of
Anúncios Sapo (text ads) team

Team of 5 (+ me) *software engineers*,
designing & developing the
text ads platform

How to serve *VS* How we serve

Are we a role-model?

Are there recipes ?

Should there be a how to?

Summary

- Full System Overview
- Serving Text Ads
- Speed & Latency
- Scaling on the Front
- Backend & backend Services
- Availability, Reliability & Fault Tolerance
- Scalability Issues
- Monitoring & Logging
- Programming Languages & Technologies

Full System Overview


Serving
Text Ads

Concepts

- Pay Per Click Business Model
- CPC - Cost Per Click
- QPS – Queries Per Second
- CTR - Click Through Rate
  (clicks / impressions)

Serving text-ads ..

Major *features*:

- choose & serve ads
- register requests, impressions, clicks, conversions
- maintain user budget up to date
- Quickly reflect changes in ad pool

Serving text-ads ...

```
elect_get_ads() {
   if( words )
      ads = get_ads_keywords()
   else {
      if (crawled_site)
         ads = get_ads_crawler()
      else
         ads = get_ads_site_keywords()
   }
   site_ads = get_ads_site_targeting()
   merge_ads(ads,site_ads)
}
```

Serving text-ads ...

Election of ads:

- requires index
  ads['word'] -> [ad1, ad2, ad3..]
- ads ordered by:
  'score' -> f(CTR, CPC, quality)
- Auction based on
  Generalized second-price Auction

Serving text-ads..

Other *essential* features:

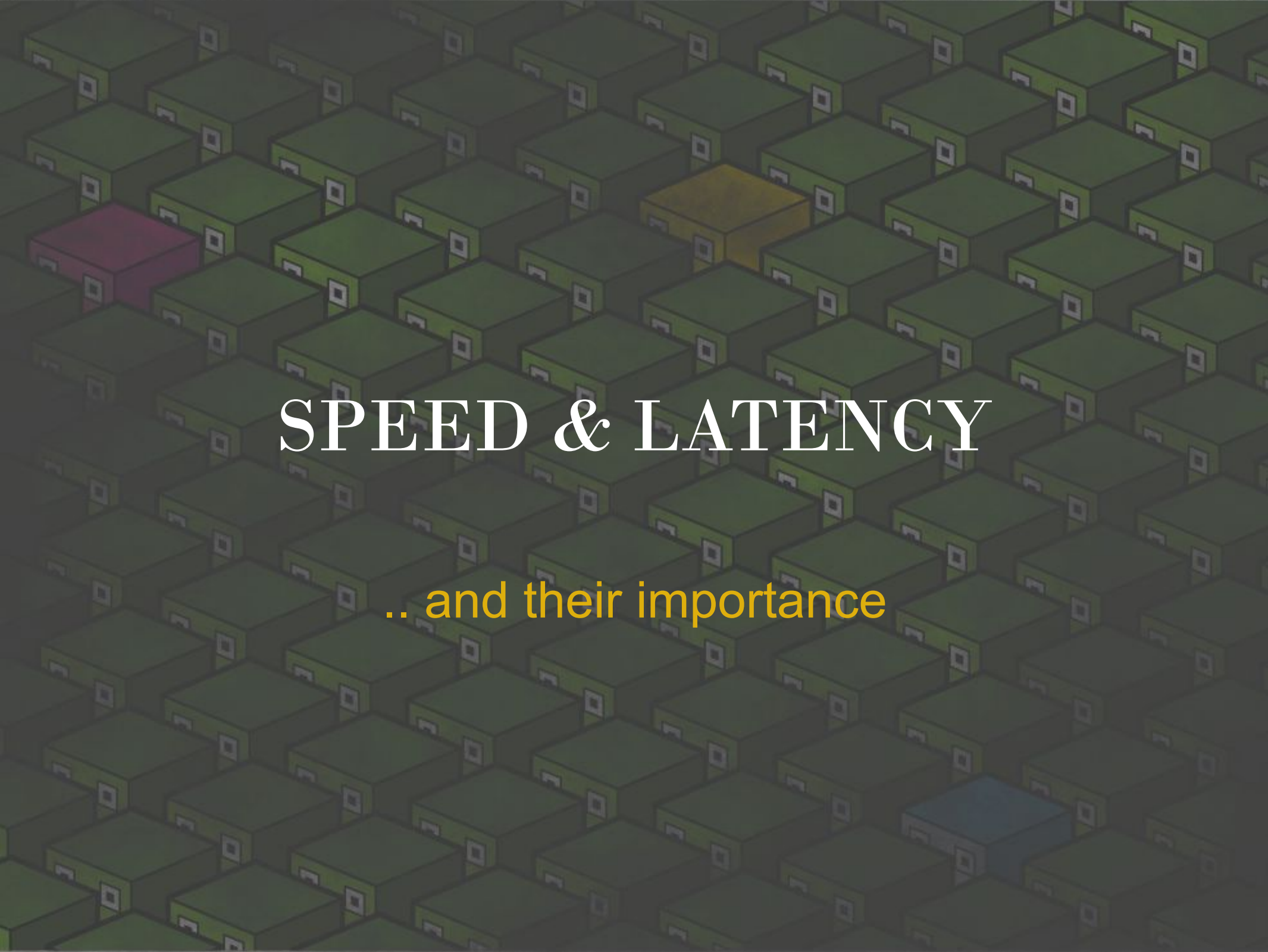
contextualization of webpages/sites

blacklisting of ads per site

reporting information

scoring quality of ads

anti-fraud systems/fraud detectors


SPEED & LATENCY
.. and their importance

Speed

Speed is .. ?

Queries per second ?

Sequential or Concurrently (Throughput) ?

Speed

Speed is ..

Queries per second.

Sequential or **Concurrently** (Throughput) ?

Throughput is obviously what matters in this case..

Latency

Low latency is required:

Search pages (BING,SAPO,..) have to:

search ads (that's us!)

search results

and merge results together.

«***ads added last***» - site developers put ad-request code at the end of the page (last thing to load, usually)

Latency

Without good latency
ads are slow to appear and
users have **moved on**...

Latency

Slow ads → Low CTR → BAD!

Latency has a **BIG** impact on **REVENUE**.

Latency Service Level Agreement

99.9% of reqs under:

150 milliseconds

Average response time is:

20 milliseconds

Never take more than 1 second.

serve blank ads in that case

How to keep low Latency ?

Pre-computing everything is essential

Fast contextualization lookup

Handle lack of information gracefully
(turning essential into optional)

How to keep low Latency ?

Decouple (and postpone) everything
that isn't essential to serve ads

.. such as DB *writes* & other *side effects* of
serving ads.

Fast word lookups - LiveDB

Fast word/site lookup(inverted index of ads)

- cache it in local **RAM** (memcached)
- 'persistent' backing store is **RAM**

Fast word lookups - LiveDB

Offline creation of index:

ads['word'] -> [ad1, ad2, ad3, ad4, ...]

Lots of details, need to compute additional information for each tuple: (word, ad, CPC):

CTR, Evaluation Score

Fast word lookups - LiveDB

Since *source* data was already in MySQL, we bet on MySQL for:

- fast 'inverted index' creation
  (by using Stored procedures & replication)
- fast index lookup based on the 'fame' of MySQL speed in simple workloads
- Replication for free using MySQL's master-slave replication

Fast word lookups - LiveDB

Workload is almost read-only.
(in fact, we can make it read-only with some tricks)

Storage engines **FAST** for read-only workloads:

MySQL **MEMORY**

MySQL **MyISAM**

MySQL MEMORY

Extremely fast lookup.
data is guaranteed to be in RAM (or in swap..)

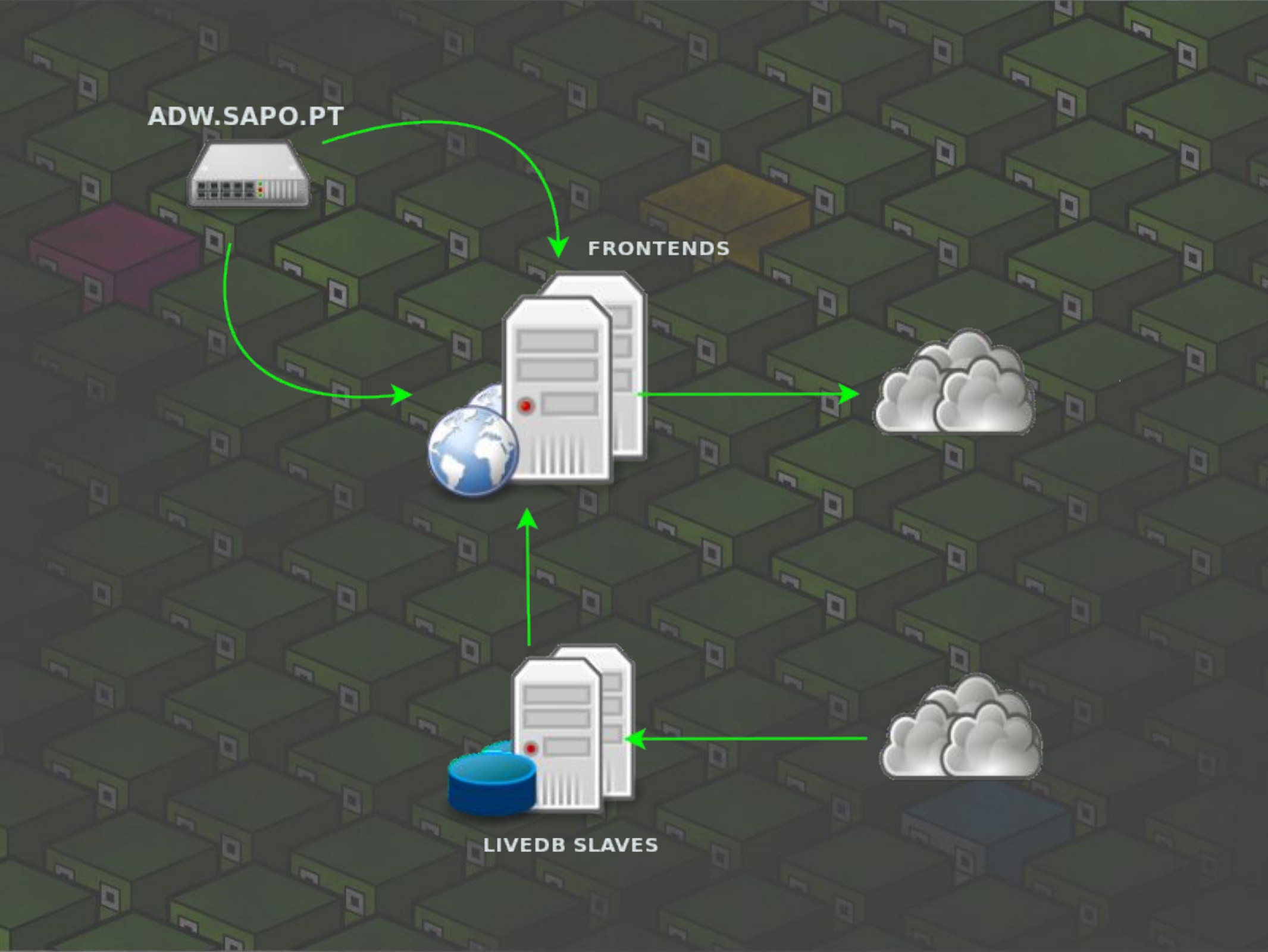
Benchmarked MySQL Memory engine:
.. avg response time was around 10-20msecs,
..within our needs!

Constraints:

MySQL MyISAM

.. After months in production use, MEMORY engine proved **problematic**..

Evaluated MyISAM, did benchmarks: same speed, lower standard deviation.


ADW.SAPO.PT
FRONTENDS
LIVEDB SLAVES

Speed!

avg time is 20msecs = 50 QPS

but... it's totally parallel workload.
And most of the time is **IOwait** on *index lookup*.

1 server **cpu** can do ~6x this: ~300 QPS

Scaling on the Front..

Se scale horizontally because:

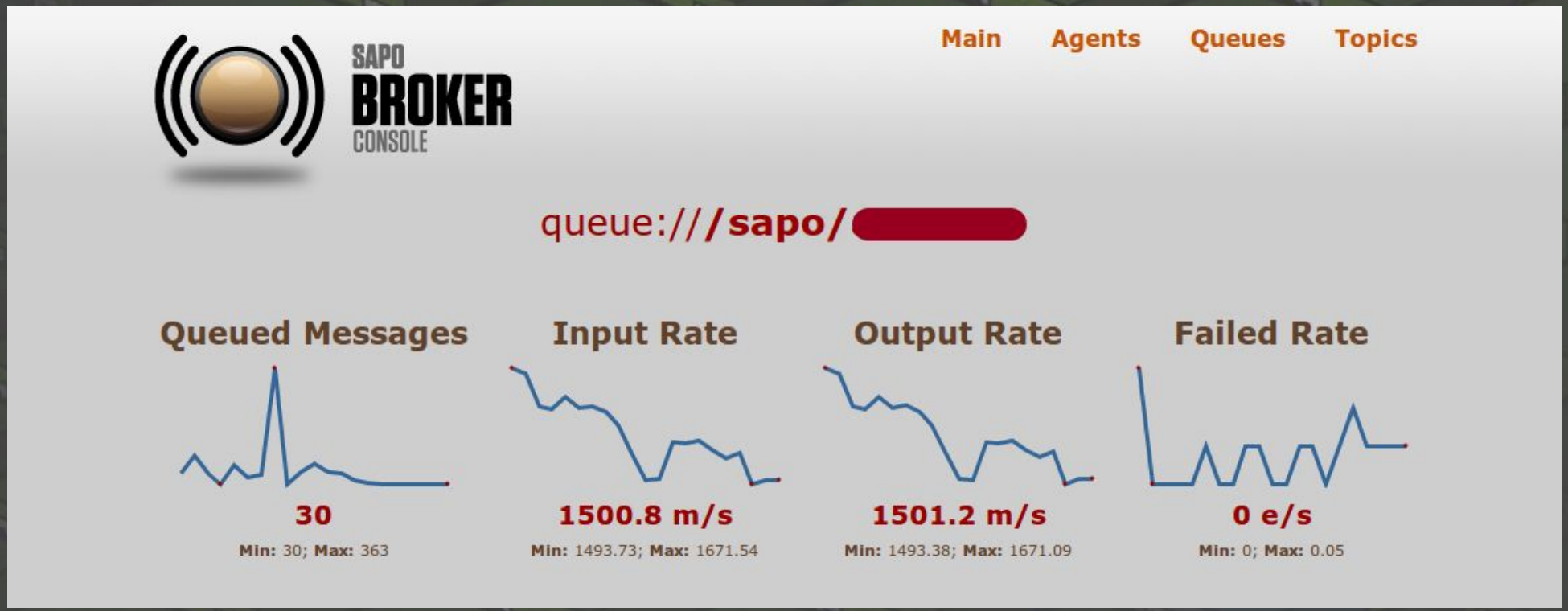
- We can add more Frontends
  to handle more QPS

- We can add more LiveDB slaves
  to handle more SQL Queries

Backend

Message queueing system:

SAPO BROKER



Backend Services

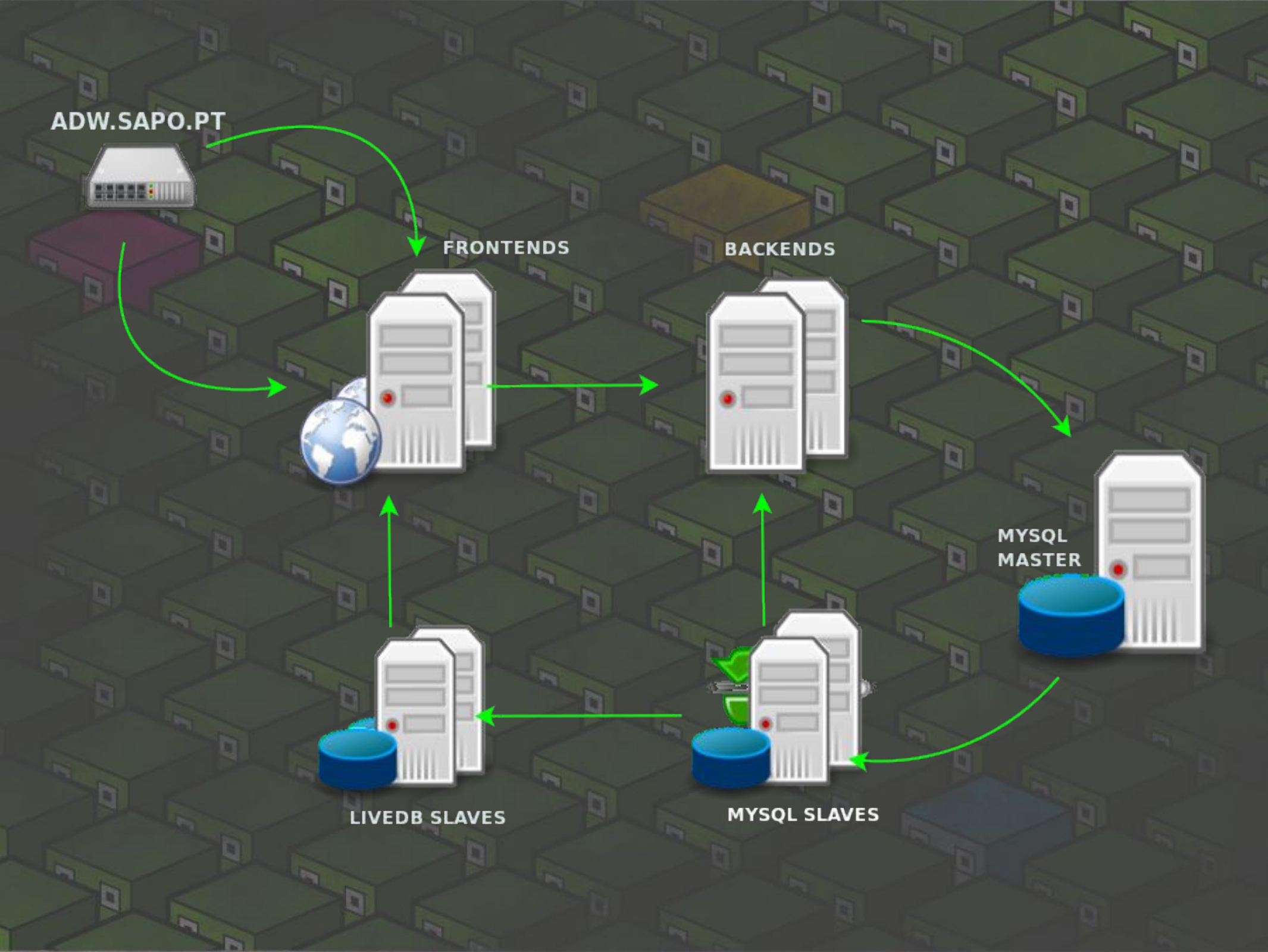
- 'compact' & apply operations to the DBMS
- runs anti-fraud system
- runs contextualizer & crawler
- runs semantic analyser
- runs reports & suggestions system

Building the LiveDB

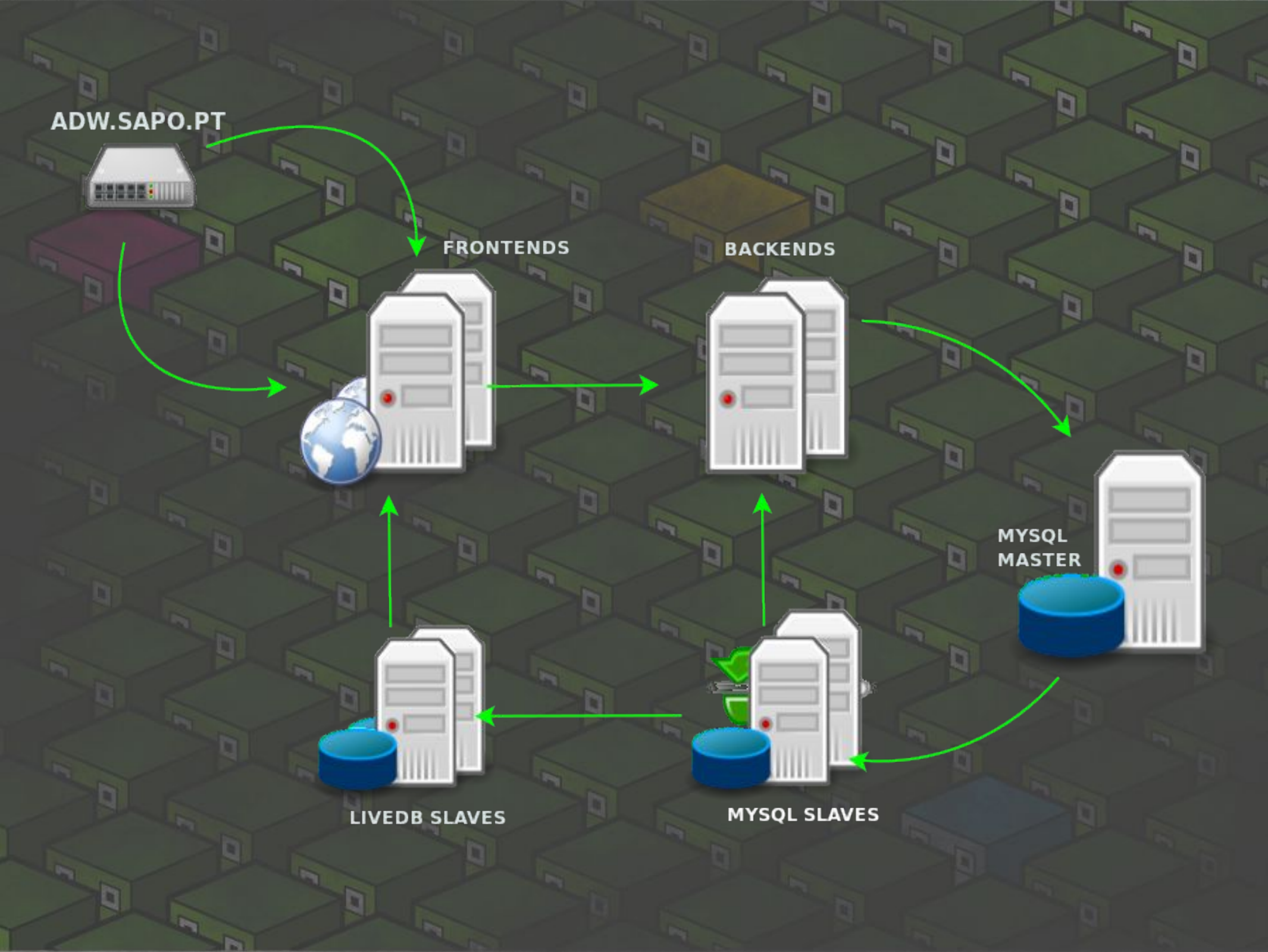
MySQL is the DBMS
MySQL is the non-ACID LiveDB.

again .. MySQL replication for fault tolerance, redudancy and balancing of LiveDB.

Python & Stored Procedures created LiveDB in a MYSQL DBMS slave,
mysql replication handles distribution to
read-only slaves (that the frontends query).


ADW.SAPO.PT
FRONTENDS
BACKENDS
MYSQL
MASTER
LIVEDB SLAVES
MYSQL SLAVES


Availability & Reliability
(no downtime please..)


ADW.SAPO.PT
FRONTENDS
BACKENDS
MYSQL
MASTER
LIVEDB SLAVES
MYSQL SLAVES

Reliability & Fault Tolerance

Almost every service/component is redundant.

Currently there are only 2 single points of failure:

- Master DB server
- Master LiveDB server

And even if BOTH FAIL, we keep on serving ads...

Reliability & Fault Tolerance

Failure in Master LiveDB server:

- We have a hot spare,
- Can change master in aprox 5 to 10 minutes

Failure in Master DB:

- Data starts to pile up on Backend services
- Backoffices are unable to operate (no new ads)
- if failure lasts a long time:
  - we might serve ads without budget
- Electing a new Master is performed manually

Has happened before, no ad serving downtime.

Scalability Issues

We can scale horizontally in all but two components currently:

- Master DBMS MySQL server
  (but we are far from saturating it..)
  we currently don't plan to 'solve' this

- Master LiveDB server

...

Scalability Issues

- Building LiveDB doesn't scale
  - We build a full new LiveDB everytime
  - It isn't distributed nor is it easily made parallel
  - Time is proportional to nº of active Bids

LiveDB should be updated only with recent changes in ad pool.

Impossible to do with current main DB data model and with current LiveDB.

We are currently investing heavily on a solution to this, Codenamed:

LiveCouchDB


Monitoring & Logging
(is everything okay?)

Monitoring & Logging

Bad things happen:

Log it, deal with it...

We need to know about it:

monitor logs

trigger alarm if errors on log..

Monitoring & Alarmistics

network failures

reconnect with exponential backoff

log error

trigger alarm ?

Monitoring & Alarmistics

machine failures

replication & redundancy
save state to disk

Monitoring & Alarmistics

software bugs..

bad (or lack of) data

radio silence

log error

trigger alarm


Programming Languages
.. and software used

Programming Languages

Python (backend)
Perl (frontend code)

C (1 app only)
Java (broker & reporting)

PHP (backoffices)

SQL
Javascript

Technologies used

Linux 

memcached

MySQL

squid

nginx

Currently Evaluating

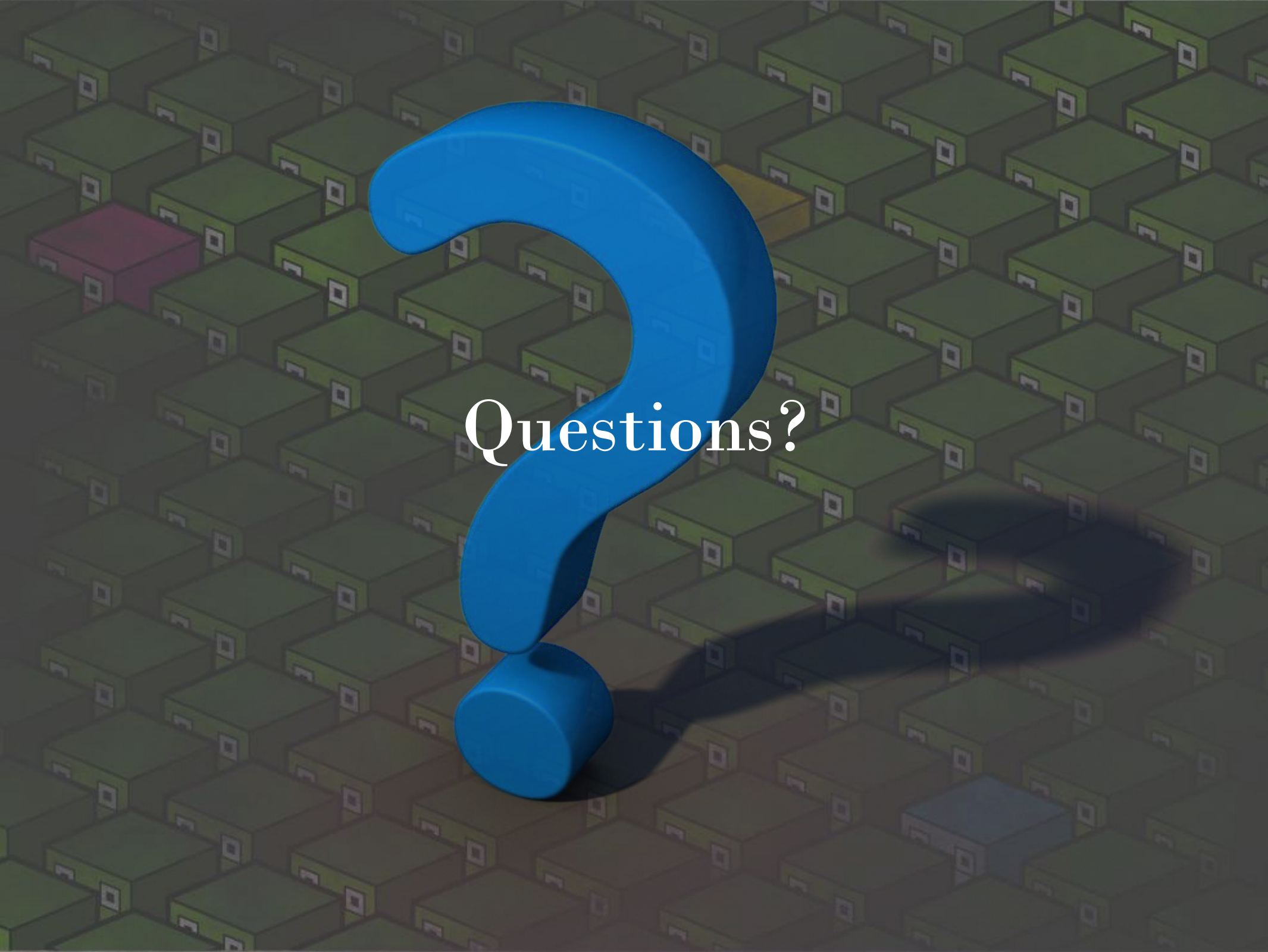
Languages:

Hadoop PIG

Technologies:

Hadoop

CouchDB


Questions?