The 8th DUW

Development plans

Federico Stagni

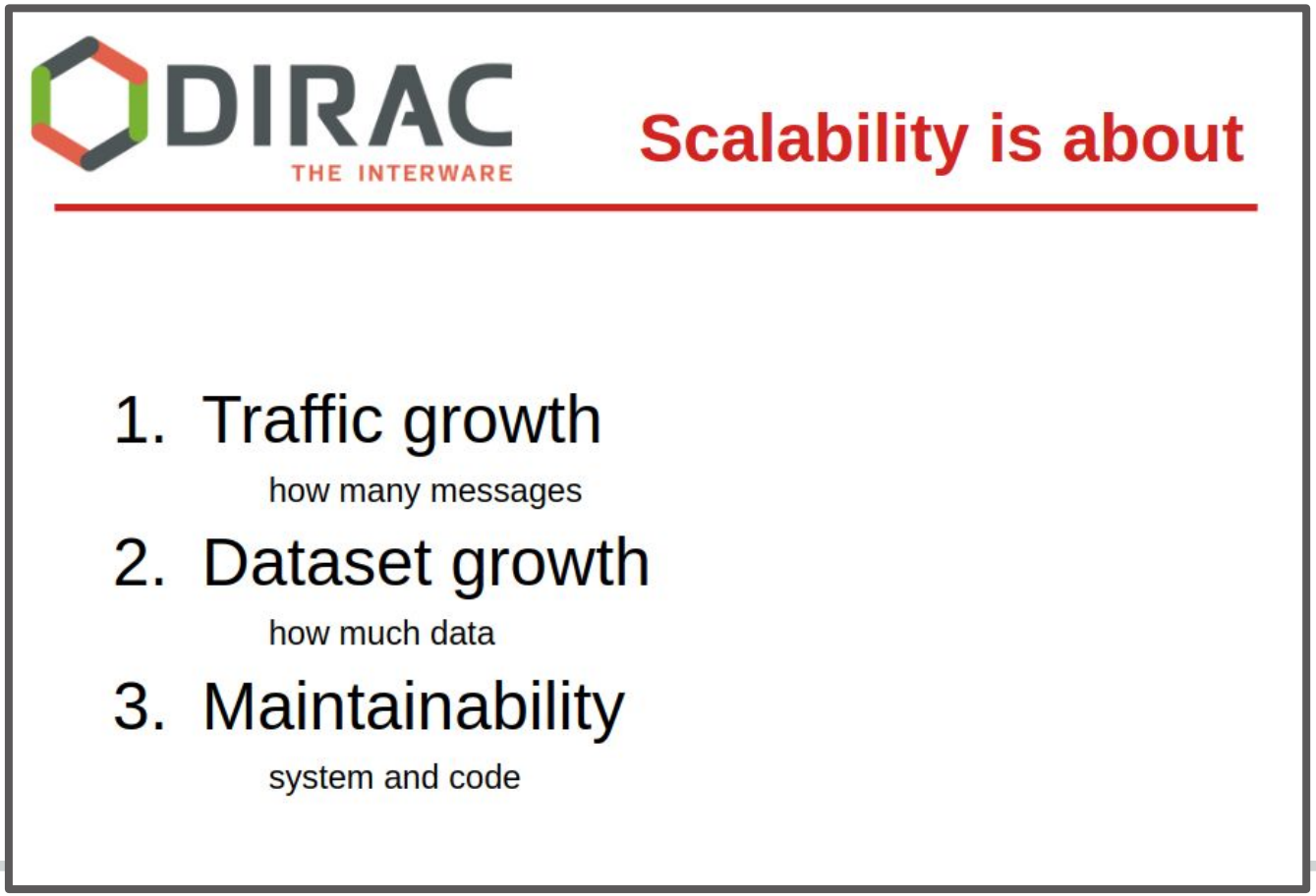
I repeat myself one more time...

Whatever we do, now and in the future:

- Need to keep a running system working, with continuity
  - whatever we decide, DIRAC won't impose you to deploy a parallel DIRAC
- Little manpower
  - Need to focus on important aspects

2 years ago...

We reckoned that DIRAC evolutions would be mainly about targeting the **scalability**:



1. Traffic growth

DIRAC architecture, and framework

Core and Framework

The DIRAC Core and Framework has been developed ~10 years ago, and now Lots of stuff in the DIRAC framework is available/maintained elsewhere

- this is already technology
  - is it worth/better than what we have now?
    - ***partly, yes!***
  - gLogger → python logging [DONE]
    - and plugins on the shelf!
  - pyGSI → M2Crypto [IN PROGRESS… STOPPED?]
  - dips → https [STARTED]
    - see later

Requirements on Traffic

Assumption:

increase of one order of magnitude

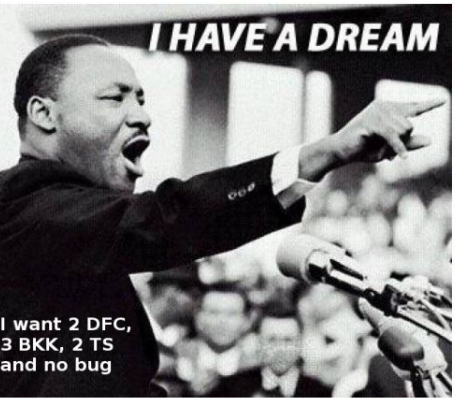
- Services: **~OK** if lots more services
  - → that "~" is meaningful!
- Agents: **KO**

DIRAC
THE INTERWARE

Scalability in mind

- **HW** scalability: more (micro)services
  - Supposing a good load balancer, this is not necessarily bad
    - self-tuning system?
  - Enter in the beautiful world of the ***orchestrators***...

I have a dream



Let “something”
run it “somewhere”
for you

Chris, 1 year ago...

What I miss in DIRAC

- **Respect the TLS RFC !**
  - Requires M2Crypto
- **Centralized logging**
  - Alexandre (ISIMA student) working on it
- **Less stateful agent**
  - Graceful stop seems complicated in Mesos
  - MQ and consumers schema ?

DIRAC
THE INTERWARE

Limitations of agents

For today's implementation:

- Parallelization is hard
- Multiple instances may not be possible
  - Lots of work anyway

→ NOT scalable

Message Queues

- MQs are in DIRAC
  - For failover purposes
  - Consumers as DIRAC components → RFC
- Push, not pull
- We can replace several agents with Consumers
  - and also (especially?) executors
  - Agents, executors, consumers as a single component?
  - … what about trying with this guy?
    - http://python-rq.org/
    - a nice project...

RQ: Redis Queue

DIRAC
THE INTERWARE

2. Dataset growth

Are we OK with MySQL?

DIRAC
THE INTERWARE

...yes!

- Shall we question MySQL?
  - Recently, new players came on the market (cockroachDB, crate.io, clustrix are just some name)
  - IMHO it's (way) too early to start thinking about investing time in exploring these solutions
    - which probably means that we are NOT questioning MySQL as RDBMS
- But we have introduced ElasticSearch, and *one day* it will be mandatory

DIRAC
THE INTERWARE

3. Maintainability

There’s no scalability if the system, and the software, are not:

- easily maintainable
- easily adoptable, and interoperable

DIRAC
THE INTERWARE

dips:// → https://

- Use HTTPS to allow usage of standard libraries
  - HTTPS is widely used
    - Big community
      - Lower risk of security breach
      - Easier to get help
      - Easier to understand what the protocol really do underneath
    - Many conventions already exists to send datas (like *JSON* or *multipart/form-data*)
    - Frameworks already exists in python 2&3 for server-side (Tornado) and client side (requests)
  - For developers it may be easier to modify the code

DIRAC
THE INTERWARE

dips:// → https://

- There is some work to do:
  - Integration with all DIRAC components
  - How to migrate services ? Clients ?
- Actually some things already works…
  - Secure transport (https)
  - AuthManager (partial) integration
  - Some performance test are written
- And others are not working yet…
  - Did not work with proxies certificates for now (problems to read all certificate chain)
  - For now you have to write diset service AND https service (no retro-compatibility)

dirac-install ++

- More flexible dirac-install:
  - Allow to deploy a non released code from:
    - Code repository
    - Local file system
    - Web server
  - Install not official:
    - LCG bundle
    - External
- Status: development has started
- Plan: use it in v6r21

DIRAC
THE INTERWARE

Python 3

- … yes, one day
  - not tomorrow
- we have been polishing the code for long time now
  - so, 2to3 won’t explode
- wide, deep, testing is fundamental

DIRAC
THE INTERWARE

to conclude

- Constant evolution
  - several developments started, many we'd like to start
    - interested? have a student? just get in touch!
- Test, test, test
  - the certification process is a daily work
  - the most automated, the best
    - we use Travis, Jenkins, gitlab-CI...
    - and the certification setup
      - want to learn? come to the dev hackhathon tomorrow
  - we'll introduce performance testing

DIRAC
THE INTERWARE

Questions

?

BACKUP Slides

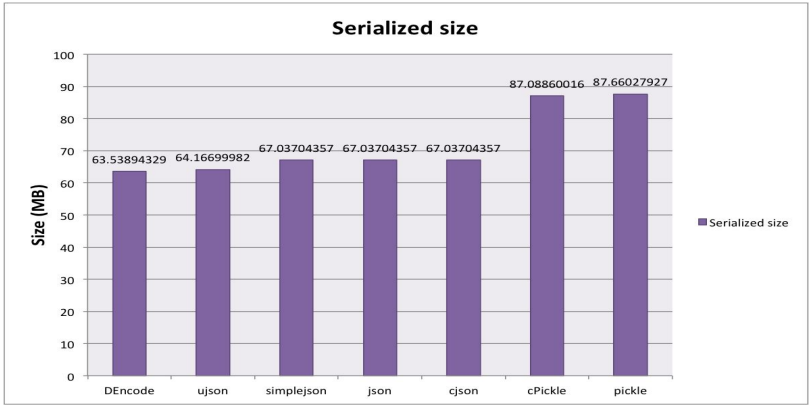
CS

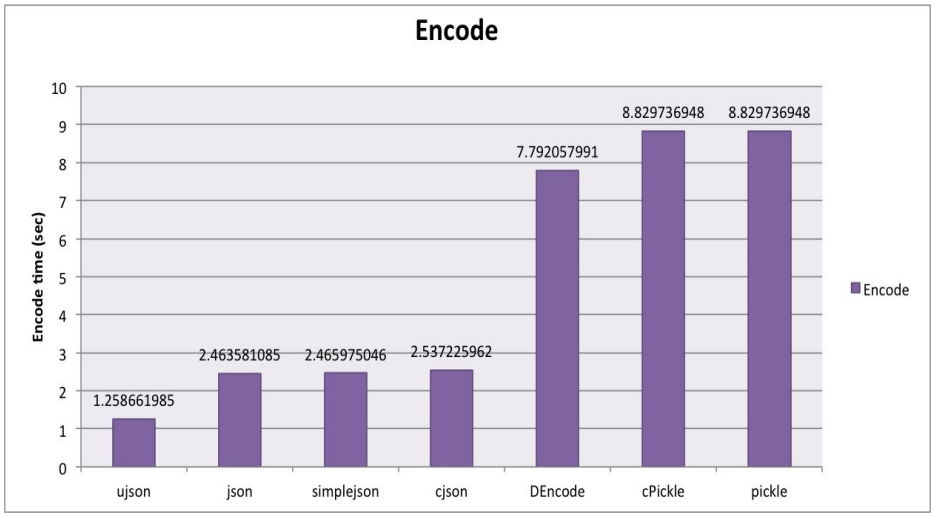
- Are we hitting it too much?
  - every RPC call goes through it
- Memory cache?
  - object cache?
- Database?
  - Redis, MemcacheDB, pickleDB, ...?
  -

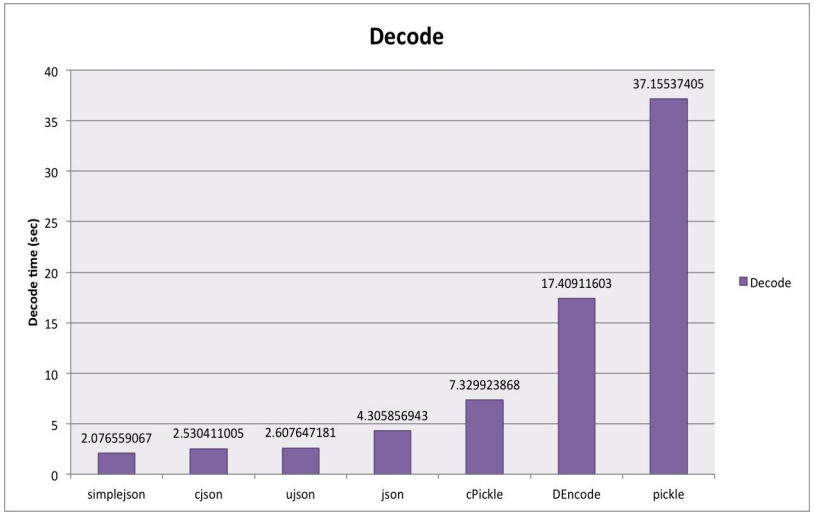
Encoding/decoding

What if we change DEncode?

Test: 178k files and their metadata







Queues, consumers, executors

- The DIRAC executor framework was an attempt of a scalable queueing system
- With an MQ system the concept of “executor” stays
- A consumer can spawn a (process)pool of executors
  - and we can have several running
- Agents can become a way to trigger executors at regular times
  - ...does it remind you the RequestExecutingAgent?
    - but you can have several “triggers” -- consumers