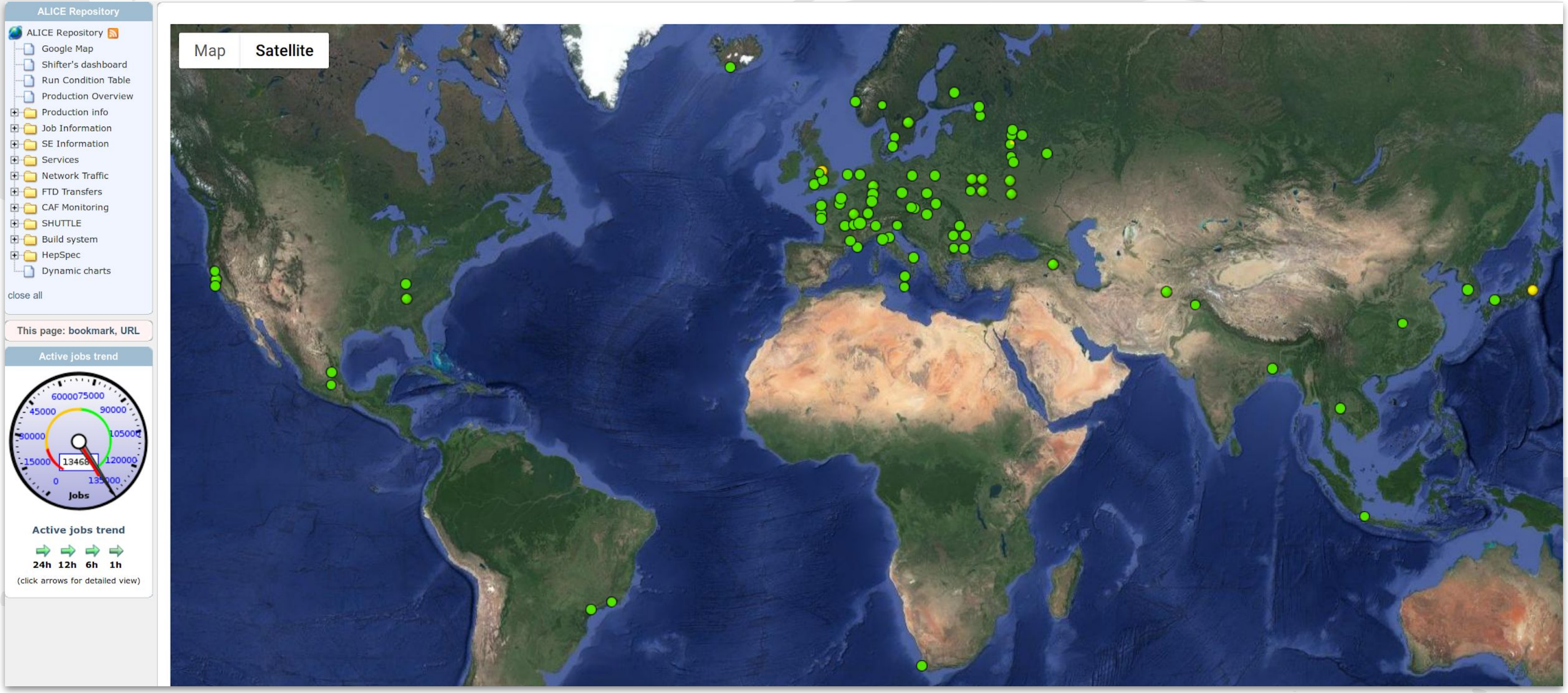
ALICE and HPC

costin.grigoras@cern.ch

ALICE Overview



ALICE Grid resources

Fully federated CPU and Storage

Up to 160000 concurrent jobs @ 80 sites

- Very heterogeneous resources

115PB of data @ 72 storage elements

- 50% of the volume is raw data, on tapes

CPU usage

70% used for MC prod

CPU eff > 90%

20% data analysis

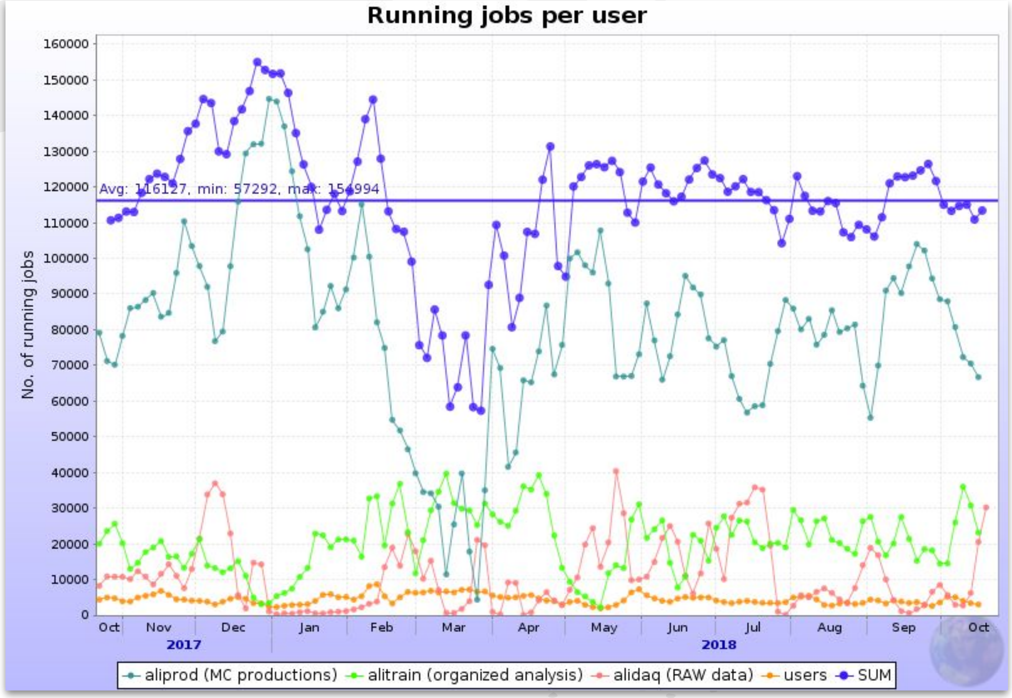
CPU eff ~50%, IO intensive

Individual users and organized

10% RAW data reco.

CPU eff ~80%

Only on T0/T1s



Computing model

- Anonymous jobs are scheduled on all sites
- At run time they get a job matching the slot
  - Input data on the nearby storage
  - Or no input data (Monte Carlo)
- Uploading output to SEs with free space
  - Might not be local (especially if >1 copy requested)

Many built-in assumptions

All of the below is a **standard GRID practice**, we did not invent it to bug the HPC folk

Some common HPC limitations -in regard to our use case of them- discovered the hard way in our attempt of using *Titan*

So, some of the assumptions

1. Point of presence per site (**VoBox**)

   Interface with the local WMS (batch or gateway)

   Local monitoring collector & topology discovery

2. Serial, independent jobs

   1 CPU core slot, performance within the Grid RMS

   At least **2GB** of RAM + **3.5GB** swap

more assumptions

3. **Local disk** scratch space - **10GB**

   For job intermediate files

   NFS/Shared FS tried with ~~disastrous~~ bad results

4. Outgoing **network access** from jobs

   Communicating with the VoBox, central services

   Direct access to data, wherever it might be

   - Both **download** and **upload**

even more …

5. Software distribution through **CVMFS**

   Kernel module + (ideally) site local squid

   Daily software releases; **calibration** files

   One more item where local disk is required (cache)

   Same binaries run on **all** resources

6. Operating system >= **SLC6**

   + **HepOSLibs** metapackage

and others …

7. Uniform authentication mechanisms

    Instead encountered keycard auth bound to a physical person, manually submitting jobs

8. 24h job slot duration (default)

And there are probably many others that I haven't thought of but these in particular hurt us.

Our vision for HPC access

Inspired by the successful testing of commercial cloud systems

- **Common interface** for resources access

  No ‘each HPC has its own rules’ please

  Full node allocation is fine

- **Common authentication/authorisation** mechanism (X.509 / GSI)

Other considerations

Generally no need for InfiniBand node interconnect

- **TCP/IP** is **required** for outside communication

Job lifetime - **allocation** or **backfill** ?

- If in ‘backfill’ mode - the payload is restricted in time, we still need reasonable time per core - average must be known for job matching

HPC usage

The simplest use case is **MC** jobs

- And a good one, it’s **70%** of our wall time!

Very limited input data

- Some configuration macros, scripts, calibration
- Binaries ran from **CVMFS**

Only generated data has to be written out

- Guaranteed average bandwidth to ‘world’ of the order of **100kB/s/core**

Full use of the HPCs

Local storage is required before any other job can match the requirements

- About **1PB** for **2000 cores** (*)
- We will **not** read data over **WAN** but as a fallback
    - It has to be possible nonetheless
- **Xrootd** is the protocol in our software stack
- ALICE analysis jobs read on average **5MB/s/core**
    - (*)Varies with the CPU core performance

Misc

Analysis jobs run on very large data sets

- We group the tasks in trains that only read the input data once
- Limited use for storage caches

Data placement algorithms assume VoBox is representative of the site actual location

- Making the resources appear at a different location is hurting IO performance

Thank you!