Hybrid batch system deployment with HTCondor and AWS spot instances

Jordi Casals, Carles Acosta, Fernando López, Vanessa Acín

IBERGRID - 23rd to 26th of September
Santiago de Compostela

Table of contents

- Port d’Informació Científica
- Hybrid Cloud Benefits
- Why Amazon Web Services?
- Local and Hybrid Infrastructures
- Configuration changes
- Issues and learning experience
- Technical conclusions

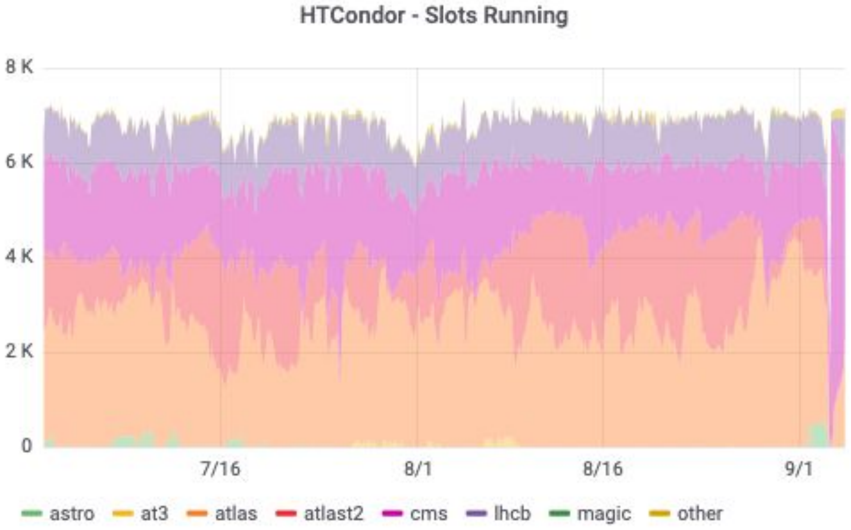
PIC
port d'informació
científica

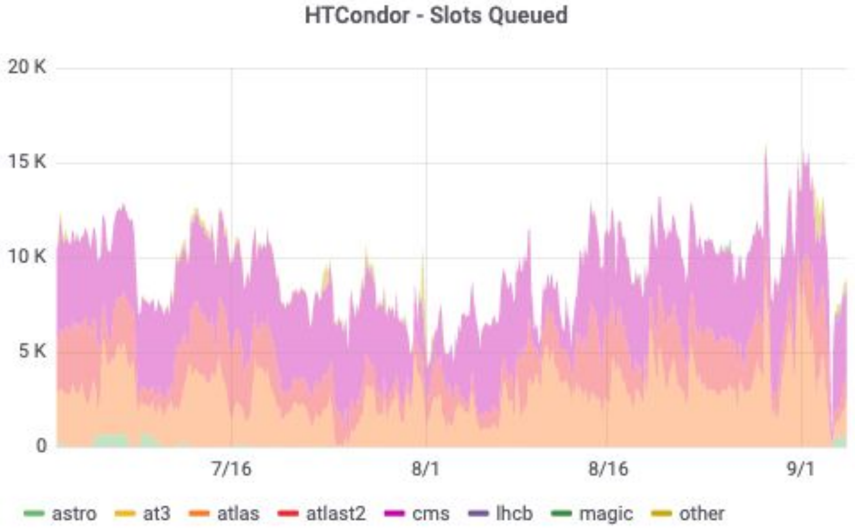
Port d'Informació Científica

- 22 people
- Tier 1 - **L**arge **H**adron **C**ollider (ATLAS, CMS, LHCb)
- Tier 2 and Tier 3 - ATLAS
- Astronomy and Cosmology Projects (MAGIC, CTA, PAU, Euclid, DES)
- Disk → 10PB
  - dCache, distributed file system
- Tape → 26PB
  - Storagetek SL8500 and IBM TS4500 (recently acquired) libraries with Enstore software
- Computing → 300 servers, ~8200 slots
  - Torque/Maui
  - **HTCondor**
- Big Data → Hadoop 16 nodes, 448 cores, 287 TB

Hybrid Cloud Benefits

- LHC experiments could get all batch resources
  - ~7500 slots running and ~8000 slots queued constantly
  - Smaller experiments and local users could be affected
- Sporadic increase of resources
  - No need for buying and setting up physical servers
- Easy to fulfil different experiments requirements
- HTCondor has an “easy” way to connect to Amazon resources
- Prepare for eventual price changes (both in local or cloud)

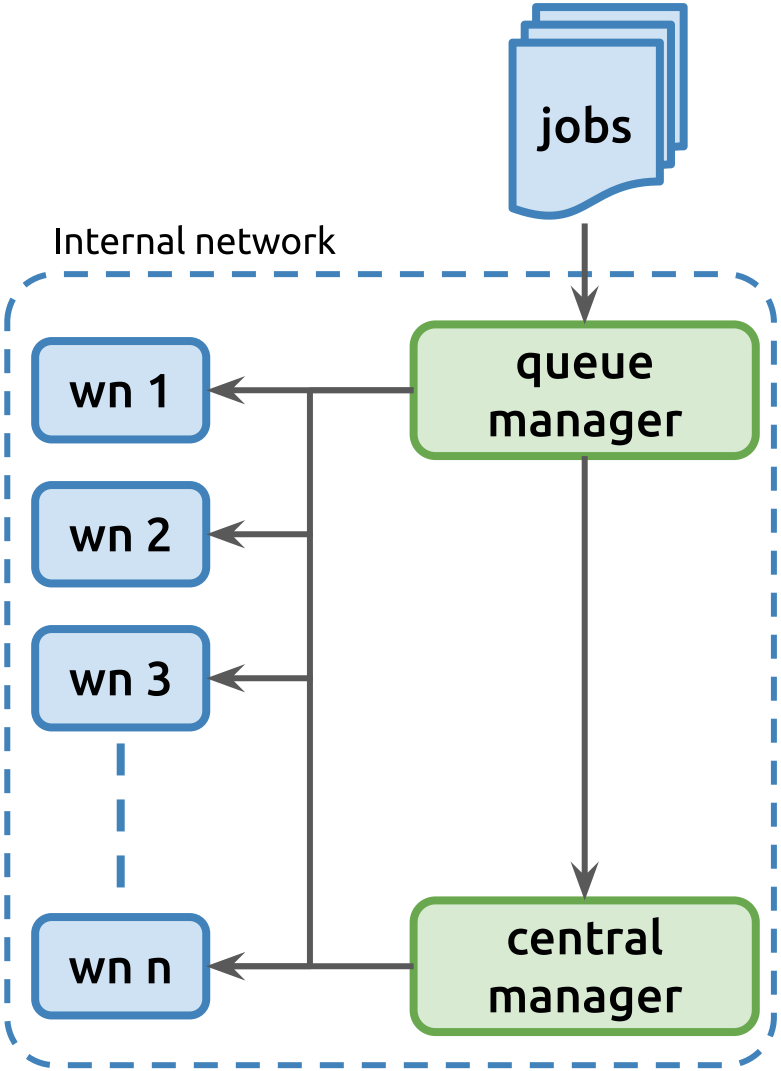




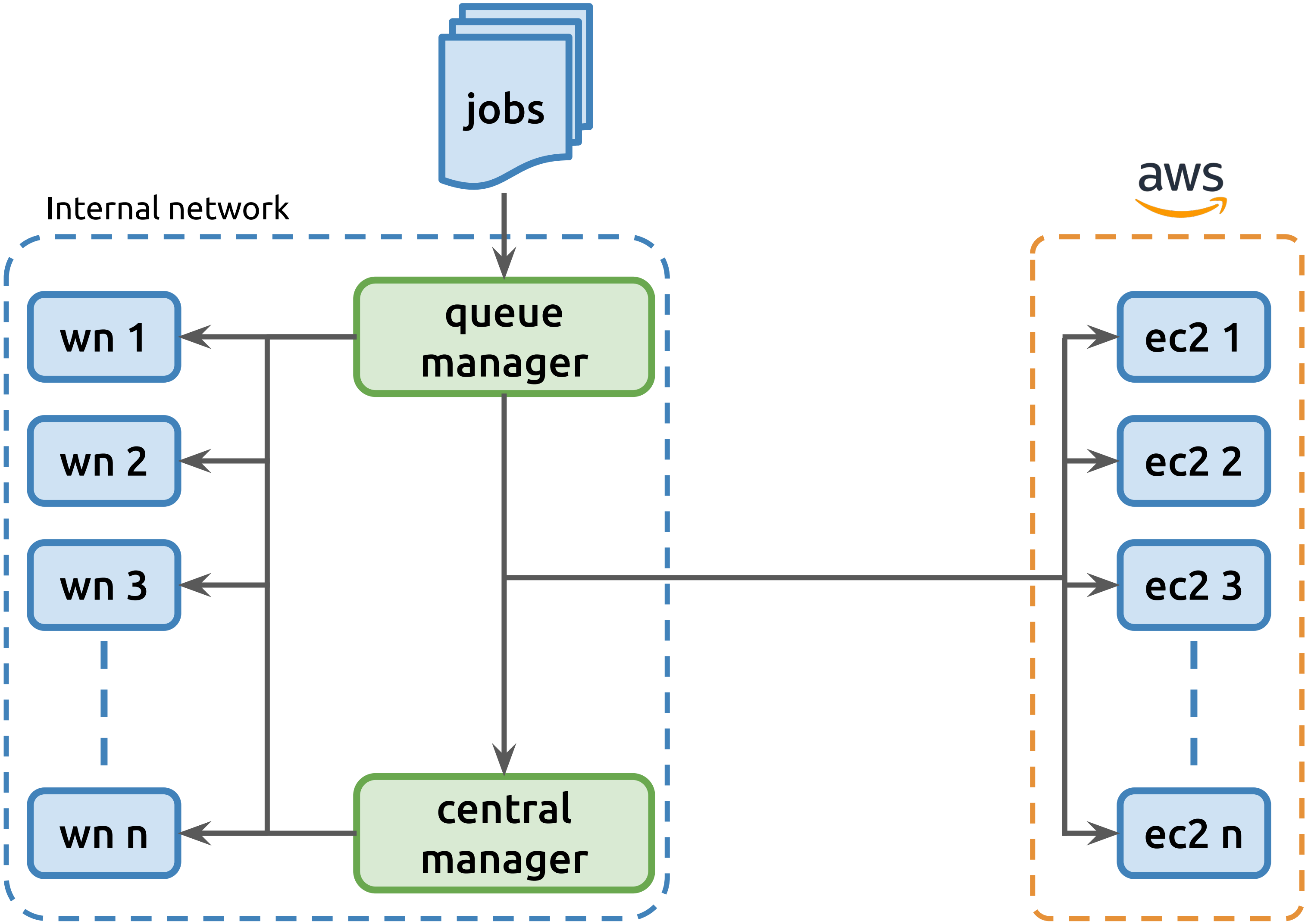
Why Amazon Web Services?

- World known cloud provider with a large community
  - Easy deployment of tests and a lot of help
- Special interest in a spot instance based scenario
- We want to test integration of a cloud environment with a local batch system
- Previous experiences with other cloud providers in a previous european experiment
  - HTCondor has a functionality to integrate itself with Amazon Web Services cloud resources
- Purchase AWS services through RedIris/GEANT IaaS Framework

Local Infrastructure



Desired Hybrid Infrastructure



How did we do it?

HTCondor

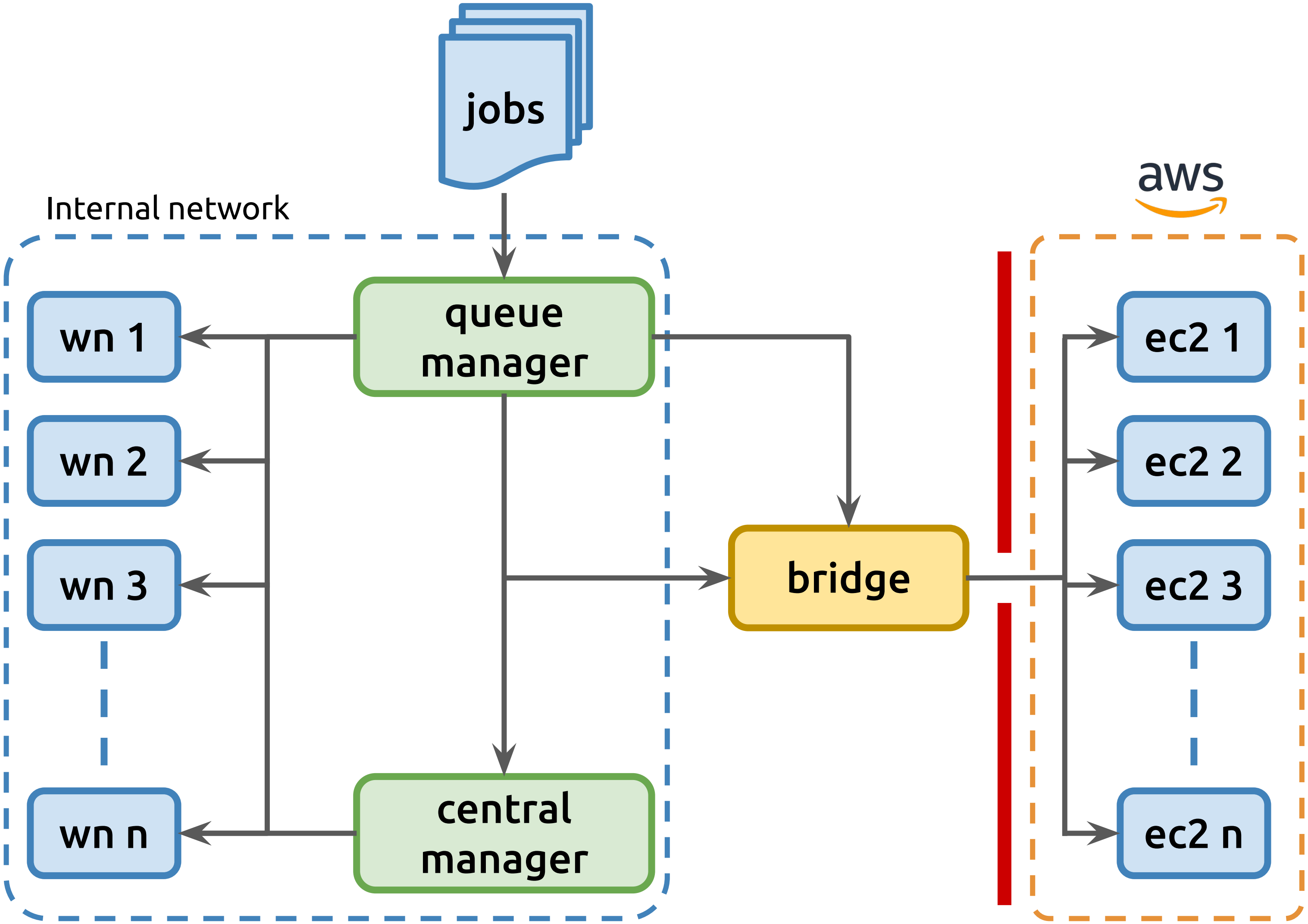
- Add AWS credentials and desired regions to condor security configuration files
- Use condor built in command to configure AWS services
  - Not really needed
- Set up HTCondor Connection Brokering (CCB)
  - Bridge server to connect the local system to the outside nodes
- HTCondor-CE routes modified so only two experiments can send jobs to both cloud and local resources
- Add IP ranges to HTCondor security, to the squids (CVMFS) and to our firewall

How did we do it?

Amazon Web Services

- Set up user credentials → everything done through API
- Create custom worker node image (everything installed manually)
  - Tried to do it with puppet, but network restrictions
    - Working on it to have a more up-to-date system
  - CollectD to send metrics, CVMFS configured, users present, public IP added to the host at startup
- Configure spot instances requirements file
  - We had a main wanted type of instance, but in spot instances it's worth to define several types to make sure the requirements are fulfilled to have machines always running

Final Hybrid Infrastructure

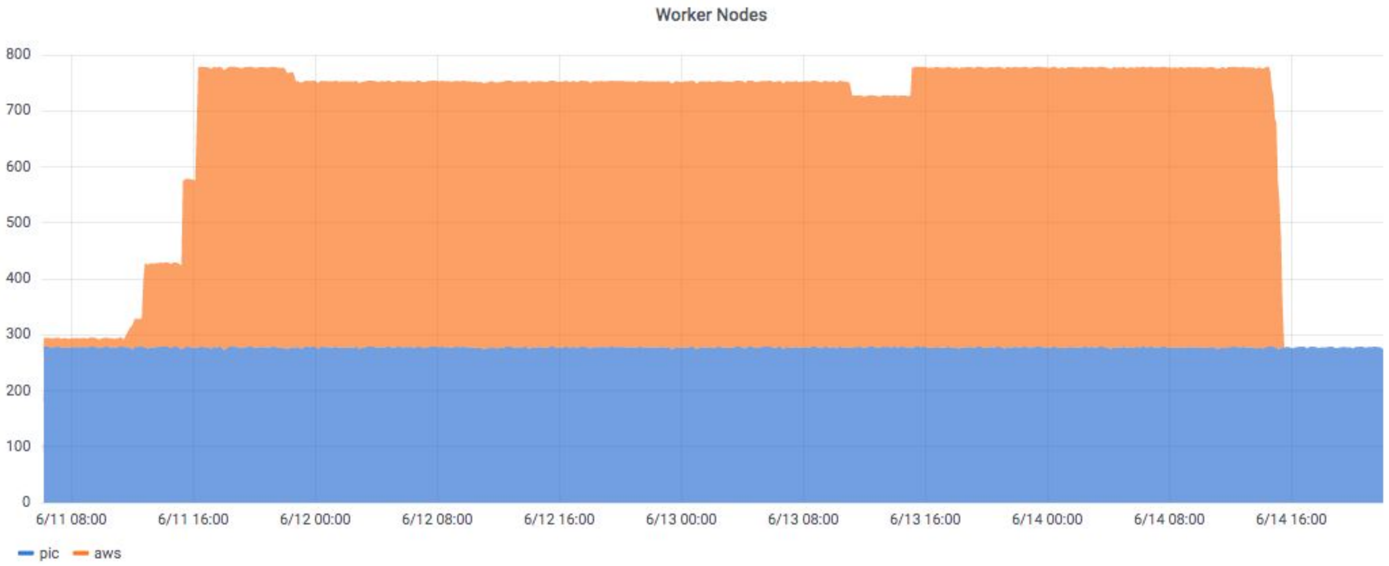


Issues

- A couple of bugs were found in the last stable version of HTCondor
    - Reported to the mailing list and the HTCondor team answered very fast and fixed the problems
    - Maybe not too many people is using HTCondor interface to AWS in real scenarios?
- Having to find all options for HTCondor to connect to AWS
    - Some default options made the system fail
- HTCondor jobs sent to AWS nodes idle time not always working
    - Machines are supposed to be stopped after some “idle” time
- Some work to understand spot instances best practices and some error messages when trying to create them
    - AWS support very helpful

Conclusions

- It’s a good option to increase computing resources sporadically
- Flexible and easy to deploy
  - Other cloud providers tested before → Helix Nebula Science Cloud
  - HTCondor may be a little help here (haven’t tested with others)
- Not very good for data intensive job processing
- Couple of nodes running to gather availability and long time data



Thank you!

Jordi Casals, Carles Acosta, Fernando López, Vanessa Acín

IBERGRID - 23rd to 26th of September
Santiago de Compostela

BACKUP SLIDES

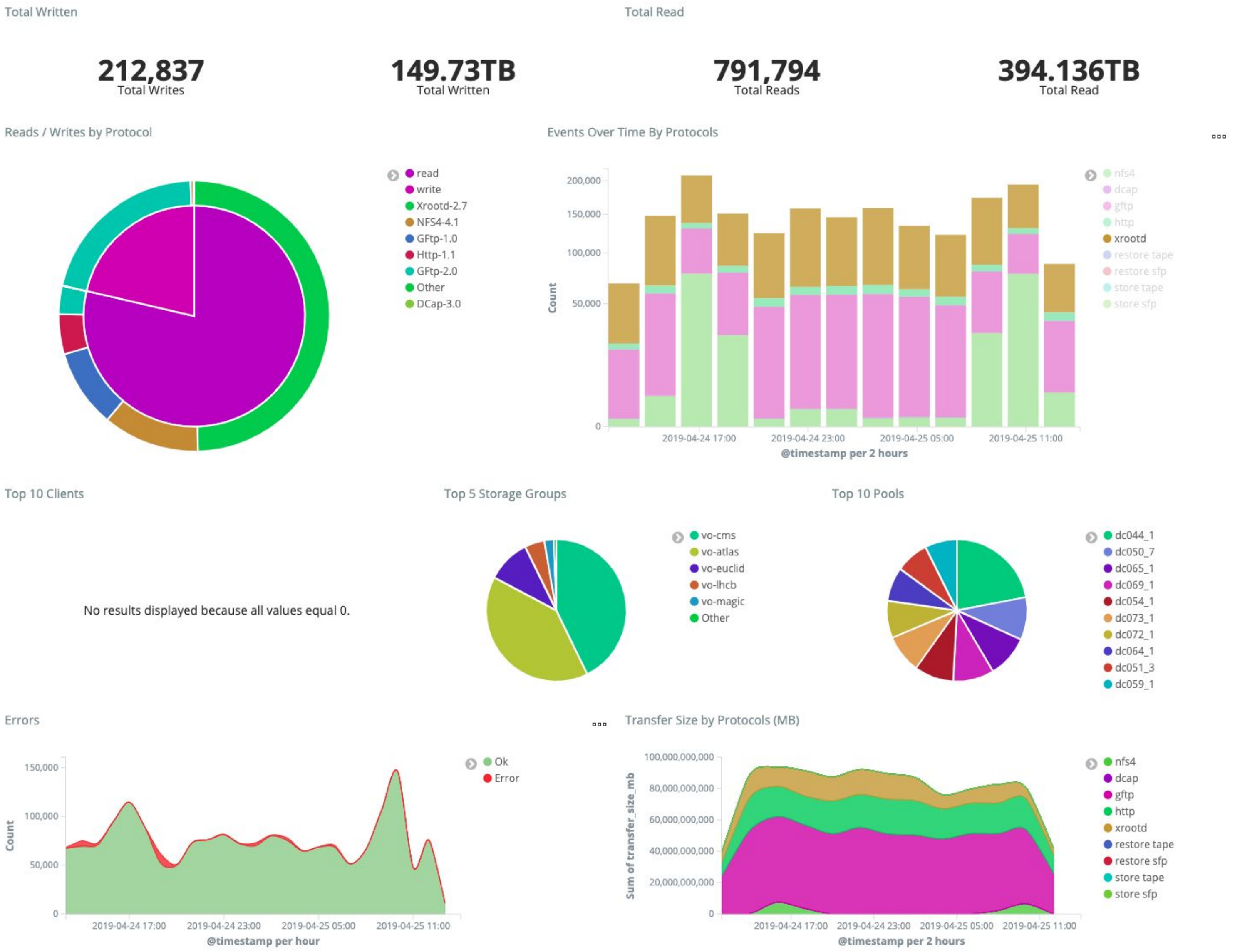
Administrative process

- Contract model: agreement inside the GÉANT Cloud IaaS framework
- Management from the IT department
  - Choose the partner → Sparkle
  - Partner sends the documents we need:
    - Call-Off Agreement
    - Contract template
- Management with our legal department:
  - Needed internal documentation:
    - Documentation certifying that RedIRIS can benefit from the GÉANT framework agreement
    - Documentation certifying that Sparkle was one of the companies selected by the GÉANT public tender
    - Documentation certifying the partnership between PIC and RedIRIS

Amazon Elasticsearch Service

- Elasticsearch es una herramienta de búsqueda de texto completo y de manera distribuida
    - Muy útil para análisis de logs *(entre otras cosas)*
- Kibana como frontend para visualizar los datos
- Puede recibir datos de un servidor de Logstash
    - Recolecta, parsea e indexa logs
- Servicio totalmente gestionado por Amazon Web Services
    - Versión, número y tipo de instancias, tamaño de disco y tipo de acceso (Público privado o limitado por rango de IP)
    - Nuestro clúster local de Elasticsearch nos daba algunos problemas de rendimiento y de durabilidad de datos
        - Valorar si vale la pena usar el servicio o pagar uno gestionado
- Más de 1.000.000.000 de líneas de log en ~ 650GB de disco

Amazon Elasticsearch Service



Total Written

212,837 Total Writes

149.73TB Total Written

Total Read

791,794 Total Reads

394.136TB Total Read

Reads / Writes by Protocol

Events Over Time By Protocols

Top 10 Clients

No results displayed because all values equal 0.

Top 5 Storage Groups

Top 10 Pools

Errors

Transfer Size by Protocols (MB)