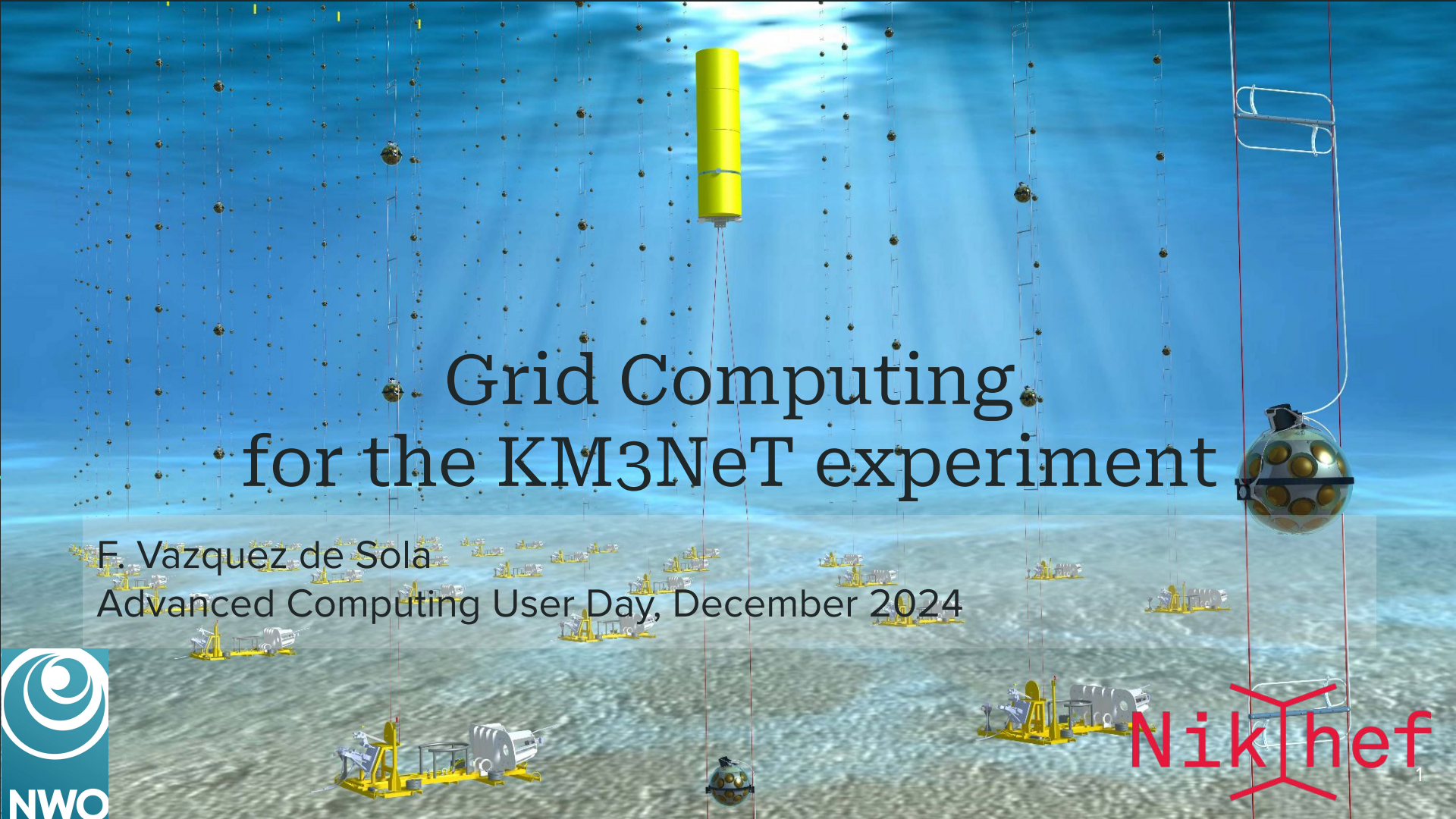
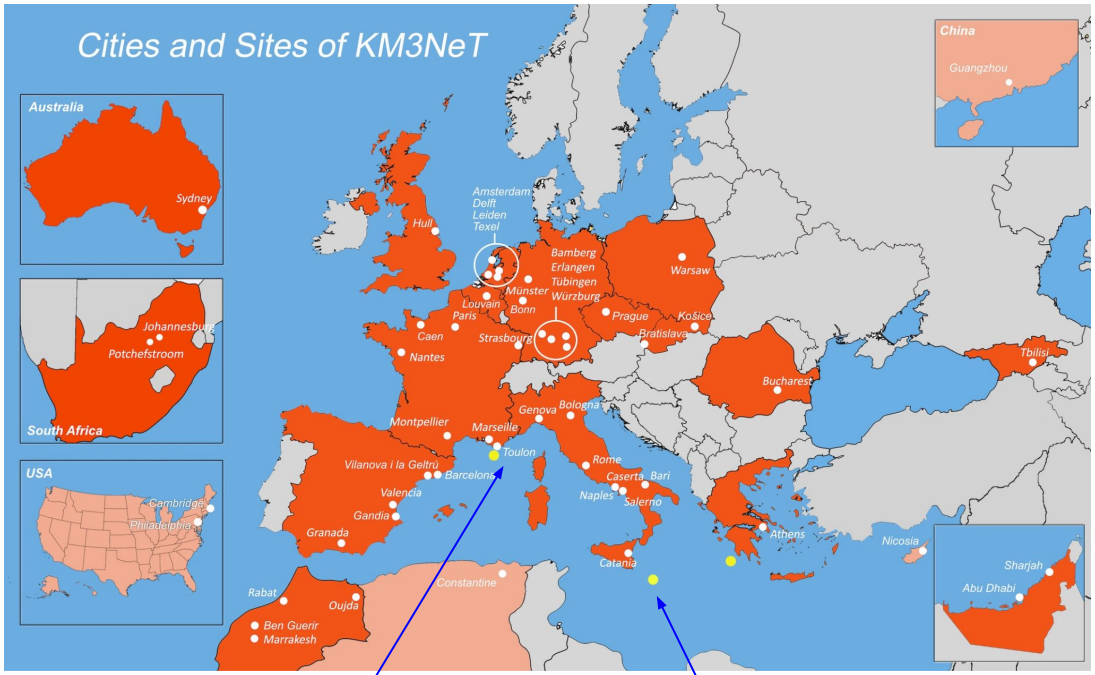


Grid Computing for the KM3NeT experiment

F. Vazquez de Sola

Advanced Computing User Day, December 2024

KM3NeT: Who?



ORCA
Vol: ~7E6 m3

ARCA
Vol: ~1 km3

~250 members, 47 partner institutes, 14 countries

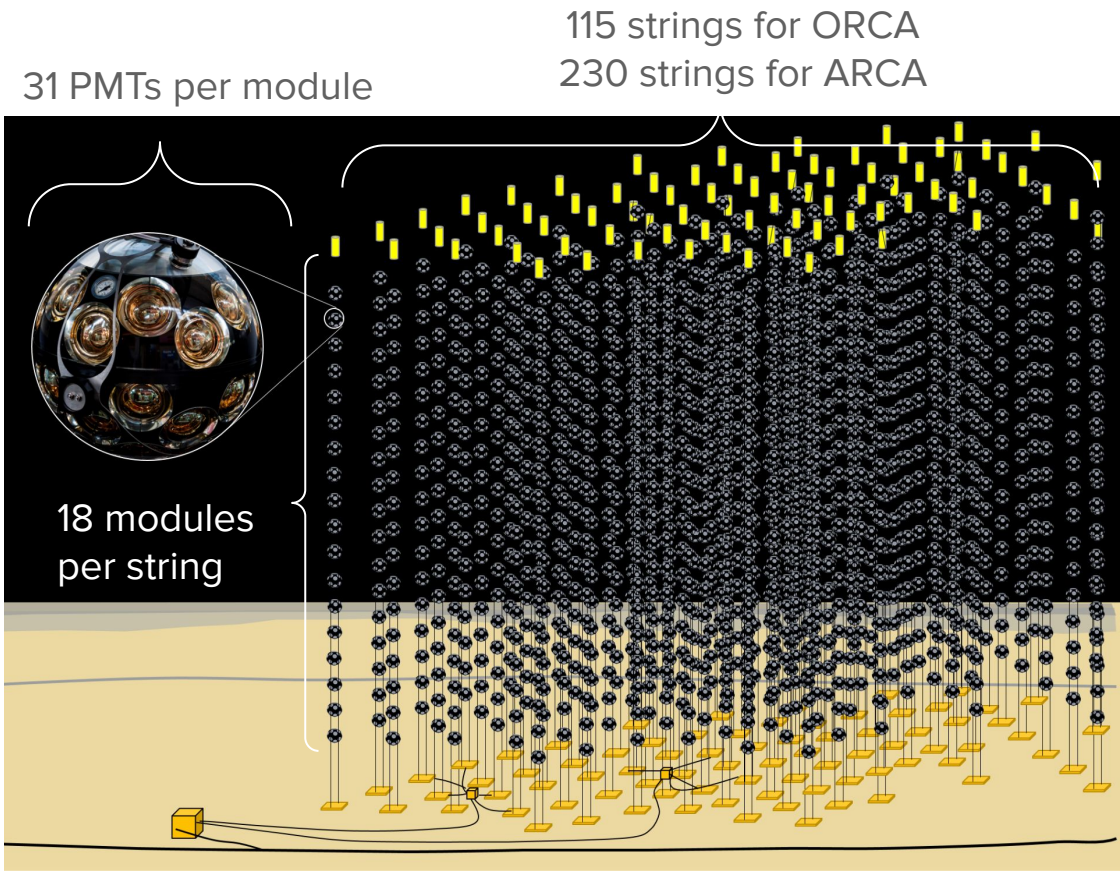
Two detectors:

- ORCA (France): study neutrino oscillations
- ARCA (Italy): study astrophysical sources of neutrinos

KM3NeT: What?

Underwater neutrino telescope array, using Photo-Multiplier Tubes (PMTs) to detect Cherenkov light from relativistic particles created by neutrino interactions in the water.

Currently ~15% of the detector deployed, but already taking data!

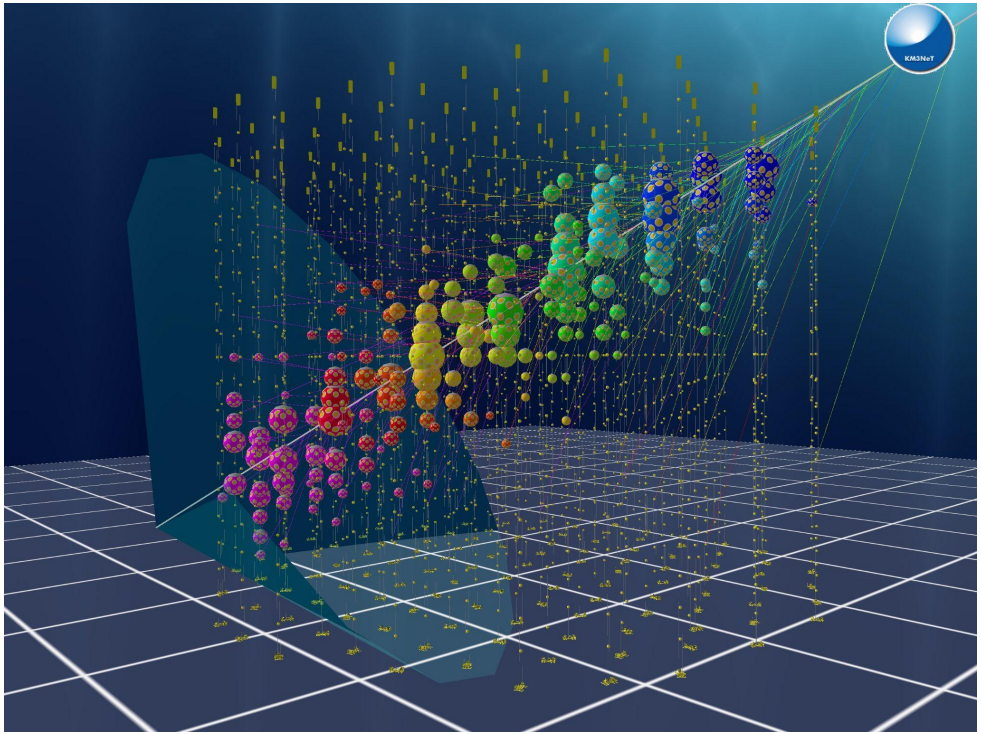


KM3NeT: How much?

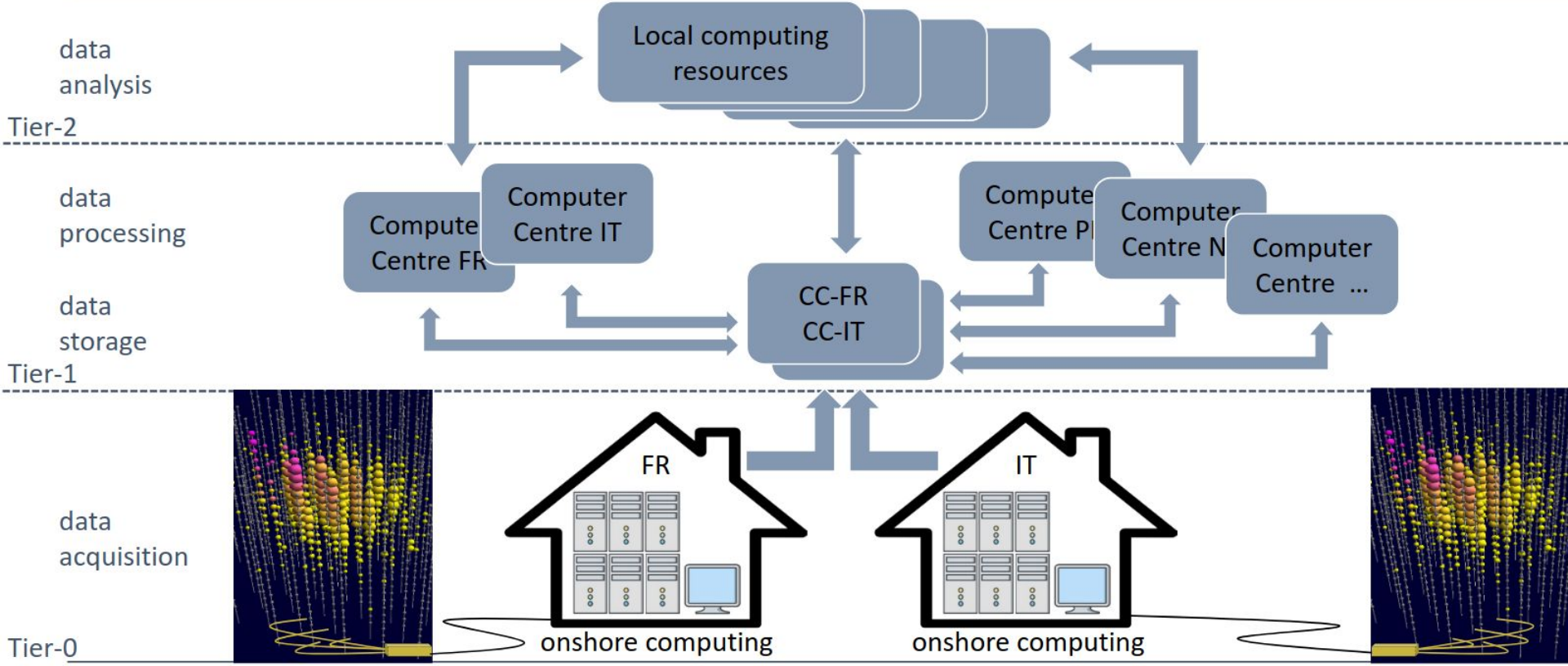
1 string: ~300 Mb/s
Full infrastructure ~100Gb/s

All data is sent to shore stations. An online filter identifies events based on coincidences between arrival time of photons and positions of PMTs, reducing event rate by factor 10^4 (estimated 500TB/year full detector)

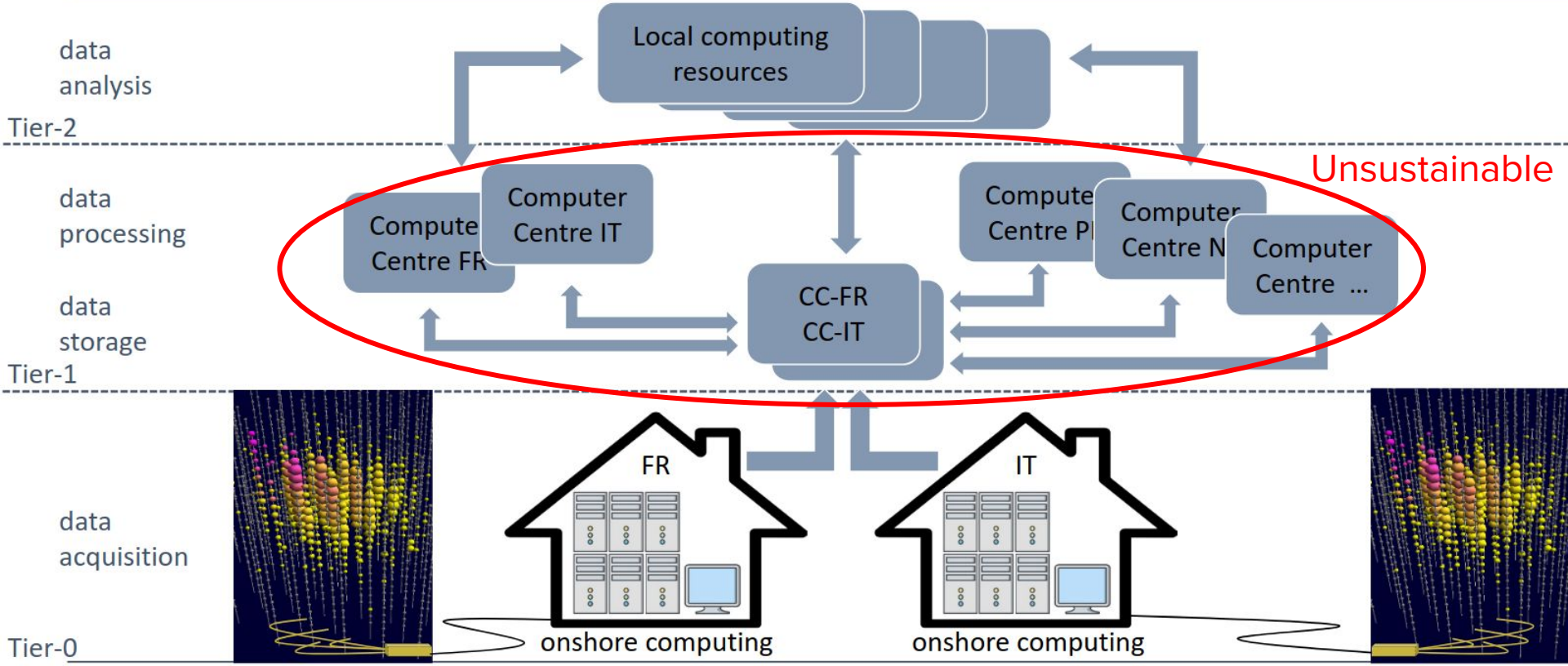
Event files copied to computing centers daily.



KM3NeT (previous) Computing Model

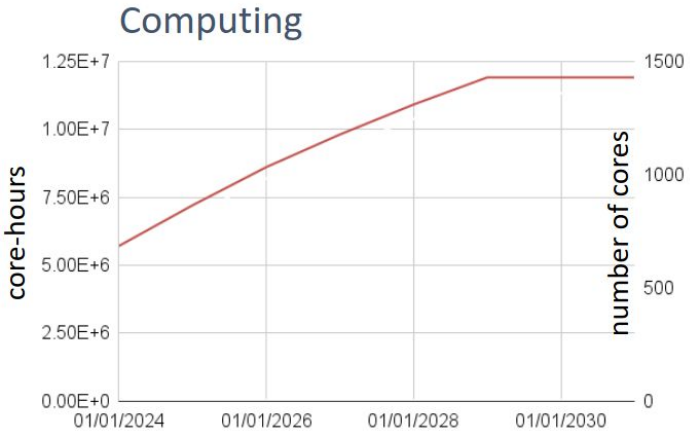


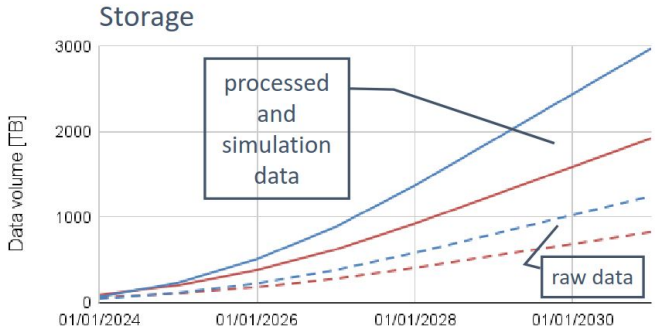
KM3NeT (previous) Computing Model



Why unsustainable?

- In practice, most data and computing ended up centralized in a single site, CC-IN2P3 (common environments, shared file system)…
  - Simulation / processing expected to grow to take O(1000) cores, and double/triple storage needs from raw data;
  - IN2P3 resources will not scale like our needs;
  - CC-IN2P3 downtime means collaboration stand-still.
- Attempting to manually split the processing across partner sites lead to large overhead: authentication, environments, file transfers, bookkeeping…

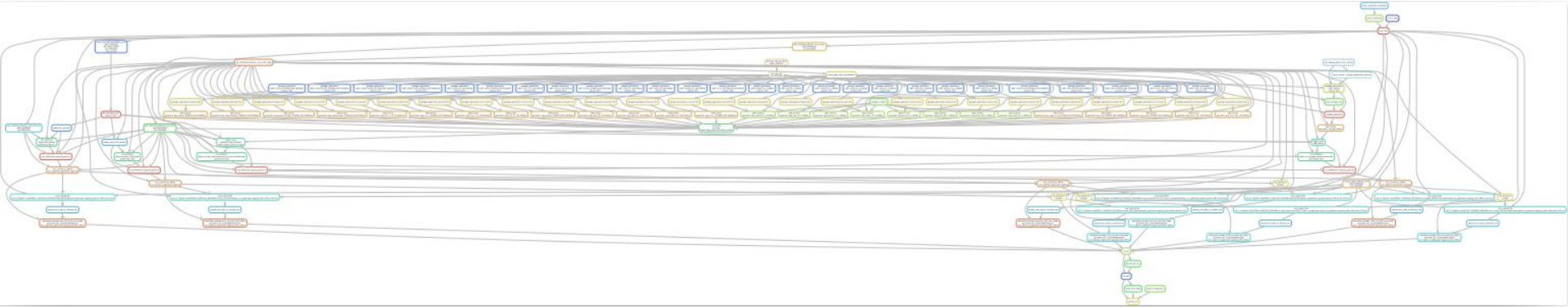




Must transition to distributed storage and computing

But first...

What do you do when your workflow looks like this?



What do you do when your workflow looks like this?

You automate it!

Snakemake

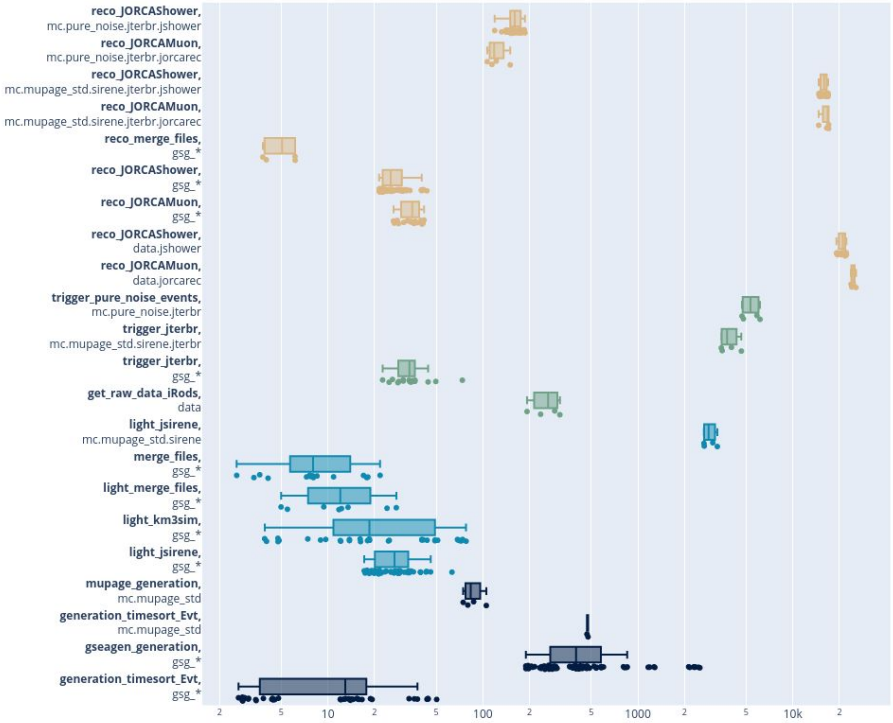
- Define rules to create output files based on input files
  - Make them generic using wildcards (e.g. {run})
- When user requests target file(s), snakemake creates a Directed Acyclic Graph (DAG) of tasks to produce target(s)
- Workflow sequence not defined by the user, but automatically by snakemake from the rules

- Can use containerized softwares:
  - Ensure software version control
  - Make workflow portable
- Managed workflow:
  - Optimize execution, monitor performances
  - Decouple job submission from workflow implementation
  - Take care of logging and (most of) provenance

What we like

- Does not require advanced knowledge of whole processing for users
- Integrated use of containers for running on different sites
- Single configuration file for all steps
- Automated file organization, log bookkeeping, benchmarking

Took one dedicated KM3NeT member (V. Pestel) 1-1.5 years to reimplement our workflow (MCs + processing) in snakemake. Now everybody wants to use it, or develop their own for their projects!



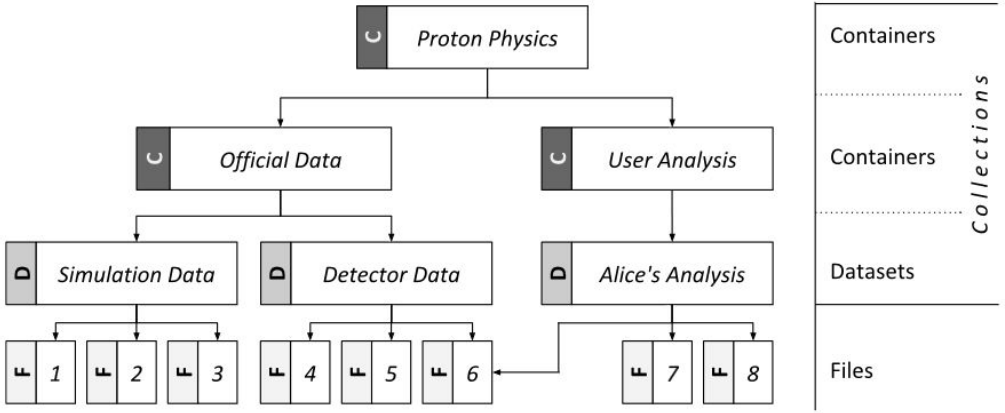
To the Grid!

Storage: Rucio

https://link.springer.com/article/10.1007/s41781-019-0026-3

Grid Storage

- Regular access to grid storage requires knowing physical location of every file:
  - Host address and port
  - Path to file within site
  - File transfer protocol (gfal, xrootd, webdav...)
- Enter RUCIO!
  - Provides Data IDentifiers and replication rules
  - Manages file transfer protocols
  - Easy to use



Rucio key concepts

Data Identifiers: unique IDs for files, datasets (collection of files) and containers (collection of datasets and containers). Each DID can hold arbitrary user-defined metadata

Declarative data management: Express what you want with rules and subscriptions, then Rucio constantly evaluates and tries to satisfy them. Examples:

1) Rules: "Three copies of this dataset, distributed evenly across three institutes on different continents, with two copies on DISK and one on TAPE"
2) Subscriptions: "Three copies (two on DISK, one on TAPE) of every new DIDs that will be produced that match the set of metadata datatype=RAW and scope=data"

Set up our own kubernetes cluster at Nikhef to run Rucio, thanks to Victor Azizi and Bouwe Andela from eScience Center

RUCIO command examples

$ rucio list-dids CORSIKA_testing: --filter type=DATASET

```
+-------------------------------------------------------------------------+-----------------+
| SCOPE:NAME                                                              | [DID TYPE]      |
|-------------------------------------------------------------------------+-----------------|
| CORSIKA_testing:SIBYLL_DefaultAtmo_TeV_low_C_20240328-1103              | DIDType.DATASET |
| CORSIKA_testing:SIBYLL_DefaultAtmo_TeV_low_p_20240328-1536              | DIDType.DATASET |
| CORSIKA_testing:SIBYLL_DefaultAtmo_TeV_low_p_20240328-1552              | DIDType.DATASET |
| CORSIKA_testing:SIBYLL_DefaultAtmo_TeV_low_p_20240402-0204              | DIDType.DATASET |
| CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_EeV_p_20240402-1553         | DIDType.DATASET |
| CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_EeV_p_20240404-1732         | DIDType.DATASET |
| CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_EeV_p_20240405-1357         | DIDType.DATASET |
| CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_TeV_low_p_20240406-1926     | DIDType.DATASET |
| CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_TeV_low_p_20240408-0051     | DIDType.DATASET |
| CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_TeV_low_p_20240408-0052     | DIDType.DATASET |
+-------------------------------------------------------------------------+-----------------+
```

$ rucio list-dids CORSIKA_testing: --filter Production=TeV_low,NumberShowers.gte=2000

```
+-----------------------------------------------------------+--------------+
| SCOPE:NAME                                                | [DID TYPE]   |
|-----------------------------------------------------------+--------------|
| CORSIKA_testing:SIBYLL_DefaultAtmo_TeV_low_p_20240328-1536 |              |
| CORSIKA_testing:SIBYLL_DefaultAtmo_TeV_low_p_20240328-1552 |              |
| CORSIKA_testing:SIBYLL_DefaultAtmo_TeV_low_p_20240402-0204 |              |
+-----------------------------------------------------------+--------------+
```

$ rucio list-files CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_TeV_low_p_20240406-1926

```
+-------------------------------------------------------------------------------------+--------------------------------------+------------+------------+----------+
| SCOPE:NAME                                                                          | GUID                                 | ADLER32    | FILESIZE   | EVENTS   |
|-------------------------------------------------------------------------------------+--------------------------------------+------------+------------+----------|
| CORSIKA_testing:DAT_SIBYLL-star-p03_DefaultAtmo_TeV_low_p_20240406-1926_000500.gz   | A77054B3-2737-43D6-9148-B82E5FB44C9F | ad:62b4e0f6 | 66.120 kB  |          |
| CORSIKA_testing:LOG_SIBYLL-star-p03_DefaultAtmo_TeV_low_p_20240406-1926_000500.tar.gz | 1961FE8B-4B25-4558-98FC-A51AB18429C7 | ad:c71c6de7 | 37.762 kB  |          |
+-------------------------------------------------------------------------------------+--------------------------------------+------------+------------+----------+
```

$ rucio download CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_TeV_low_p_20240406-1926

```
--------------------------------
Download summary
--------------------------------------
DID CORSIKA_testing:SIBYLL-star-p03_DefaultAtmo_TeV_low_p_20240406-1926
Total files (DID):                          2
Total files (filtered):                     2
Downloaded files:                           2
Files already found locally:                0
Files that cannot be downloaded:            0
```

To the Grid!

Computing: Dirac

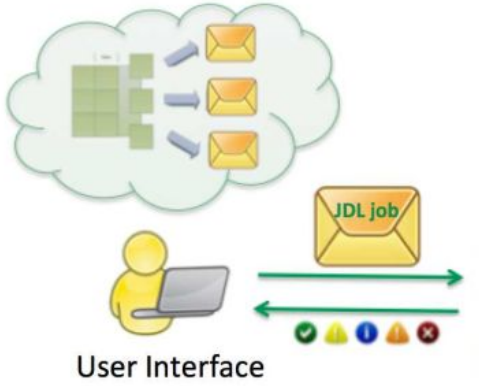
https://dirac.readthedocs.io/en/latest/index.html

DIRAC

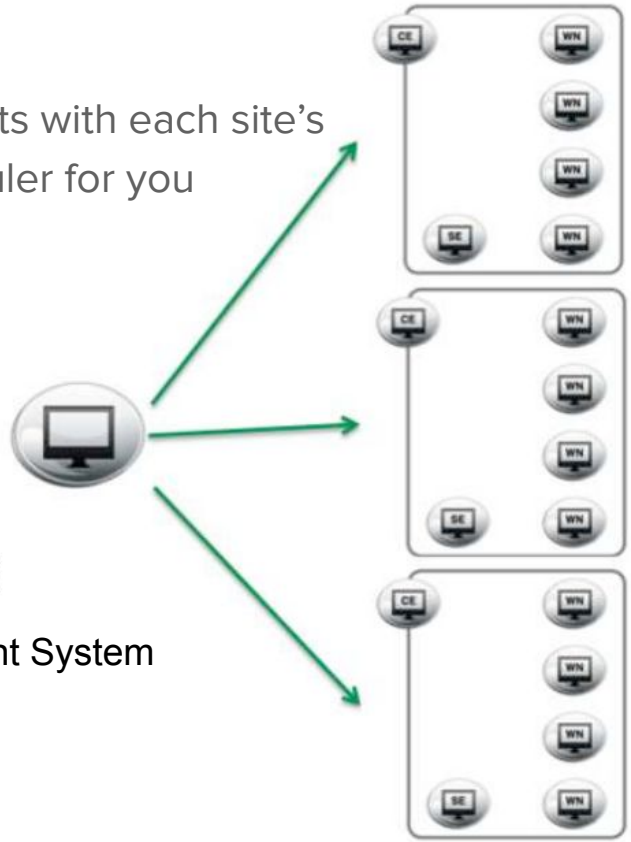
Distributed Infrastructure with Remote Agent Control

Originally developed by LHCb.

DIRAC middleware mediates between user and Grid resources.



Interacts with each site's scheduler for you



WMS

Worker Management System

https://dirac.readthedocs.io/en/latest/index.html

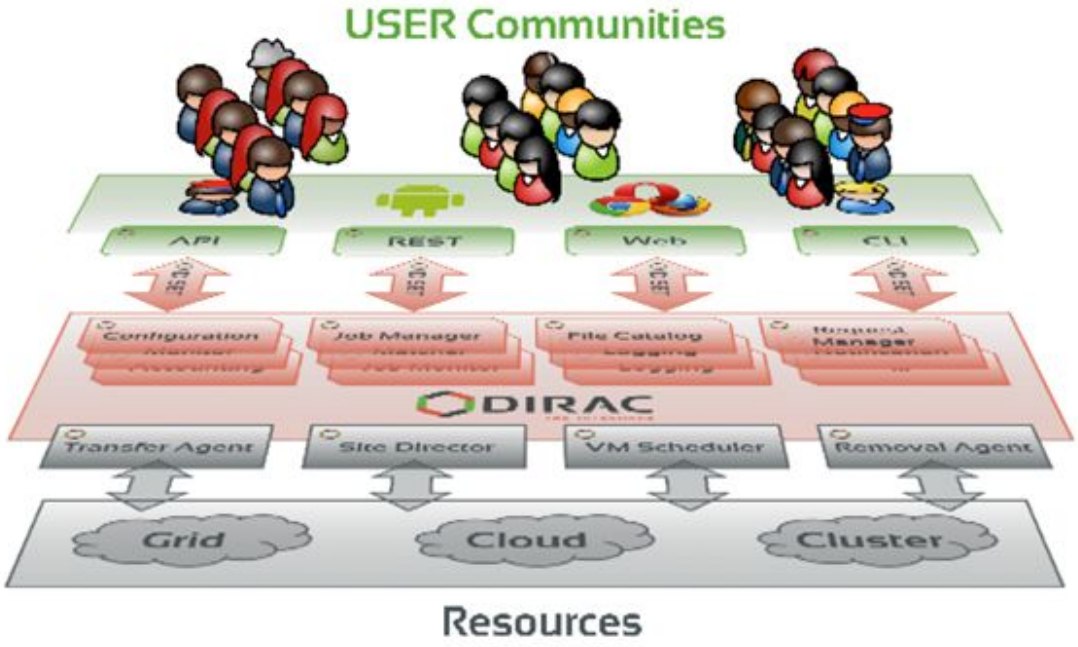
DIRAC

Distributed Infrastructure with Remote Agent Control

Originally developed by LHCb.

DIRAC middleware mediates between user and Grid resources.

Manages your authentication, submits your jobs to “free” sites, monitors their progress, recovers outputs...



(and many more things we don’t use, such as their own file catalog and workflow management system)

Running on the grid

-> Either use the Job Description Language, or the Python API

Job.jdl

```
Executable="runstuff.sh inputfile.txt";
StdOutput="stdout.txt";
StdError="stderr.txt";
InputSandbox={"runstuff.sh","inputfile.txt"};
OutputSandbox={"stdout.txt","stderr.txt"};
CPUTime=10000;
NumberOfProcessors=4;
```

$ dirac-wms-job-submit Job.jdl

RunJob.py

```
#!/usr/bin/env python

from DIRAC.Core.Base import Script
script_pcl = Script.initialize()

from DIRAC.Interfaces.API.Job import Job
from DIRAC.Interfaces.API.Dirac import Dirac

dirac = Dirac()
j = Job()

j.setInputSandbox(['runstuff.sh', 'inputfile.txt'])
j.setOutputSandbox(['stdout.txt', 'stderr.txt'])
j.setCPUTime(10000)
j.setNumberOfProcessors(4)
j.setExecutable('runstuff.sh', arguments='inputfile.txt')

jobSubmission = dirac.submitJob(j) # mode='local'
print( jobSubmission['Value'] )
```

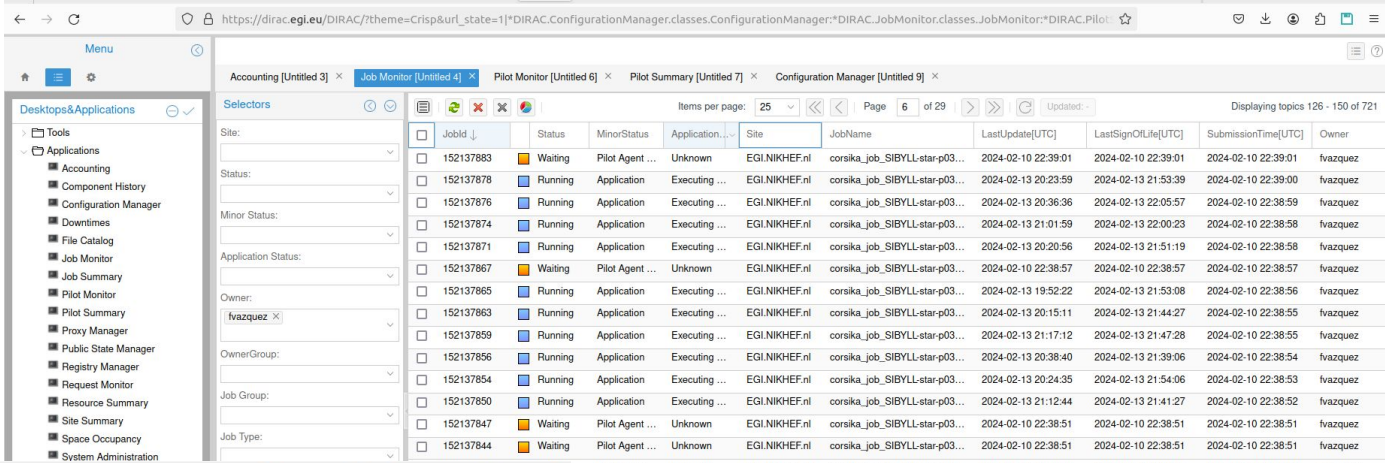
$ ./RunJob.py

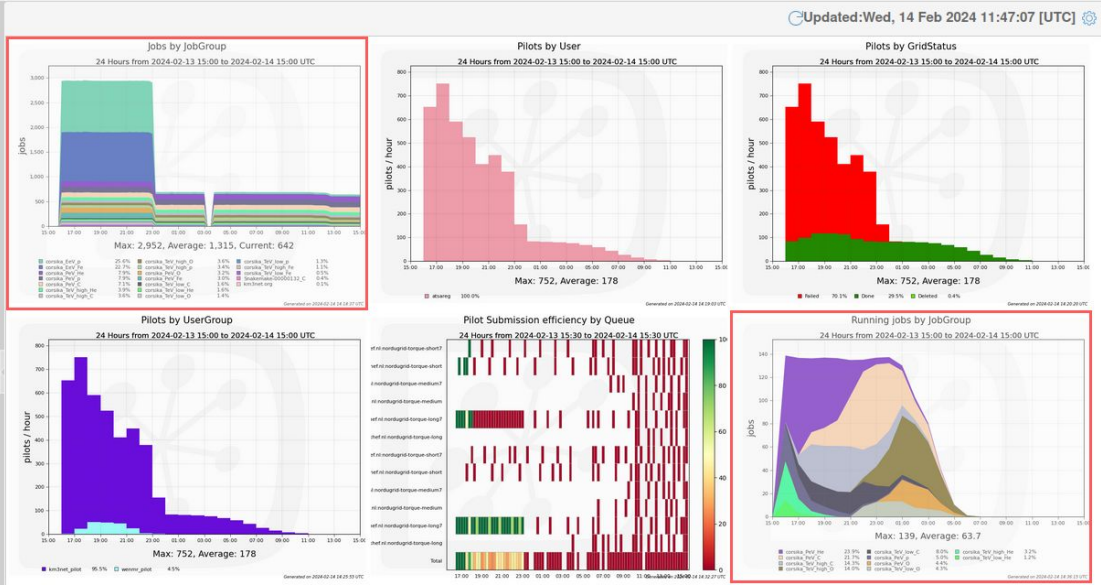
Note that CPUTime is in HS06 seconds (not walltime)

Can’t use sandbox to automatically transfer files above 10MB: need to download/upload inside job itself (i.e. with Rucio commands)

DIRAC: Dashboard

Web interface for monitoring and logging your jobs





E.g. EGI:
https://dirac.egi.eu/DIRAC/

SURF:
https://dirac.surfsara.nl/DIRAC

Infrastructure

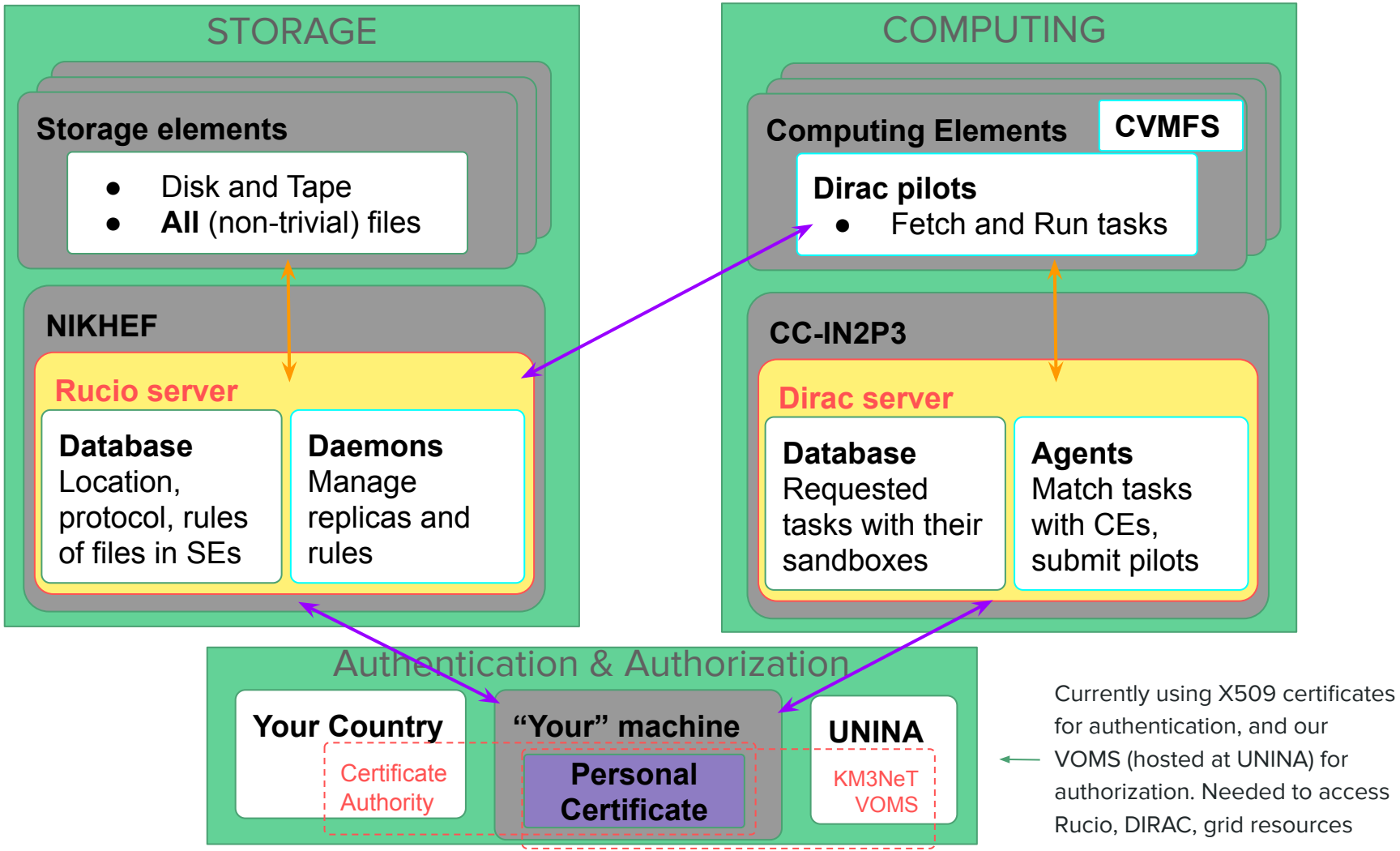
Using pre-existing EGI DIRAC infrastructure at CC-IN2P3, with invaluable support from Andrei Tsaregorodtsev. SURF also has DIRAC server, but less mature.

Authentication managed automatically by Dirac through proxies of the user's certificate.

Containers deployed to all Worker Nodes through the CernVM File System (CVMFS).

No integration with Dirac's File Catalog (we already had Rucio) or Transformation System (we already had Snakemake) - maybe in the future.

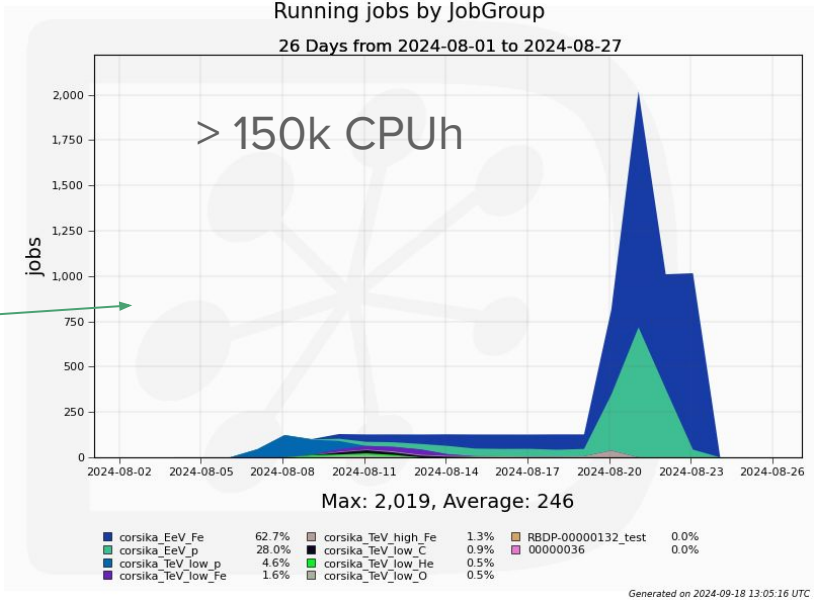
Summary


STORAGE
Storage elements
Disk and Tape
All (non-trivial) files
NIKHEF
Rucio server
Database
Location, protocol, rules of files in SEs
Daemons
Manage replicas and rules
COMPUTING
Computing Elements
CVMFS
Dirac pilots
Fetch and Run tasks
CC-IN2P3
Dirac server
Database
Requested tasks with their sandboxes
Agents
Match tasks with CEs, submit pilots
Authentication & Authorization
Your Country
Certificate Authority
"Your" machine
Personal Certificate
UNINA
KM3NeT VOMS
Currently using X509 certificates for authentication, and our VOMS (hosted at UNINA) for authorization. Needed to access Rucio, DIRAC, grid resources

First large-ish test

Simulation of cosmic ray air showers (CORSIKA). Successfully submitted jobs to grid resources via DIRAC, stored results and logs with Rucio.

Now testing full snakemake workflow, to be scaled up to mass processings.




```
(Linux-x86_64)(improve_workflow) vi: grid-snakemake$ rucio list-files Snakemake_testing:RBDP_v9.0_complete_00000132_00014590
+-------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------------------------------+------------+------------+
| SCOPE:NAME                                                                                                                                                  | GUID                                 | ADLER32    | FILESIZE   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------------------------------+------------+------------|
| Snakemake_testing:KM3NeT_00000132_00014590.data.jppmuon_jppshower-upgoing_dynamic.offline.dst.v9.0_complete.root                                            | A7C90E27-84AB-478E-870E-86D463C8F4C4 | ad:c4db4fc2 | 77.633 MB  |
| Snakemake_testing:KM3NeT_00000132_00014590.data.jppmuon_jppshower-upgoing_dynamic.offline.v9.0_complete.root                                                | FF61D6E9-890F-42A6-84C9-AD0E3AAD31B3 | ad:c3e6188c | 2.349 GB   |
| Snakemake_testing:KM3NeT_00000132_00014590.jppmuon_jppshower-upgoing.v9.0_complete.archived_logs.tar.gz                                                     | B4B5DCF7-E3D2-49B0-B709-20CE7461F105 | ad:69597686 | 24.494 MB  |
| Snakemake_testing:KM3NeT_00000132_00014590.mc.gsg_neutrinos.km3sim_sirene_merged.jterbr.jppmuon_jppshower-upgoing_static.offline.dst.v9.0_complete.root     | 6D36FE98-190E-4471-889A-82D8306B75A1 | ad:8f23144b | 2.485 MB   |
| Snakemake_testing:KM3NeT_00000132_00014590.mc.gsg_neutrinos.km3sim_sirene_merged.jterbr.jppmuon_jppshower-upgoing_static.offline.v9.0_complete.root         | F9B2185G-BFA8-4AA2-8899-024B2793D721 | ad:fa17a562 | 55.725 MB  |
| Snakemake_testing:KM3NeT_00000132_00014590.mc.mupage_default.sirene.jterbr.jppmuon_jppshower-upgoing_static.offline.dst.v9.0_complete.root                  | FD0B92A0-FD59-4E43-B9B2-7D1F78D7360D | ad:1298b23f | 98.040 MB  |
| Snakemake_testing:KM3NeT_00000132_00014590.mc.mupage_default.sirene.jterbr.jppmuon_jppshower-upgoing_static.offline.v9.0_complete.root                      | 46BDE537-6866-460B-A140-772E45C50D8E | ad:b8a0326d | 2.664 GB   |
| Snakemake_testing:KM3NeT_00000132_00014590.mc.pure_noise.jterbr.jppmuon_jppshower-upgoing_static.offline.dst.v9.0_complete.root                             | C2958719-E7B2-4D1D-88E3-C19354E24380 | ad:66cfb245 | 8.530 MB   |
| Snakemake_testing:KM3NeT_00000132_00014590.mc.pure_noise.jterbr.jppmuon_jppshower-upgoing_static.offline.v9.0_complete.root                                | FBDC3D1E-BE08-442E-8885-373EFF368EAE | ad:3db17105 | 189.602 MB |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------------------------------+------------+------------+
Total files : 9
Total size : 5.470 GB
```

Output files from one run generated through grid computing

Some takeaways

The trickiest part is getting information on how the software works (documentation is limited) and how the infrastructure is configured. First step is to identify where knowledge is (documentation or support), focus on tools where you know where to turn when problems arise. Then keep that knowledge within your collaboration!

Testing grid computing is inherently slow due to scheduling, and can easily become a bottleneck for implementation efforts. Make sure your payload software/scripts are robust before testing starts, develop solid naming and bookkeeping conventions, and recruit personpower to help with testing.

If starting grid journey from scratch, consider using DIRAC as single-stop for all grid needs instead of multiplying interfaces: computing, storage and workflow management. Heavier to get started, but maybe saves time in the end?

What have we achieved?

We started with almost no in-collaboration knowledge, and now we have:

- Grid storage accessible with Rucio - all raw and calibration data available with a single command, and integrated into our existing workflow, transparently for our users. Implementation time: roughly two people working for a year.
- Grid computing accessible with DIRAC. Ran both simulations and the whole workflow. Implementation time: roughly one person working for two years (but cheating a bit: we did not set up our own DIRAC instance).

-> Not tackled: better AAI solutions, splitting jobs into “grid-friendly” chunks, processing trains, automated job submission… The Grid journey can be a long one.

What have we achieved?

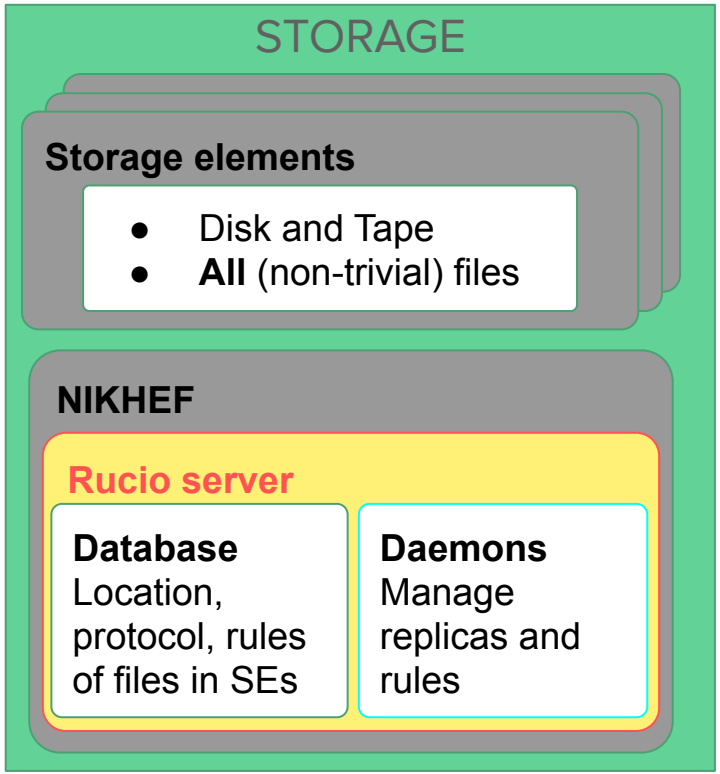
We started with almost no in-collaboration knowledge, and now we have:

- Grid storage accessible with Rucio - all raw and calibration data available with a single command, and integrated into our existing workflow, transparently for our users. Implementation time: roughly two people working for a year.
- Grid computing accessible with DIRAC. Ran both simulations and the whole workflow. Implementation time: roughly one person working for two years (but cheating a bit: we did not set up our own DIRAC instance).

-> Not tackled: better AAI solutions, splitting jobs into “grid-friendly” chunks, processing trains, automated job submission… The Grid journey can be a long one.

Thank you for your attention!

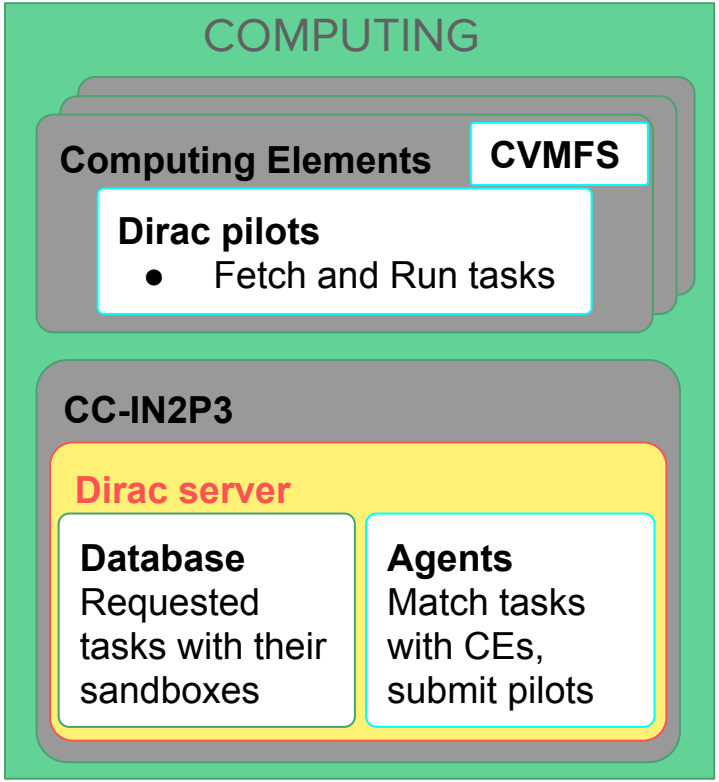
Backup



Thanks to Victor Azizi and Bouwe Andela from eScience Center

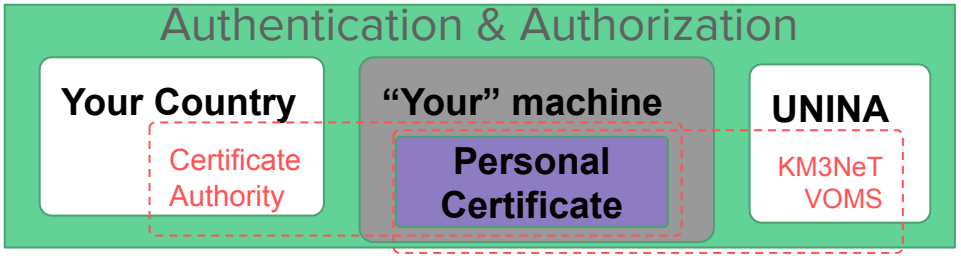
- Rucio is an interface software to grid storage.
- Provides user-chosen Data Identifiers (DIDs, effectively “aliases”) instead of true paths and protocols to files.
- Can organize files with datasets and containers, and add metadata.
- Convenient way to share files between sites / collaborators.
- Manages replicas automatically through replication rules.

- Dirac is the middleware interface to computing resources, originally developed by LHCb.
- Mediates between user and site schedulers, propagates user authentication, monitors job progress... (and much more, that we don't use)
- Computing Elements are equipped with the CernVM File System (CVMFS), allowing access to our software for processing.



Thanks to Andrei Tsaregorodtsev. (and many others) from EGI

- Authentication & Authorization is what allows the user to access Dirac and Rucio, and any other grid resource.
- Currently using X509 certificates, which requires both a country-specific and a collaboration-specific procedure.
  - Arguably the most annoying part of working with grid resources.
- Our VOMS is at UNINA, but stopping support in next few years. We want to move to alternative (i.e. tokens) before then.



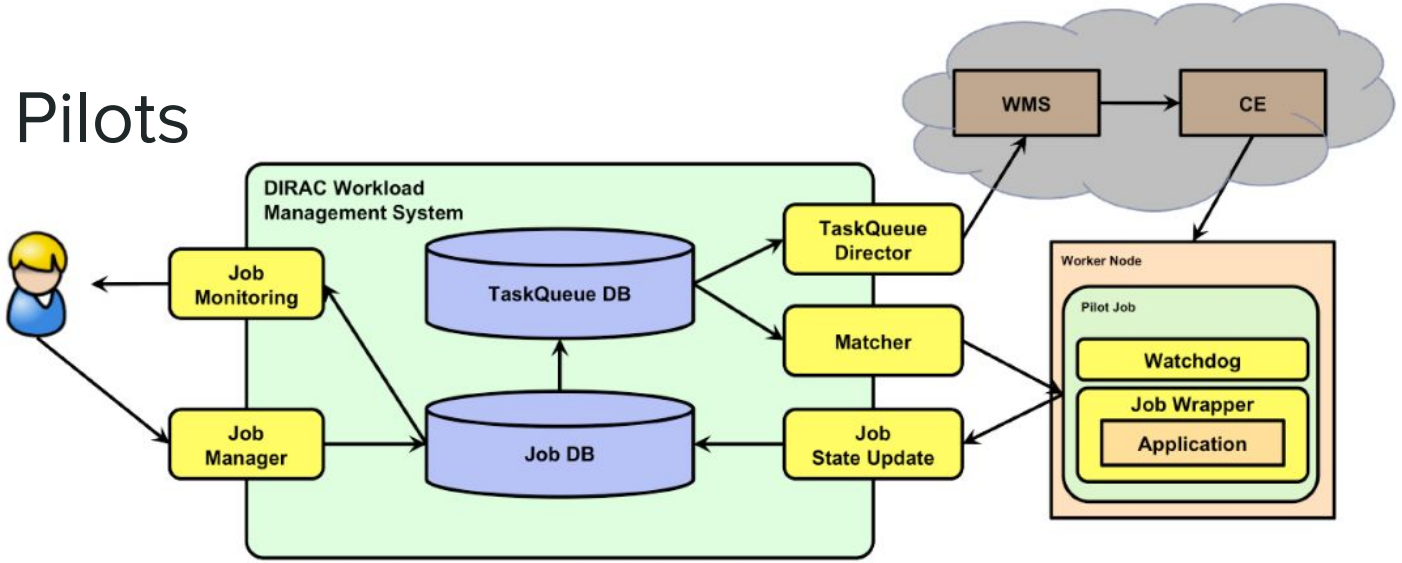
Authentication & Authorization: Certificates

- **Authentication** from Certification Authority: National entity that issues certificates

- **Authorization** from “Virtual Organisation”: Group sharing scientific field and research interests (e.g. lsgrid (lifesciences), escape, lofar, km3net.org...)

    - Determine which resources (compute/storage) you can access via extensions to your certificate
    - VOMS (VO Management Service) sets user roles and privileges in a VO, maintains server that returns certificate attributes (eg. membership)

DIRAC Pilots



Pilots are wrapper jobs sent by DIRAC to Worker Nodes at clusters. They setup the environment and then request “tasks” (jobs) from DIRAC server until they expire.

- Minimize resources spent setting up : can take multiple successive jobs
- Simplifies discovery of free resources (pull vs push system)

Monitoring

Dashboard for short term monitoring, eg. EGI DIRAC server: https://dirac.egi.eu/DIRAC/

EGI accounting portal for long term use, eg. Netherlands: https://accounting.egi.eu/egi/country/Netherlands/