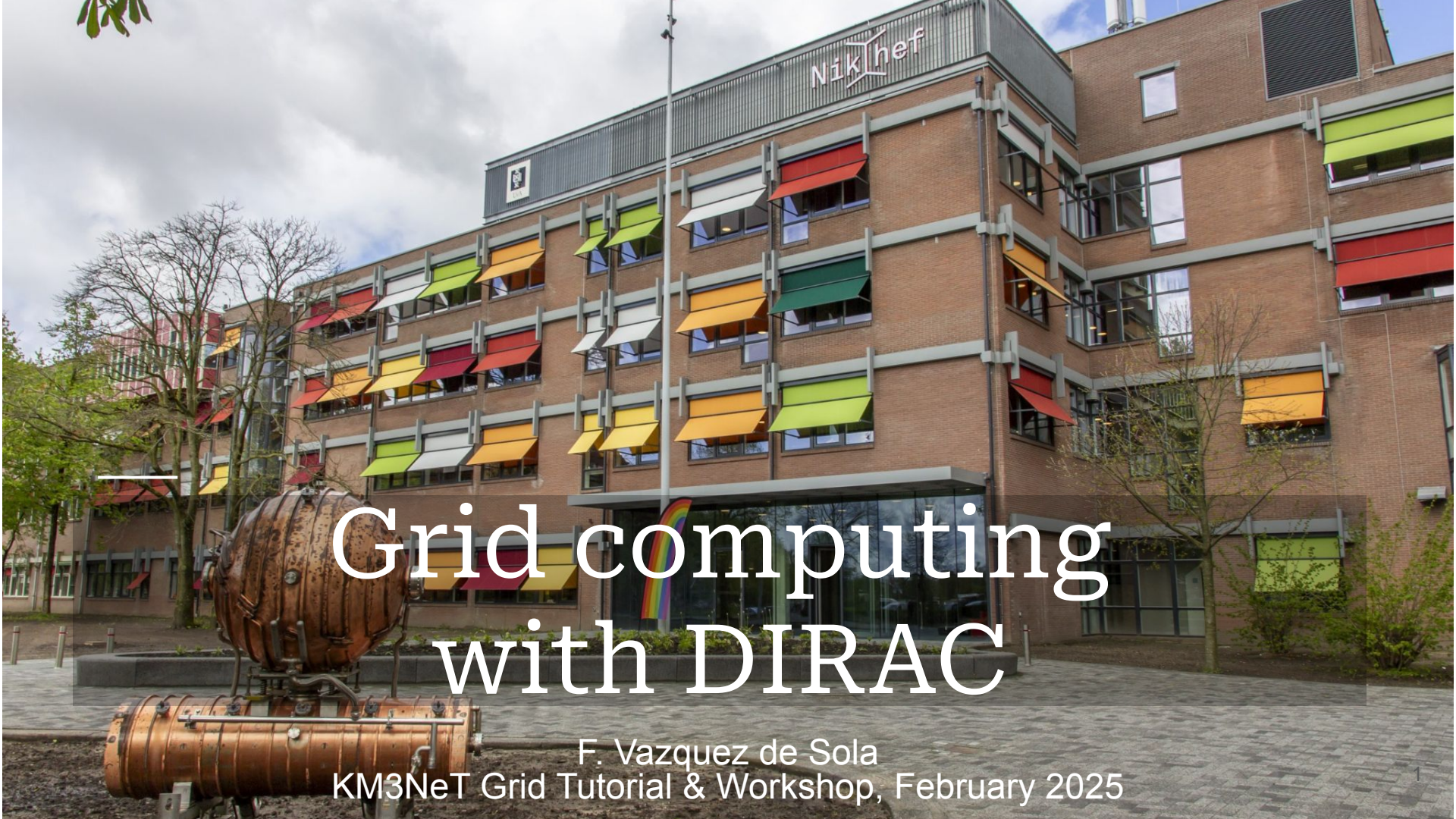



# Grid computing with DIRAC

F. Vazquez de Sola

KM3NeT Grid Tutorial & Workshop, February 2025

# Goals

Familiarize yourself with the key concepts and resources of Grid computing. Learn how to submit basic jobs on the grid.

# From Clusters to Grid

| **Cluster**: | **Grid**: |
| --- | --- |
| ● One site<br>● Shared local storage with home account<br>● Username/password based authentication<br>● Relatively homogeneous hardware<br>● Direct job submission | ● Multiple sites<br>● No shared storage<br>● Certificate/Token based authentication<br>● Heterogeneous hardware<br>● Job submission through middleware |

Scaling out requires some form of scheduling (e.g. Torque, SLURM, HTCondor)

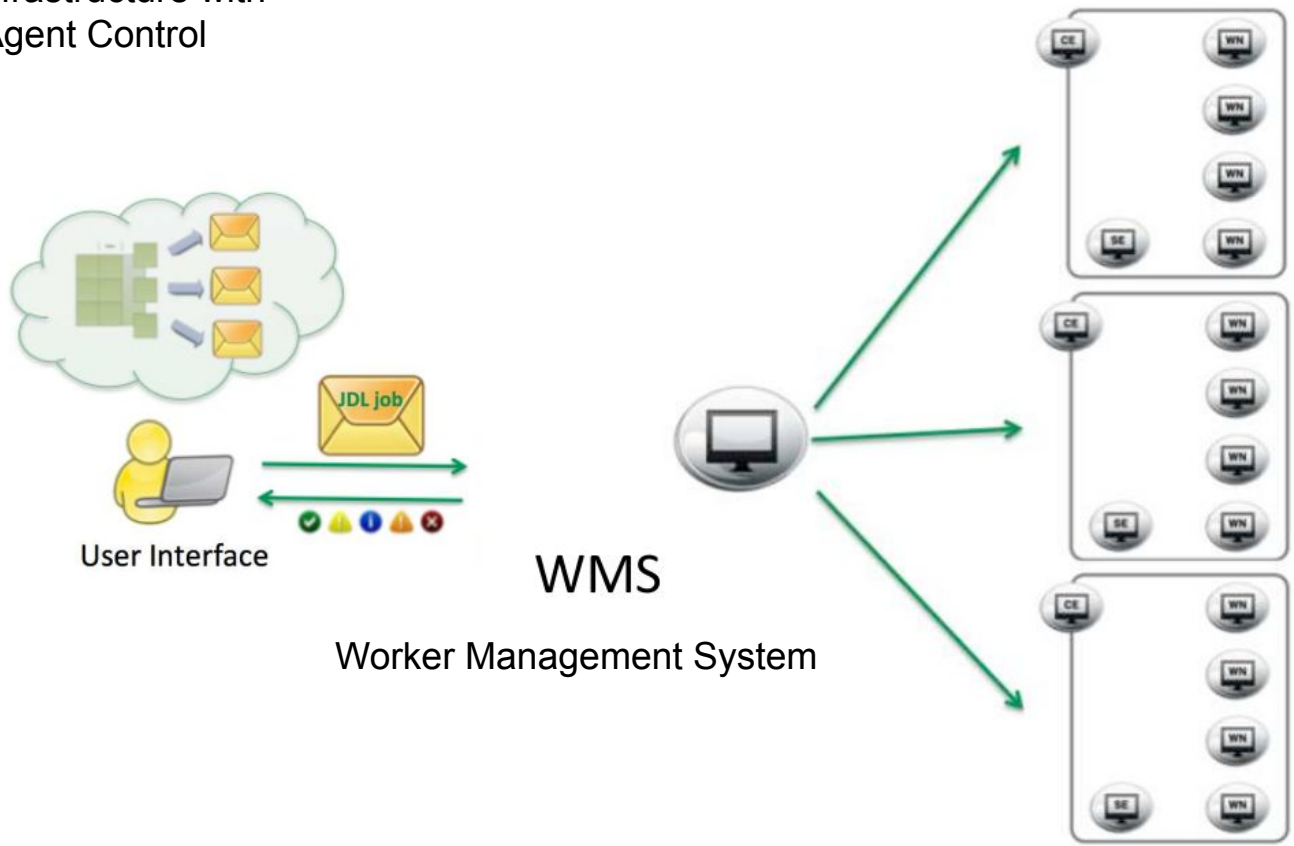
- Spreading load across machines
- Prioritization & queuing
- Accounting & monitoring
- Tasks dependencies

# Grid Infrastructure

# DIRAC

Distributed Infrastructure with
Remote Agent Control

DIRAC middleware mediates between user and Grid resources.

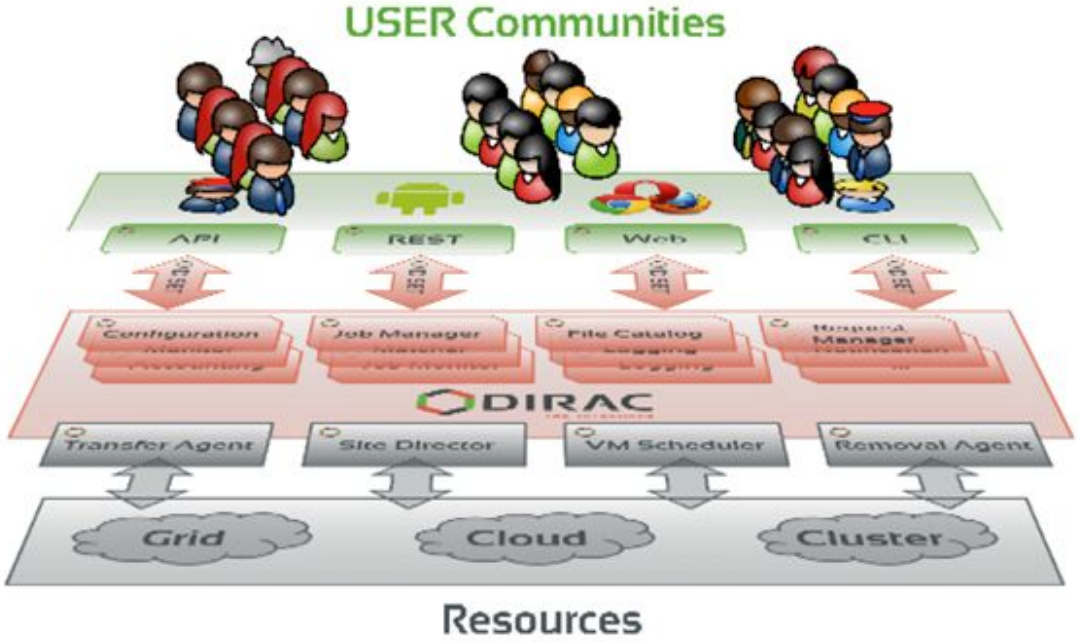


# DIRAC

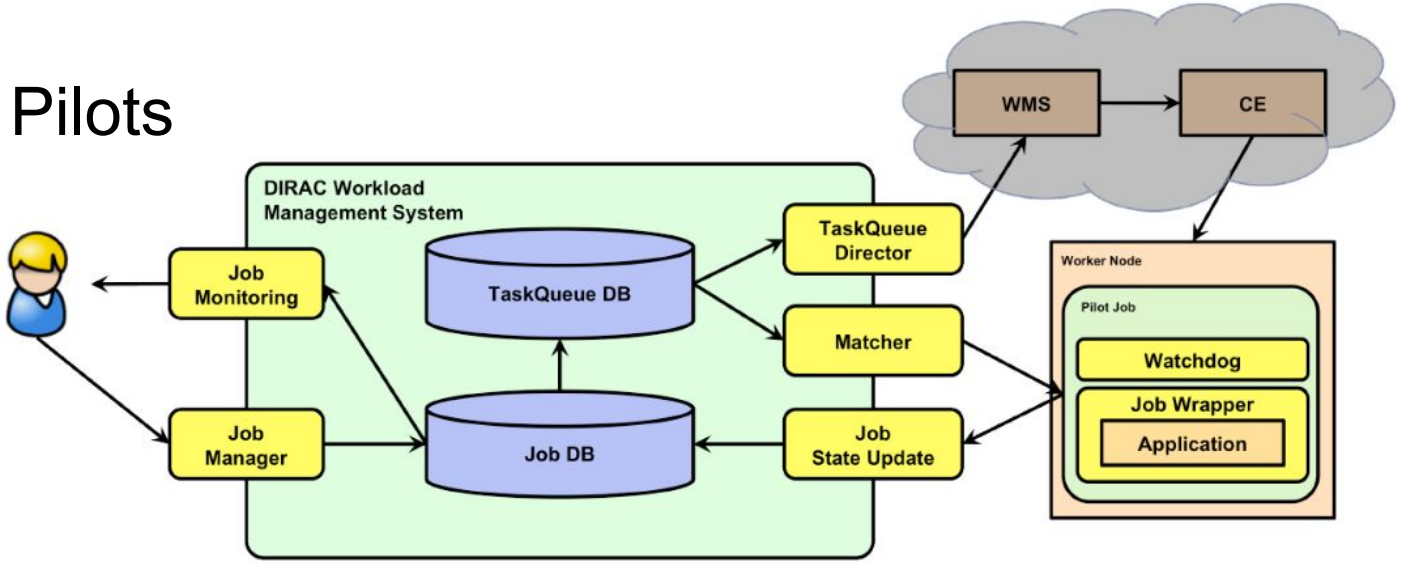
Distributed Infrastructure with
Remote Agent Control

DIRAC middleware mediates between user and Grid resources.

Manages your proxy, submits your jobs to “free” sites, monitors their progress, recovers outputs…



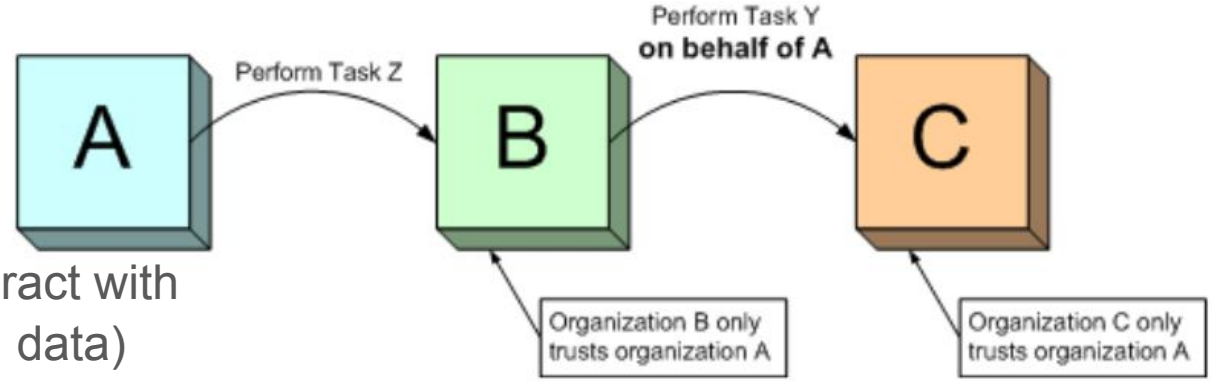
# DIRAC Pilots




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

- Minimize resources spent setting up
- Simplifies discovery of free resources (pull vs push system)
- Allows advanced workflow management

# Certificate Proxies




- Jobs often need to interact with the grid (e.g. download data)
- Delegation:
  - Create proxy
  - New key pair
  - Limited lifetime
  - A generates a proxy with a mutually agreed key pair with B. Then B acts on behalf of A and C who trusts A, trusts also B.
- Single Sign On!
  - Less pass-phrases

# Job lifecycle

| Source | Status | MinorStatus |
|---|---|---|
| JobManager | Received | Job accepted |
| JobPath | Checking | JobSanity |
| JobSanity | Checking | JobScheduling |
| JobScheduling | Waiting | Pilot Agent Submission |
| Matcher | Matched | Assigned |
| JobAgent@GRID.SURF.nl | Matched | Job Received by Agent |
| JobAgent@GRID.SURF.nl | Matched | Submitting To CE |
| JobWrapper | Running | Job Initialization |
| JobWrapper | Running | Downloading InputSandbox |
| JobWrapper | Running | Application |
| Job_307222 | Running | Application |
| Job_307222 | Running | Application |
| JobWrapper | Completing | Application Finished Successfully |
| JobWrapper | Completing | Uploading Output Sandbox |
| JobWrapper | Completing | Output Sandbox Uploaded |
| JobWrapper | Done | Execution Complete |

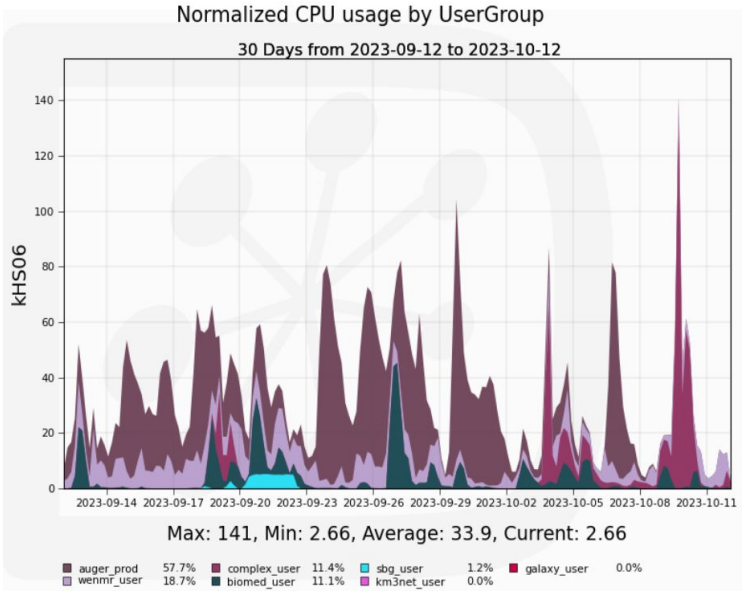
Dirac takes care of two types of tasks with respect to your jobs:

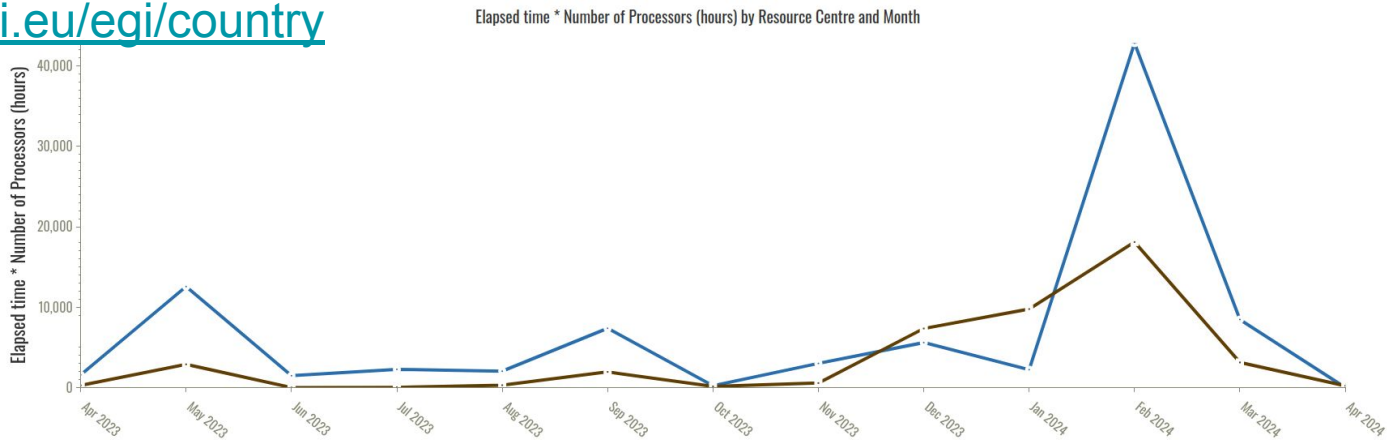
- Job handling:
  - Task distribution to CEs
  - Logging and Bookkeeping service
  - (Small) Output management
- Match-making
  - CE must fulfill the JDL requirements
  - Effective scheduling to Grid queues
  - Load balance

# 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/



# Job Description Language (JDL)

- A file describing what to execute on the grid
- Contains following information (possibly more):
  - Executable to run
  - Executable parameters
  - WN requirements (cores, memory…)
  - Max runtime requirement (**in s@HS06**)
  - File to redirect stdout & stderr
  - **In- & output sandbox:** only “direct” data exchange with UI. <10MB

Dummy example

```
Executable=”run.sh";
StdOutput="stdout.txt";
StdError="stderr.txt";
InputSandbox={”run.sh"};
OutputSandbox={"stdout.txt","stderr.txt"};
CPUTime=500;
NumberOfProcessors=4;
```

For files above 10MB, need to set up up/download inside job itself (more on that later)

# Sites, according to EGI DIRAC config

**Available EGI.SARA.nl queues**

MaxRAM: 48251

| | | |
|---|---|---|
| nordugrid-slurm-short: | MaxCPUtime: 240, | (eff: x10.2), Cores: 1 |
| nordugrid-slurm-medium: | MaxCPUtime: 2'160, | (eff: x10.2), Cores: 1 |
| nordugrid-slurm-long: | MaxCPUtime: 5'760, | (eff: x17.2), Cores: 1 |
| nordugrid-slurm-extreme: | MaxCPUtime: 7'210, | (eff: x17.2), Cores: 1 |
| nordugrid-slurm-long-8cores: | MaxCPUtime: 5'760, | (eff: x17.2), Cores: 8 |

>> OBSERVED NORMALIZATIONS: 21-31

**Available EGI.NIKHEF.nl queues**

MaxRAM: 49152

| | | |
|---|---|---|
| nordugrid-torque-short: | MaxCPUtime: 1'000, | (eff: x15), cores: 1 |
| nordugrid-torque-short7: | MaxCPUtime: 237, | (eff: x15), cores: 1 |
| nordugrid-torque-medium: | MaxCPUtime: 2'138, | (eff: x15), cores: 1 |
| nordugrid-torque-medium7: | MaxCPUtime: 2'138, | (eff: x15), cores: 1 |
| nordugrid-torque-long: | MaxCPUtime: 1'400, | (eff: x15), cores: 1 |
| nordugrid-torque-long7: | MaxCPUtime: 5'759, | (eff: x15), cores: 1 |
| nordugrid-torque-long-8cores: | MaxCPUtime: 5'759, | (eff: x15), cores: 8 |

>> OBSERVED NORMALIZATIONS: 26-31

**Available EGI.IN2P3-CC.fr queues**

MaxRAM: 128195

| | | |
|---|---|---|
| condor (cccondorcexx): | MaxCPUtime: 1'152, | (eff: x10.2), cores: 1 |
| condormc (cccondorcexx): | MaxCPUtime: 1'152, | (eff: x10.2), cores: 1 |
| condor (cctbcondorxx): | MaxCPUtime: 14'400, | (eff: x10.2), cores: 1 |

>> OBSERVED NORMALIZATIONS: 19.7

**Available at EGI.CPPM.fr**

MaxRAM: 144674

| | | |
|---|---|---|
| nordugrid-Condor-grid: | MaxCPUtime: 1'152, | (eff: x10.2), cores: 1 |

>> OBSERVED NORMALIZATIONS: 15.6

**Available at EGI.CNAF.it (cert or token)**

MaxRAM: 1768 (ce02-ce06) or 4096 (ce07)

| | | |
|---|---|---|
| condor (ce02-ce06): | MaxCPUtime: 1'152, | (eff: x10.2), cores: 1 |
| condor (ce07): | MaxCPUtime: 1'200, | (eff: x10.9), cores: 1 |

>> OBSERVED NORMALIZATIONS: 13-24

**Available at EGI.RECAS-NAPOLI.it**

MaxRAM: 8000

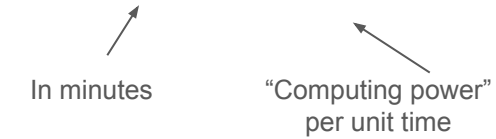
| | | |
|---|---|---|
| condor: | MaxCPUtime: 3'456, | (eff: x7.5), cores: 1 |

>> OBSERVED NORMALIZATIONS: 21-30

**Available at EGI.BARI.it**

MaxRAM: 144674

| | | |
|---|---|---|
| condor: | MaxCPUtime: 1'152, | (eff: x10.2), cores: 1 |



# Hands-on part!

# Basic operations

- submit jobs for execution
- cancel jobs (kill vs delete)
- query the status of jobs and retrieve their output
- copy, replicate and delete files from the Grid

# Using DIRAC from UI machine

- Connect to UI machine with access to CVMFS
- Set up environment:

$ source /cvmfs/dirac.egi.eu/dirac/bashrc_egi #“configures” you for EGI server

- Generate (VOMS-extended) proxy and upload it to DIRAC server

$ dirac-proxy-init --group km3net_user -b 2048 --VOMS

- Clone the repository with the examples:

$ git clone git@git.km3net.de:workflow-management/grid-tools.git

# Exercise

**Write a JDL and script**

Inspire yourself from Test folder in the cloned repository

**Submit job**

Check status (command line / dashboard)

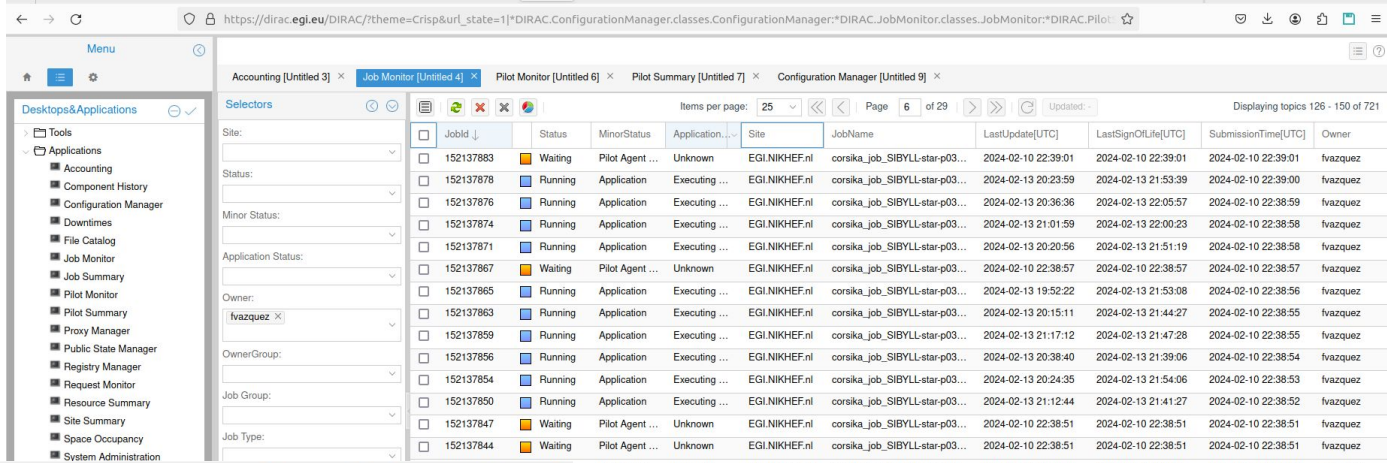
Recover output (command line / dashboard)

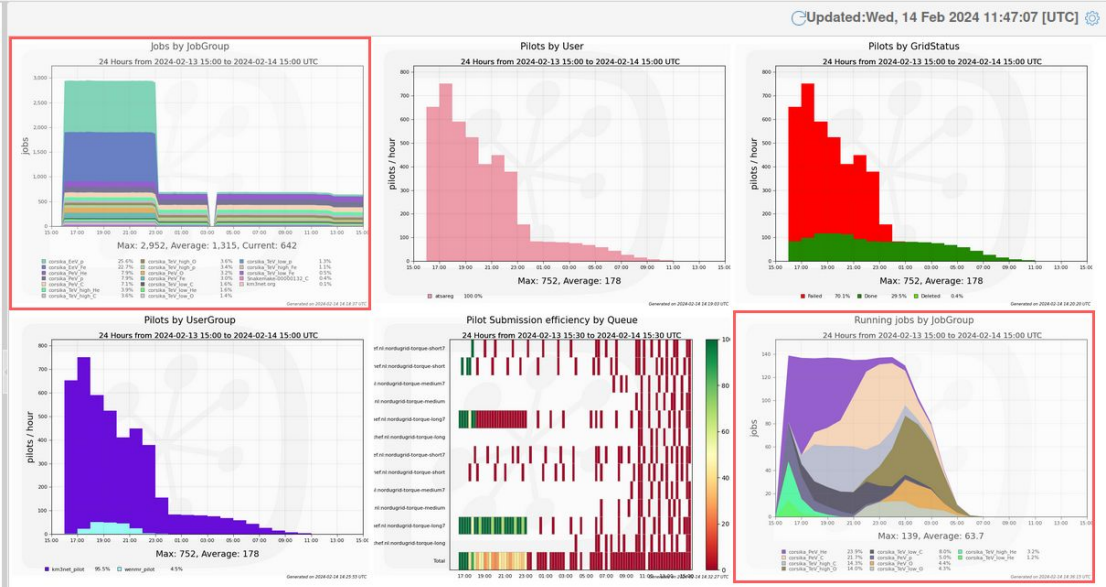
Submit, monitor and recover output from job. CLI examples:

- **dirac-wms-job-submit** job.jdl
- **dirac-wms-job-status** <jobid>
- **dirac-wms-job-get-output** <jobid>

# DIRAC: Dashboard

Web interface for monitoring and logging your jobs




https://dirac.egi.eu/DIRAC/s:Dirac-Production/g:km3net_user/

Find the job you submitted.
Did it run yet? Where?

# Using DIRAC: API

Abstracts away the JDL construction with Python!

Try to submit the same job with the python API. Inspire yourself from the Test folder again!

- Try using Status.py and Output.py to track results, instead of Dirac CLI commands.

You can also run job locally for testing, by using `mode='local'` in submitJob()

fvazquez@ui:~/gridtest/PythonApiTests/firstjobs

```
#!/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'] )

```

# Other considerations: Software distribution

- CernVS File System (CVMFS)
  - Software deployment across Grid sites
  - Available from all Worker Nodes
- Use containers / environments
  - Singularity, conda, (micro)mamba
  - KM3NeT software already available: JPP, Snakemake, CORSIKA… (Thank you Mieke B.!)

No need to set up main software tools in each job. Only job-specific/handler scripts. But not everybody can / should be able to upload software to CVMFS!

# Example: CORSIKA

- Executables on CVMFS, in their respective containers
  - '/cvmfs/km3net.egi.eu/singularity/corsika-CCCVVV_HHHHHH', where CCCVVV and HHHHHH are the Corsika version and Hadronic model respectively
- Based on Piotr K. code, split production into many 2-10h jobs. For each, write new input datacard and handler script, then submit to DIRAC
  - Add rudimentary log information to handler script (host information, wall time, cpu used…)
  - Most of the additional code is just command line arg parsing and grid storage checks. No conceptual difference with running on a cluster
- Store all output files and log files to Grid storage (through RUCIO!) at the end of each job

# How to use

- Already got the scripts when downloading grid-tools, no further setup necessary beyond DIRAC and `export RUCIO_CONFIG=/path/to/rucio.cfg`
- Run:

$ ./start_corsika.py -p $PrimaryName -H $HadronicModel -P $Prod -R $runnumber -F [-N 1 -n 1000] -r ‘workshop’

( can launch multiple at once with $ ./launch_Corsika_FullProd, but please don’t do that now)

E.g PrimaryName = ‘p’ , HadronicModel = ‘SIBYLL’, Prod = ‘TeV_low’, runnumber = ‘0’

File outputs and log files saved through Rucio.

- Check outputs with Rucio

**e.g.** rucio list-dids CORSIKA_testing:

Try to submit a CORSIKA job, see if it succeeded!

Tip: reduce the requested CPU time under 10 minutes

# DIRAC: Other Features

Transformation Management System
- Workflow management system, similar to snakemake; potential to oversee productions and perform error recovery (more tomorrow?)

Data Management System (“File Catalog”) & Request Management System
- We are using RUCIO instead

Monitoring network
- Tools to check current grid status

…Other stuff?

# SITES

# Conclusion

# When to use Grid computing?

“Embarrassingly” parallel problem (e.g. Monte Carlo, processing lots of events):

- Easily partitionable in independent parallel tasks / calculations (jobs)
- Benefit from submitting hundreds, thousands of jobs simultaneously
  - Large scale computational problems / data processing
  - Can still take advantage of parallel processing

*But!*

- *A single grid job is slow*
- *Scheduling and bookkeeping overhead*
- *Need to prepare task for grid environment / make portable code*
- *Need to implement logging steps and deal with errors (not always easy…)*
- *Need to be a bit mindful of limitations of (different) sites*
- *Need to first ask if we have the resources available :)*

# Extra slides

# Glossary : Jobs

- **WMS (Workload Management System):** distributes and manages tasks across computing and storage Grid resources.
- **Job:** a program that will run somewhere on a Grid machine.
- **JDL (Job Description Language):** describes job, parameters and requirements.
- **In/Output Sandbox:** Input Sandbox defines any names of files to be uploaded. Output Sandbox contains filenames of data to retrieve after the job is done. Limited to <10MB files.
- **Pilot:** job that runs on a WN to set up the environment and fetch jobs from a central task queue as required (see DIRAC, PanDA, PiCaS)

# Glossary : Infrastructure

- **UI (User Interface):** your access point to the Grid. From here you submit the jobs and retrieve logging information.
- **CE (Computing Element):** is the interface to the cluster. Accepts jobs via a batch system (LRMS) and dispatches them to a collection of site Worker Nodes. The jobs are submitted by the users either through a WMS or directly.
  - LRMS (Local Resource Management System): it is the batch system on a CE
- **WN (Worker Node):** the machines that do the actual work & execute jobs.
- **Middleware:** software that mediates between users and Grid resources. Covers a variety of roles (DIRAC, Globus, gLite, dCache…)
- **BDII (Berkeley Database Information Index):** information system which holds information about all services available on the grid. Hopelessly outdated, unfortunately.

# Glossary : Certificates

- **CA (Certification Authority):** Signs the certificates to certify your identity.
- **RA (Registration Authority):** is a local authority that signs your request form for a Grid certificate and verifies your identity.
- **VO (Virtual Organization):** is a user community with geographically distributed people that have common objectives or projects.
- **VOMS (VO Membership Services):** handles authorization of resource access, based on the VO membership of the user
- **Proxy:** a short lived temporary certificate which is used for the actual authentication to Grid services.

# Glossary : Authentication

- **GSI (Grid Security Infrastructure):** uses public key cryptography (or asymmetric cryptography) to enable secure authentication and communication over an open network. GSI is based on:
  - **X509: digital certificate.** Users are authenticated to the Grid with a digital X.509 certificate. It is issued by a Certification Authority (CA), which is trusted by the infrastructure running the middleware.
  - **SSL (Secure Sockets Layer):** communication protocol for secure authentication with extensions for single sign-on and delegation.
  - **Tokens** (...to be expanded)
- **Proxy:** a short lived temporary certificate which is used for the actual authentication to Grid services.

# Glossary : Storage

- **SE (Storage Element):** is a Grid component where files can be stored. At least one SE per site.
- **SRM (Storage Resource Management):** manages the storage resources. It is a middleware service providing capabilities like migration from disk to tape. Various implementations like DPM, dCache, CASTOR, StoRM.
- **File locations** (Note LFC solution != RUCIO solution):
  - **GUID (Grid Unique Identifiers):** the identifier assigned to an uploaded file to Grid
  - **LFN (Logical File Names):** a generic file name. This is assigned to a file directly by the user
  - **SURL (Site Unique Resource Location):** physical location of a file on the Grid (disk or tape)
- **LFC (Local File Catalog)** e.g. LCG or DIRAC's DMS: it is a database that holds a mapping to associate LFN, GUID, SURL and replicas. A generic file name (LFN) has an identifier (GUID) and this filename is mapped on to one or more physical locations on the Grid (SURL).
  - RUCIO works differently, need to integrate with DIRAC

# Authentication & Authorization

# Certificate trust chain

# Certificates

- Certification Authority
  - Which? Check either
    - https://www.eugridpma.org/members/worldmap/
    - https://www.igtf.net/pmamap
- Set up instructions in https://ca.dutchgrid.nl/tcs/
- 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):
  - User roles and privileges in a VO
  - Manages certificate extension
  - VOMS server returns attributes (eg. membership)