Pipeline2 – EGEE Grid interfacing

- Today's minigrid
- EGEE grid procedures
- Procedure implemented & tested by Pisa team
- Interface P2-EGEE proposal
- Suggestions for implementation

Today's Pipeline2 distributed system

Java Packages:

@ SLAC:
org-glast-pipeline-client
org-glast-pipeline-server

@ SLAC and CC-IN2P3:
org-glast-jobcontrol with 2 branches: LSF and BQS

Launching procedure:

From WEB interfaces or client machines, through pipeline command at SLAC:

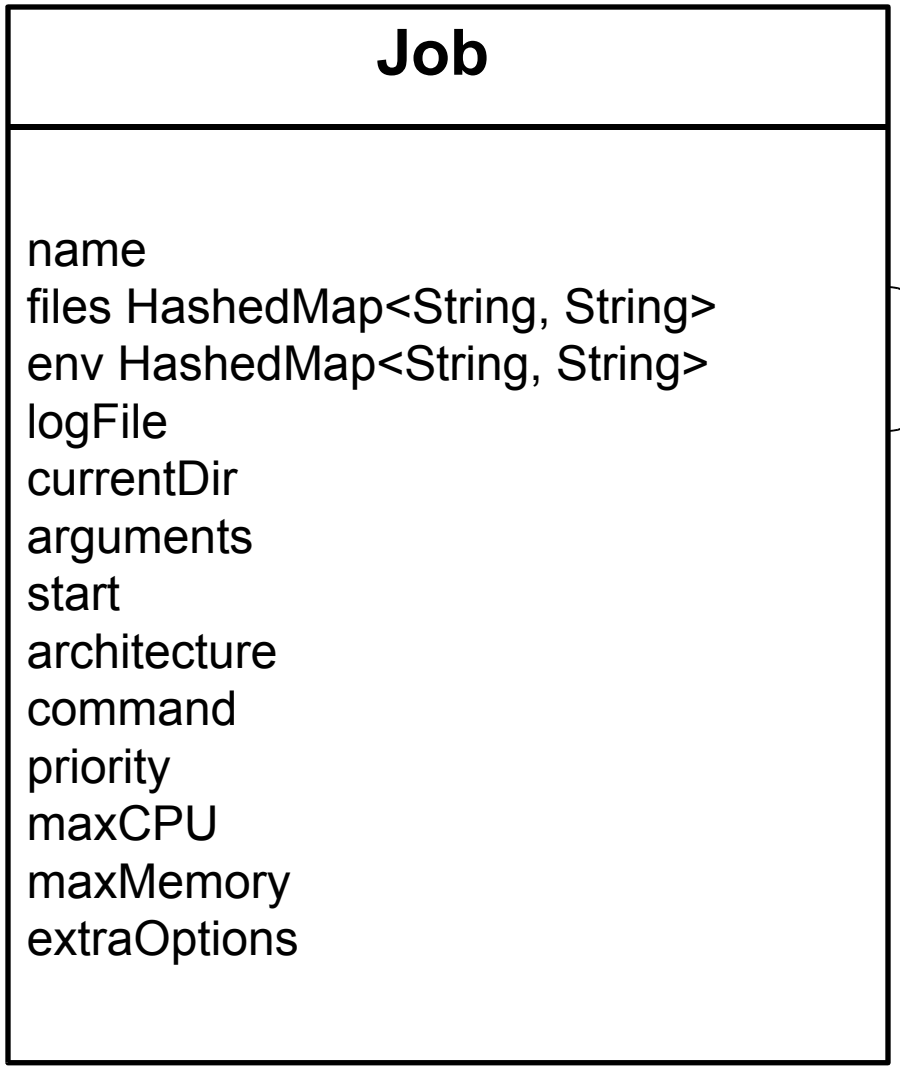
> pipeline → org-glast-pipeline-client

Build streams from taskConfig and runMonteCarlo.py:

~glast/pipeline-II/prod/pipeline load taskConfig.xml

~glast/pipeline-II/prod/pipeline createStream <taskName> -n <number-of -Streams>

Create streams → create Job Objects



Filled out partially at SLAC trough:
org-glast-pipeline-server

and partially on deamon side through:

org-glast-jobcontrol :
LSFJobControlService (SLAC)
BQSJobControlService (Lyon)

Job instances go to deamons (Java RMI method)…

- **Deamons: LSFJobControlService or BQSJobControlService**

  - Create file system (one Output directory per stream)
  - Put files needed to launch job into the stream directory
  - Build submit command (different command if at SLAC or at CC IN2P3)
  - Launch job

- **Files needed to send the job to batch farm and register streams (~the same for LSF or BQS, stored in SPS system for IN2P3 tasks):**
  - **Task level (config directory, need to be at SLAC & at CC IN2P3):**

    taskConfig.xml --> Task definition given by user (located at SLAC & CC IN2P3)
    runMonteCarlo.py --> Task definition given by user
    allGamma.txt: JO File --> given by user (required by runMonteCarlo.py)
    mcRegistration.py
    runWrapper.sh --> attached to Job in SLAC side

  - **Stream Level (one output directory per stream):**

    bqs_script --> built by BQSJobControlService
    pipeline_wrapper: attached in SLAC side (org-glast-pipeline-server)
    script: attached in SLAC side (org-glast-pipeline-server, from taskConfig.xml contents)
    pipeline_env: from job.env varialbe, attached in SLAC side line-server(?)
    outFiles.list: defined in runMonteCarlo.py & taskConfig.xml

The files @ SLAC or CC IN2P3

- **Libraries and additional Files (in AFS, SPS or XROOTD @ CC IN2P3):**
  - **GlastRelease (binary)**
  - **GLAST_EXT**
  - **Calibration Files**
  - **Pointing History Files**
  - **OverlayFiles**
  - **GPLtools + GPL (Python Scripts)**

- **Files produced by job (in SPS @ IN2P3, stream output directory):**
  source_info.txt
  checksum.txt
  logscan.txt
  DONE.success
  pipeline_summary
  logFile.txt

- **Output ROOT files are sent back to SLAC XROOTD servers directly from workers scratch space using BBFTP**

- **logFiles are sent to SLAC by an recursive procedure launched asynchronously with respect to jobs lifeTime, and independent from Pipeline2 interfaces**

EGEE Grid: glossary

UI: user interface
CE: computing Element (entrance to a Worker Node cluster)
WN: worker node
RB: ressource broker (WMS: workload management system)
SE: Storage element
VO: virtual organisation (corresponding to a research community)

One site corresponds to a CE (mandatory) allowing the access to a WN cluster, linked to some central EGEE information services. In addition, some sites offer SEs.
Each site decides which VOs it will support and the priority of this VO in its batch system.

Each VO is responsible of its software management. Software installation in CEs is made using EGEE jobs by VOs software managers.

Pisa team created GLAST VO and ensure software management.
GLAST VO Status is given in Michael Kuss presentation: VO100209.pdf

France-Montpellier site "open-MSFG" (managed at LPTA) will support the VO as soon as it is declared "in production" (~end of February): ~100 cores + some Tb for storage

Launching jobs using EGEE

- **UI → RB (WMS) → CE supporting VO → Output stored in EGEE SEs**

- **UI: authentication procedures (personal certificate), proxy renewal (passphrase entered manually).**
  - **It is possible to ask for generic certificates to certification authorities (has been made for biologist communities).**
  - **It is possible to manage proxy renewals without any manual input: passphrase can be extracted from private key. Other more secure procedures are also available (to implement and test)**

- **WMS system decides which one of the CEs will run the job. We can also force the job destination CE.**
  - **CEs efficiency is very variable from one CE to another but also for one single CE during a time interval. Some applications have been developed to qualify the sites in real time in order to improve production efficiency.**

- **Storage space reserved to a VO in EGEE SEs is reachable from every UIs through proxy associated to that VO, via LFC (LCG File Catalog, LCG:LHC computing Grid) commands**

Procedure developped and tested by Pisa Team

- **Needed files in EGEE context:**

  glast.org.sh: same file for every stream of a task
  glast.org.jdl: one per stream (to be built by org-glast-pipeline-server ?)
  jobOptionsFile.txt
  sourcesFile.xml

- **Binary Files, Libraries and additional Files for EGEE (need to be available on CEs):**
  - *** GlastRelease (binary)**
  - *** GLAST_EXT**
  - *** Calibration Files**
  - *** Pointing History Files**
  - *** OverlayFiles**

- **Jobs are launched from an UI, using 'bulk' job collections built via shell scripts.**

- **Output ROOT and log files stored in a structured file system in SE and sent back to SLAC by procedures independent from launching jobs ones.**

Interface P2 – EGEE proposal

- **Keep taskConfig.xml file as parameterization basis, new job type: EGEE (in addition to SLAC and IN2P3)**
- **Keep the Structured file System @ CC IN2P3**
- **Keep JobControlService Deamon at CC IN2P3**
- **use EGEE UI environement @ Lyon (generic certificate & proxy manual or automatic renewal, if possible) to launch Job.**
- **Completely separate job launching from data retrieval**

Some implementation suggestions:

- **org-glast-pipeline-server: the files attached to the Job instance should be different for EGEE**
  - → a new branch to be added to org-glast-pipeline-server (?).
    The branch choise could be based on "type" variable value in taskConfig.xml file.

- **A new branch to be created in org-glast-jobcontrol pakage (?):**
  - **EGEEJobControlService** (equivalent to LSFJobControlService and BQSJobControlService) **including submit(), getStatus() and cancel() methods, corresponding to EGEE commands**

- **Jobs should produce status files during life time (equivalent to DONE.failed or DONE.success file), easly reachables on SE through an UI.**

- **A sensor could then be in charge of status files monitoring and trigger bulk (for the whole task) data retrieval (ROOT files and logFiles). Such a sensor should be installed in an UI (IN2P3 or SLAC?):**
  - **If in IN2P3, a first stage @ CC IN2P3 (sps/output directories) then to SLAC XROOTD via bbftp (ROOT files) & to SLAC NFS systems (logFiles) via rsync**
  - **If UI at SLAC (still to be tested), a stage could be implemented, then to XROOTD servers.**