PILOT FRAMEWORKS

Joachim Schultes
Bergische Universität Wuppertal

Overview

- Introduction
- Motivation for pilot jobs
- Definition of pilot jobs
- Disadvantages (mostly site-related)
  - Security, Scheduling, Accounting, Traceability
- Examples of pilot driven workload management system
  - DIRAC
  - PanDA

Introduction

- Pilots are minimal scripts
- They can steadily be submitted by a central instance to all clusters for a specific VO
- They run (at the moment) via a static user mapping (all pilots belong to one DN)
- They were established, as the traditional push-model „User → Scheduler → Cluster“ did not scale on a „LHC-wide“ basis
- Their aim is to maximize the job efficiency while minimizing unsuccessful runs.

Motivation for pilot jobs (1)

- Jobs are not queued several times
- They speed up the start-up in most cases (pre scheduling)
- Enables centralized job submission
- Enables resource brokerage across different sites together with improved logging and monitoring capabilities

Motivation for pilot jobs (2)

- Enables coherent monitoring system
- Current site / node environment might have changed since job submission
- VO pilot jobs allows „live“ intra-VO scheduling fitting to its internal priorities (late-binding)
- Overloading of site BS by heavy user job submission can be avoided

Definition of pilot jobs

- Generic small common (VO specific) script
  - Monitors the local resources
    - If environment is not okay: is does NOT start anything!
  - Requests a job from a central repository fitting to the actual situation
  - Downloads the real job (if existing)
  - Reports the result of job execution
  - May start from the beginning, if queue is long enough
  - Cleans up the workspace and exits

Security issues

- A site administrator has no direct overview whose jobs are running in the cluster
- As pilots downloads the user code after being started, one can not examine the user code a-priori (also true for other approaches, as long wget etc. is not forbidden)
- Users with site-problematic jobs can not been blacklisted separately (only whole VO or at least “all pilots”)

Problems for Scheduling and Accounting

- Local scheduling is annulled as
  - User is a generic one
  - Jobs are prescheduled
  - Expected job execution is hidden to the batch system
  - Job execution time is unknown
- Local accounting can not distinguish between different users
  →Intra-VO accounting has to be done

Problems for the Traceability

- Generic user can cause problems for sites:
  - For legal reasons some sites have to know who the real owner of the job is
  - Many (more than one) pilot jobs of the generic user are executed on the same worker node, makes it difficult to trace the original jobs for debugging

DIRAC

- DIRAC is a distributed data production and analysis system used by the LHCb experiment
  - Includes workload and data management components
  - Was developed originally for the MC data production tasks
  - Extended to data processing and user analysis
  - The goal was:
    - Integrate all the heterogeneous computing resources available to LHCb
    - Minimize human intervention at LHCb sites

DIRAC pilot jobs (1)

- Every DIRAC Pilot Job performs an in situ DIRAC installation including a full download of the most current version of the configuration
- Ckecks the working conditions
  - exact location where execution is taking place
  - available disk space, memory and cpu
  - available Grid environment
  - running platform
- A DIRAC Job Agent is launched…

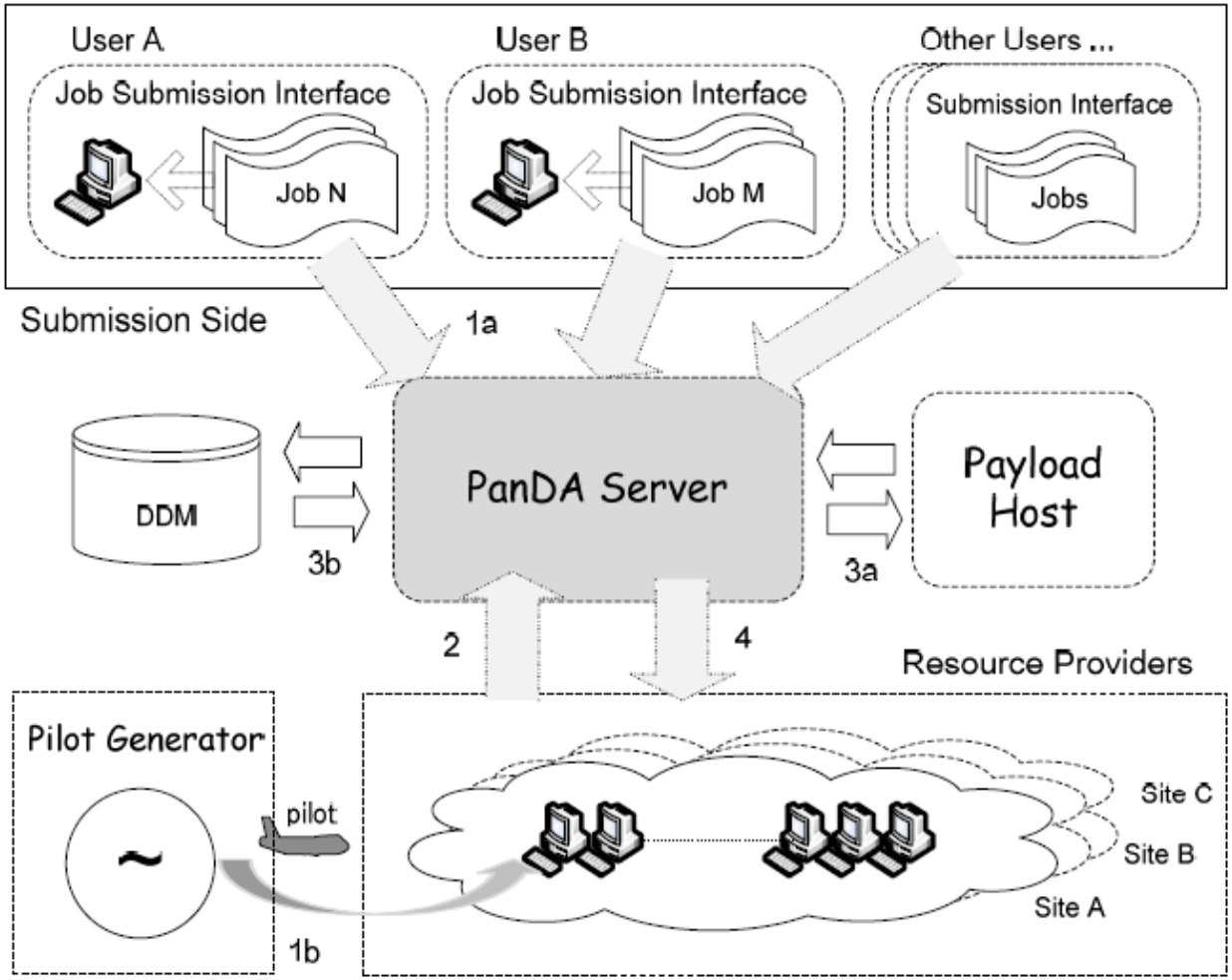
DIRAC pilot jobs (2)

- A DIRAC Job Agent is launched…
  - places the payload request to the central DIRAC WMS server
  - Instantiate a Job Wrapper object
    - execution of the received payload
  - Instantiate a Watchdog
    - Checks periodically the situation of the wrapper
    - takes actions in case the disk or available cpu is about to be exhausted or the payload stalls
    - reports to the central WMS…

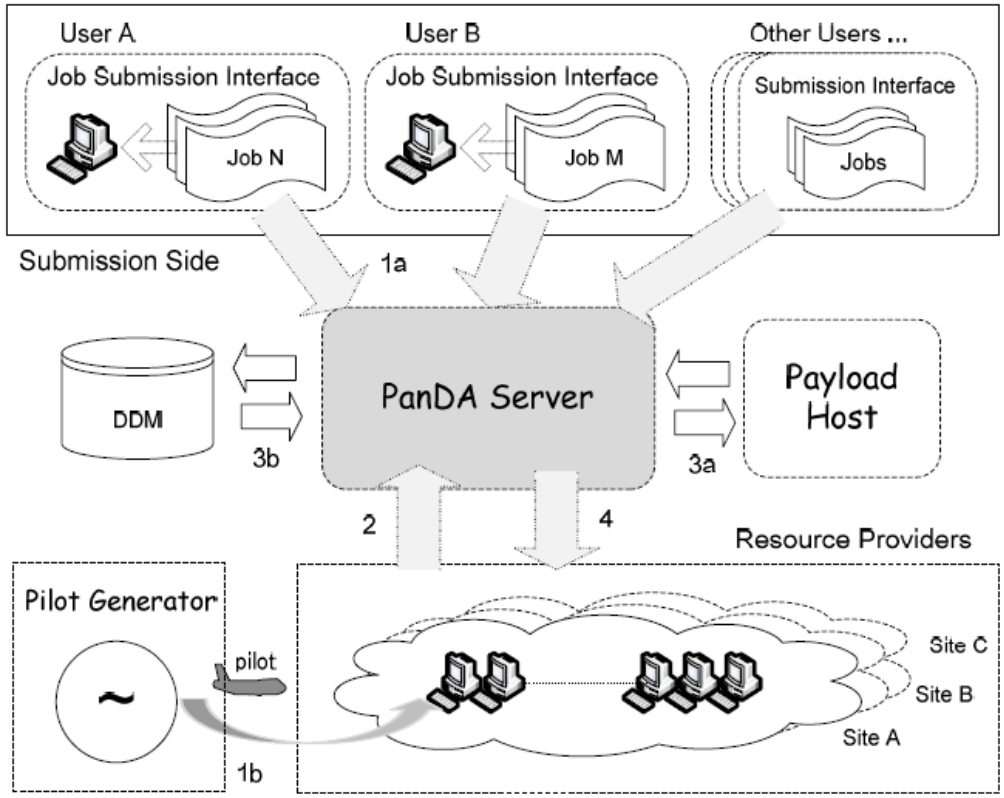
DIRAC pilot jobs (3)

- Instantiate a Watchdog…
  - reports to the central WMS…
  - can also execute management commands received from the central WMS (e.g. killing the payload)
  - retrieves the input sandbox
  - checks availability of required input data and software
  - Executes the payload
  - reports success or failure of the execution
  - and finally uploads output sandbox and output data if required

PanDA system architecture



PanDA Server components



- Task buffer.
  PanDA job queue manager which keeps track of all active jobs in the system.
- Brokerage
  Matching of job attributes with site and pilot attributes. It manages the dispatch of input data to processing sites, and implements PanDA's data pre-placement requirement.
- Job dispatcher
  Dispatcher receives requests for jobs from pilots and dispatches job payloads.
- Data service
  Data dispatch to and retrieval from sites

PanDA Pilot job (1)

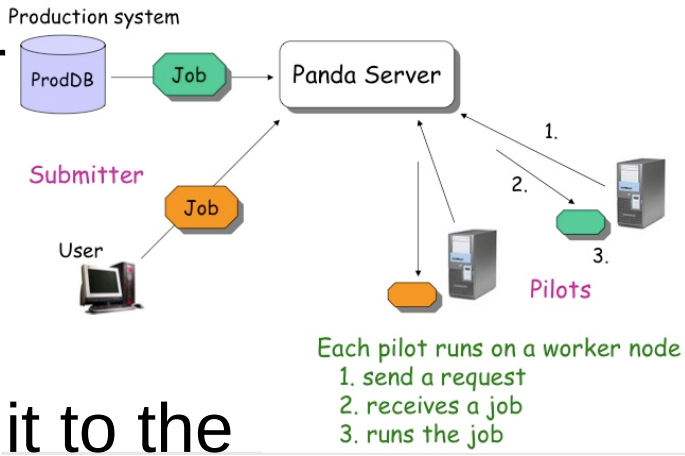
- Starts a wrapper script
  - performs a number of preliminary checks
  - downloads the site configuration from the SchedConfig database
  - Download the main pilot code
- Launch pilot...

PanDA Pilot job (2)

- …Launch pilot
  - scans the local disk for remaining work directories of previously failed jobs
    - → it will try to update the Panda server and create and/or register the log files
    - Remaining output data files will be transferred to the local SE
  - it collects information about the worker node
  - sends it to the job dispatcher…

PanDA Pilot (3)

- …sends it to the job dispatcher
- pilot will fork a separate thread for the job and start monitoring its execution
- creates a job log and transfers it to the local storage element (SE) which is then available from the PanDA monitor
- In case of transfer problem, the pilot reports the error to the dispatcher (error code and a relevant extract of the payload and pilot log files)



PanDA Autopilot

- generic implementation of the PanDA pilot and pilot-scheduler
  - including both ATLAS-specific and general pilots and schedulers
  - governs the submission of pilots to target sites (currently via Condor-G, although other methods are possible).
- using a number of database tables to automatically manage information on queues, pilot wrappers, etc

PanDA Pilot factory components

- gLidein launcher
  responsible for the dynamic deployment of gLideins to eligible sites
- gLidein monitor
  detects occasional failures due to site's temporary downtimes, upon which it then invokes the gLidein launcher to perform another gLidein deployment
- pilot generator
  that disseminates pilots through the schedd gLideins running on remote resources
- Debugging facilities
  Provides debugging information to shift crew via a web server

Future Plans of Panda

- Switching to gLexec ()
- Enable CREAM CEs (upgrade of underlying Condor-G)
- Maybe pilot factories can be avoided in the far future

Further informations

- Job Submission Comparison
  https://twiki.grid.iu.edu/bin/view/Documentation/JobSubmissionComparison
- DIRAC
  https://twiki.cern.ch/twiki/bin/view/LHCb/DiracProjectPage
- PanDA
  http://www.usatlas.bnl.gov/twiki/bin/view/AtlasSoftware/PanDA.html
- Panda Pilot
  https://twiki.cern.ch/twiki/bin/view/Atlas/PandaPilot
- PanDA Pilot Factory
  http://www.usatlas.bnl.gov/twiki/bin/view/AtlasSoftware/PilotFactory.html