Paul Nilsson

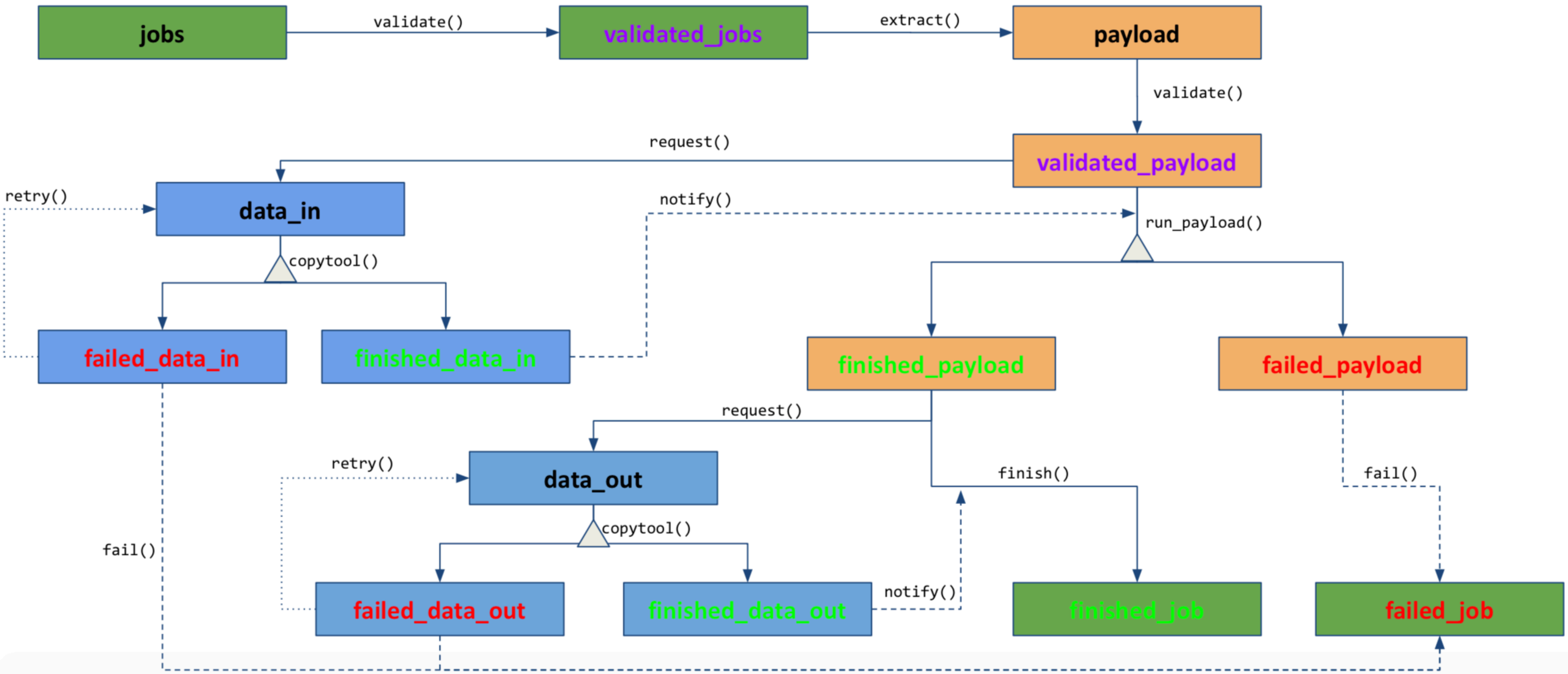
PILOT 2

Pilot 2 Project Overview

- Decision to replace the aging PanDA Pilot was taken in 2016
  - Old Pilot has served us well but after over a decade of development it has become difficult to keep building on
  - Pilot 2 is a complete rewrite
- Development of Pilot 2 is already in its second year and is going forward
  - Development is often done in sprints
  - Note: Pilot 1 is still often highest priority
- Component Model chosen last year
  - Project is in implementation stage since early this year
- First components delivered in March 2017
  - Harvester is using Pilot 2 Data API with stage-in/out (rucio)
  - Other APIs to follow soon (later slide)

Internal Flow of the Jobs Objects

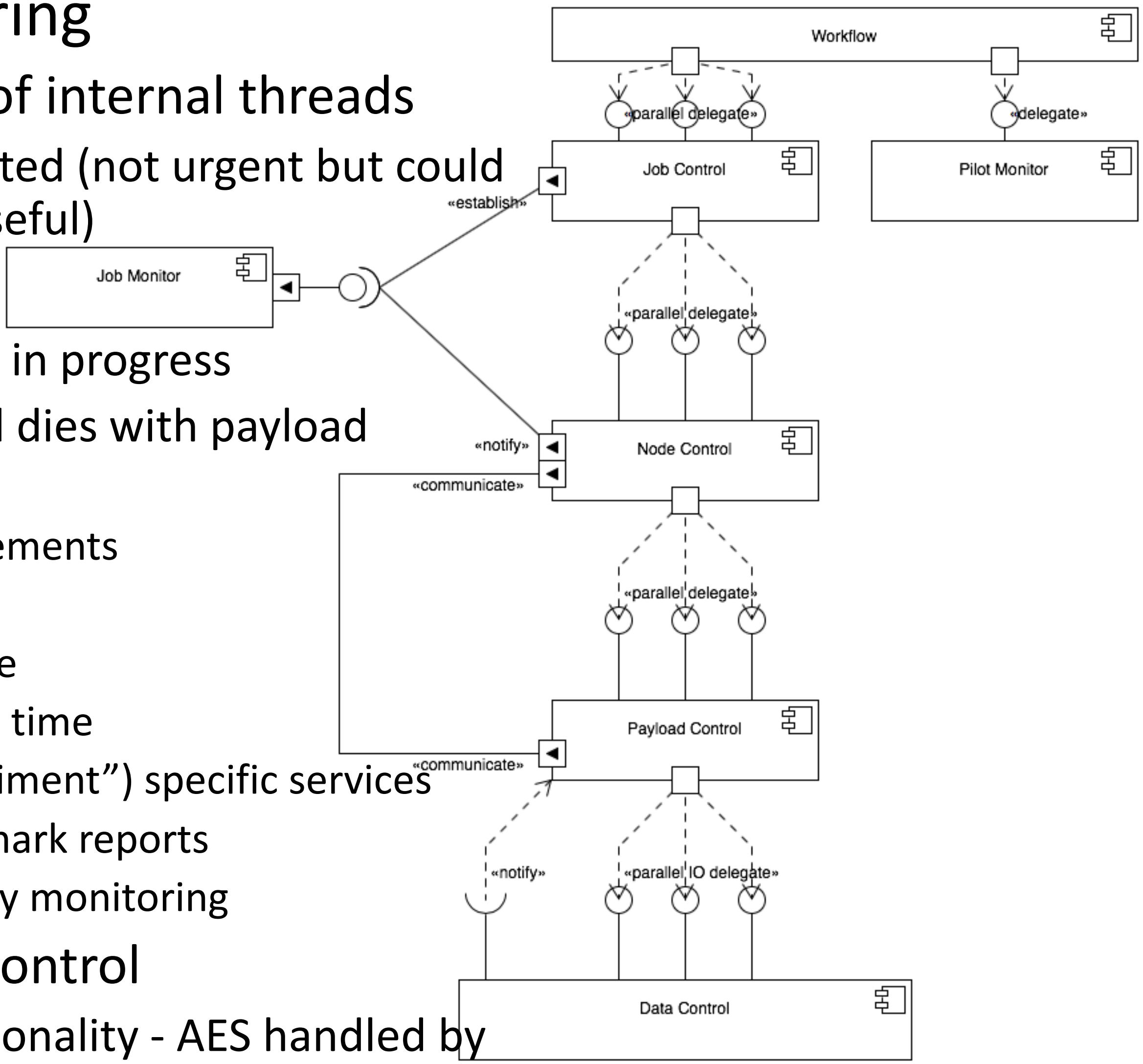
- Job objects are kept in a job queue and are handled by the different pilot components



M. Lassnig

Component Model Updates

- Extended monitoring
  - Pilot monitoring of internal threads
    - To be implemented (not urgent but could eventually be useful)
  - Job monitor
    - Implementation in progress
    - Thread lives and dies with payload
      - Heartbeats
      - Size measurements
      - Looping jobs
      - Proxy lifetime
      - Pilot running time
      - User ("experiment") specific services
        - » Benchmark reports
        - » Memory monitoring
  - Removed Event Control
    - No loss of functionality - AES handled by dedicated workflow module



Pilot Options (1/2)

- Implemented support in current development version (not final, might very well be changed)

Option	Description
-a	Pilot work directory
-d	Debug mode
-w	Force work flow; generic (implemented), event service, production or user jobs
-l	Maximum life time
-q	Queue name
-s	Site name (only needed by the dispatcher)
-j	Job label (user, managed, ptest)
--cacert	CA cert (not used on the grid)
--capath	CA cert path (not used on the grid)

Pilot Options (2/2)

Option	Description
-p	PanDA server port (default stored in config file)
--config	Config file path (default config stored in pilot source)
--countrygroup	Country group for getJob request
--workinggroup	Working group for getJob request
--allowothercountry	Is the resource allowed to be used outside the privileged group?
--allowsameuser	Multi-jobs will only come from the same taskID
--url	PanDA server URL (default stored in config file)
--pilotuser	Pilot user, e.g. name of experiment (used to select user specific code stored in special pilot directory)

- More options will be implemented when they are needed

Pilot 2 APIs

- Some Pilot functionality is exposed to external users by APIs; currently being planned for, or is already available
  - **Data API**

    DELIVERED

    - Basic stage-in/out already used by Harvester
    - New request: asynchronous stage-in/out
  - **Communicator API**
    - Functions for communicating with PanDA server, Harvester, aCT, ..
    - API defined; contains functions for downloads/updates of jobs and event ranges
  - **Environment API**
    - Interface to the job execution environment on HPCs
  - **Services API**
    - Possible new API which could expose functionalities related to services run by the pilot (being discussed), see later slides
  - **Container API**
    - Possible new API, see next slide

Container Support

- Work in progress
  - Usage of containers discussed in separate session (how exactly to implement; tests done with Pilot 1)
- Pilot 2 container module defined
  - May later be used via Container API in case of interest
    - Could be used e.g. by wrapper
    - Can be delivered soon if needed
  - Currently consists of single function for command execution
    - Function can be used to wrap/decorate a command with timer, time-out, container command, ..

Utilities

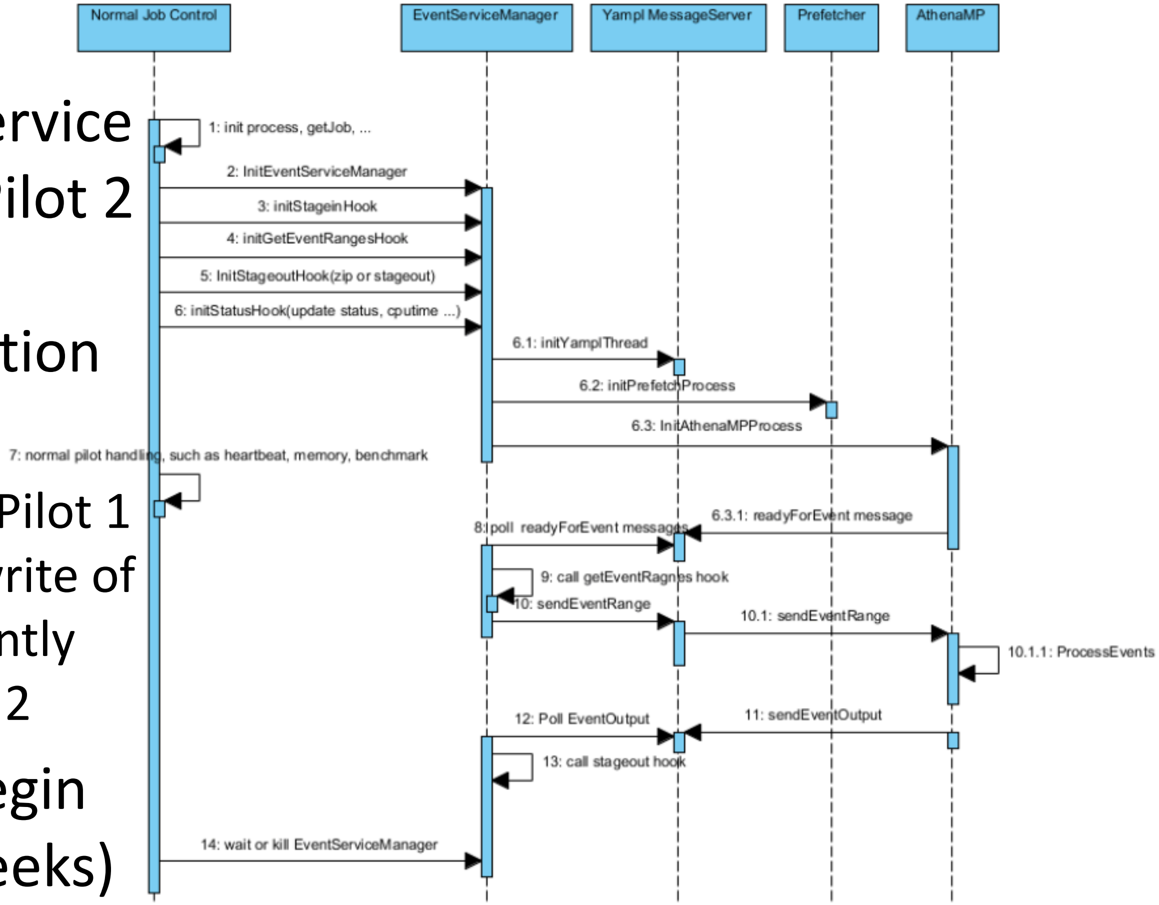
- Pilot 1 has hundreds of major and minor functions
  - A large part of Pilot 2 development is to re-implement many of these (cut and paste in some cases)
- Pilot 2 has utilities organized in dedicated util/ folder
  - Current code base include functions in multiple modules
    - E.g. constants, disk, filehandling, https, information, ..
      - Preliminary information module presents interface to AGIS and schedconfig
        - » To be replaced by a full Information Service component (where AGIS/schedconfig are not hardcoded but accessed via user code)
        - » Development to start as soon as possible
    - Pilot 2 now supports standard configuration files
      - Config files are shipped with pilot source (default values), but can be preplaced either in /etc or in init directory

Copy Tools and Data API

- Copy tools rucio and xrdcp copy are supported in Pilot 2 development version
  - Only rucio is available in current production version Data API
- We want to provide functionality for non-Rucio users/experiments
  - Current xrdcp copy tool implementation relies on rucio for PFNs
    - Can be made independent if pilot is given PFNs from somewhere else
  - Restructuring / refactoring of the code is underway to facilitate adding additional copy tools (< 2 weeks)
- About to start development for supporting additional copy tools
  - gfal-copy, lsm (mv/storm could be considered special cases of lsm)
- Support for asynchronous stage-in/out has been requested (currently only basic stage-in/out is implemented)
  - To be developed asap (to be decided)

Event Service in Pilot 2

- A new algorithm / workflow for event service is being planned for Pilot 2 based on years of experience and evolution of AES
  - AES development for Pilot 1 continues but the rewrite of the algorithm is currently only planned for Pilot 2
- Implementation to begin shortly (within 2-3 weeks)



Normal Job Control
EventServiceManager
Yampl MessageServer
Prefetcher
AthenaMP
1: init process, getJob, ...
2: InitEventServiceManager
3: initStageinHook
4: initGetEventRangesHook
5: InitStageoutHook(zip or stageout)
6: initStatusHook(update status, cputime ...)
6.1: initYamplThread
6.2: initPrefetchProcess
6.3: InitAthenaMPProcess
7: normal pilot handling, such as heartbeat, memory, benchmark
6.3.1: readyForEvent message
8: poll readyForEvent messages
9: call getEventRagnes hook
10: sendEventRange
10.1: sendEventRange
10.1.1: ProcessEvents
11: sendEventOutput
12: Poll EventOutput
13: call stageout hook
14: wait or kill EventServiceManager

Wen Guan

Benchmarking on HPCs and Tier-0

Step #1

- Expose benchmarking functions (for execution, output dictionary manipulations, reporting, ..) in new Pilot 2 Services API
  - Note: pilot adds some info not present in the default benchmark dictionary, and removes some fields that are not wanted

Step #2

- ***Idea #1***:
  - Harvester can add benchmark step to worker / MPI rank #0 which will execute it before running the actual job
- ***Idea #2:***
  - Introduce new Benchmark_tf.py transform that allows for having dedicated benchmark tasks running on the grid / HPCs whenever needed [ok with Graeme Stewart and Marcelo Vogel]
  - **Complications**: 1) want to run benchmark suite with same number of cores as payload, but if there's no Athena payload in these jobs then what (re-run the task) ; 2) we loose a valuable HPC queue slot for a mere benchmark job
  - **Advantage**: Allows for benchmarking on Tier-0 where pilot is not normally running
- When job has finished, Harvester will discover the output benchmark dictionary in the shared file system and again use Pilot 2 Services API functions to report results to intermediary Elastic Search service (which populates ES)
  - On Tier-0, corresponding wrappers take benchmark results from the transform and upload them post-facto using pilot functions [ok with Graeme Stewart and Marcelo Vogel]

Memory Monitoring on HPCs (Yoda)

Step #1

- ***Idea***:
  - Expose memory monitoring functions (for execution, parsing, ..) in new Pilot 2 Services API
    - More useful for Yoda than standard HPC job since Yoda should be able to monitor each rank, while this is not true for standard HPC job

Step #2

- Yoda adds memory monitor step to worker / MPI rank #0 which will execute it in parallel with the actual job
  - Memory info available during running, can be reported to Harvester and PanDA server
- When job has finished, Harvester will discover the final memory monitoring dictionary in the shared file system and again use Pilot 2 Services API functions (to add it to jobReport.json / upload it to intermediary ES service)

Pilot Wrappers

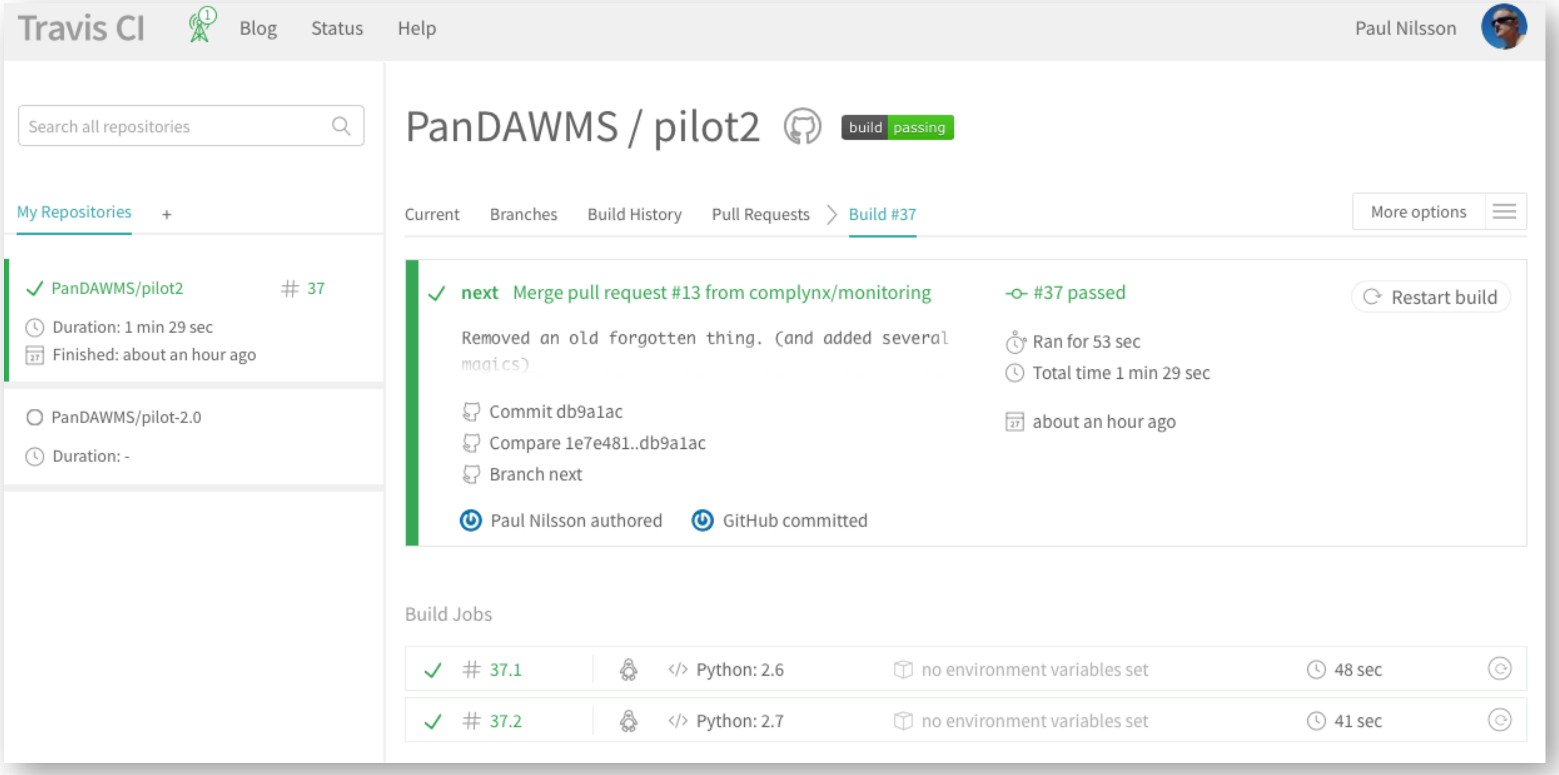
- Known changes for production wrappers
  - Will need to migrate to use new pilot options
    - Details to be discussed/worked out
    - Wider grid testing towards the end of the year (2017)
    - HC tests in (early?) 2018
  - Be able to run pilot within container
    - Being discussed
    - Possibly use Pilot 2 Container API (Singularity and Docker?)
- Development version of Pilot 2 uses special dev wrapper
  - Practical to have our own wrapper
    - E.g. Pilot 2 uses different/new pilot options, has ES functionality, and the wrapper is minimal
    - Currently located in personal GitHub repo(s)
  - Currently used for tests on EU and US resources
  - Not intended for production use or as replacement of other wrappers

MiniPilot

- Special pilot version for minimalistic workflows
  - Paper: http://ceur-ws.org/Vol-1787/197-201-paper-33.pdf
  - Intended for testing during early Pilot 2 development
  - Note 1: we do not intend to make major developments on this code - others may modify it as they like [see below..]
- Recently migrated into final Pilot 2 GitHub repository
  - Currently using its own Pilot modules, i.e. no current usages of Pilot 2 code
- Special GitHub branch for version used by Harvester
  - minipilot-harvester [to be created shortly]
  - Developed by D. Benjamin, T. Childers

GitHub Technology

- Pilot 2 GitHub is using Travis CI for automatic code verification/validation and unit tests
  - GitHub pull request into Pilot 2 repo triggers external service (runs pep8, flake8 and unit tests)



PanDAWMS / pilot2

✓ PanDAWMS/pilot2 # 37
Duration: 1 min 29 sec
Finished: about an hour ago

○ PanDAWMS/pilot-2.0
Duration: -

✓ next Merge pull request #13 from complynx/monitoring

Removed an old forgotten thing. (and added several magics)

Commit db9a1ac
Compare 1e7e481..db9a1ac
Branch next

Paul Nilsson authored GitHub committed

#37 passed
Ran for 53 sec
Total time 1 min 29 sec
about an hour ago

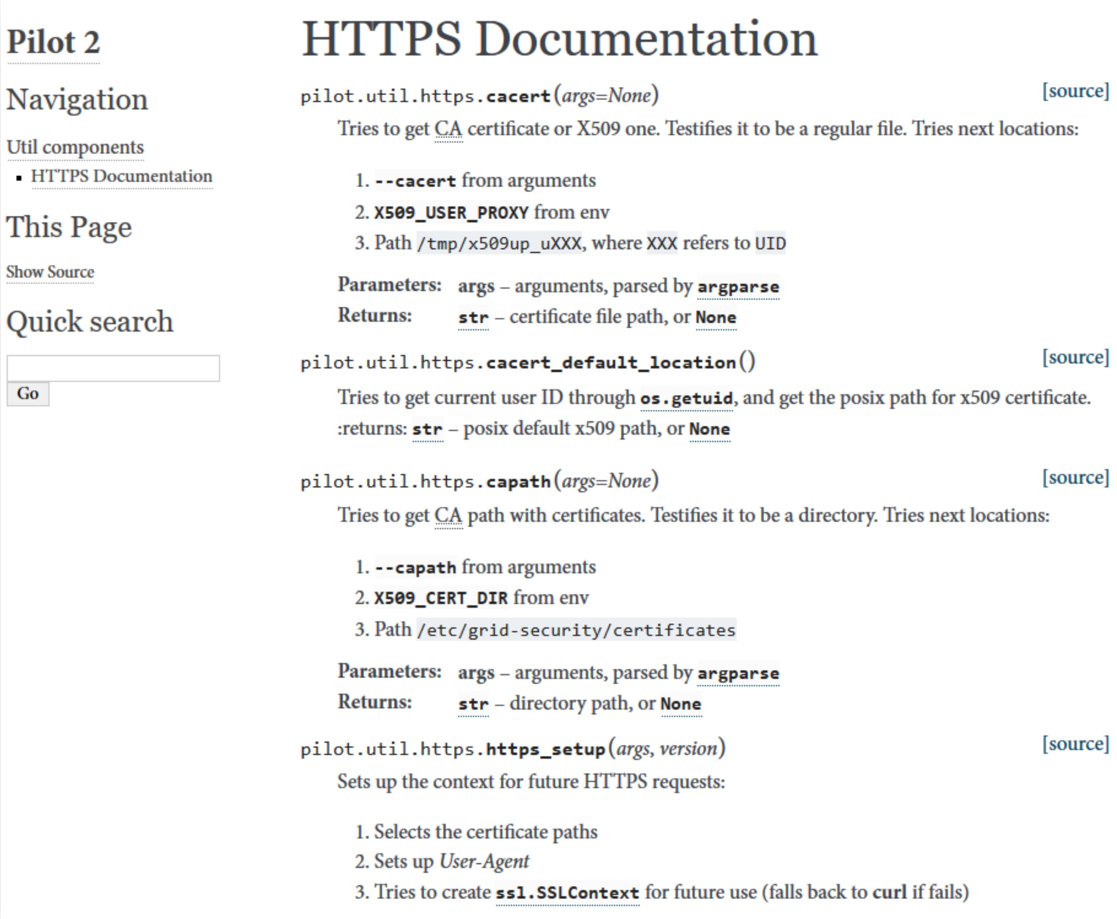
Build Jobs

✓	# 37.1	Python: 2.6	no environment variables set	48 sec
✓	# 37.2	Python: 2.7	no environment variables set	41 sec

D. Drizhuk

Pilot 2 – Sphinx Documentation

- Semi-automatic code documentation using Sphinx
  - Module to be documented must be accompanied by related sphinx file
  - Pull request followed by [currently] local sphinx script execution which builds the documentation
  - Output need to be moved to www server
- Service can be added to GitHub (hosted)
  - To be investigated further



HTTPS Documentation

`pilot.util.https.cacert(args=None)` [source]

Tries to get CA certificate or X509 one. Testifies it to be a regular file. Tries next locations:

1. `--cacert` from arguments
2. `X509_USER_PROXY` from env
3. Path `/tmp/x509up_uXXX`, where `XXX` refers to `UID`

Parameters: args – arguments, parsed by `argparse`
Returns: `str` – certificate file path, or `None`

`pilot.util.https.cacert_default_location()` [source]

Tries to get current user ID through `os.getuid`, and get the posix path for x509 certificate. :returns: `str` – posix default x509 path, or `None`

`pilot.util.https.capath(args=None)` [source]

Tries to get CA path with certificates. Testifies it to be a directory. Tries next locations:

1. `--capath` from arguments
2. `X509_CERT_DIR` from env
3. Path `/etc/grid-security/certificates`

Parameters: args – arguments, parsed by `argparse`
Returns: `str` – directory path, or `None`

`pilot.util.https.https_setup(args, version)` [source]

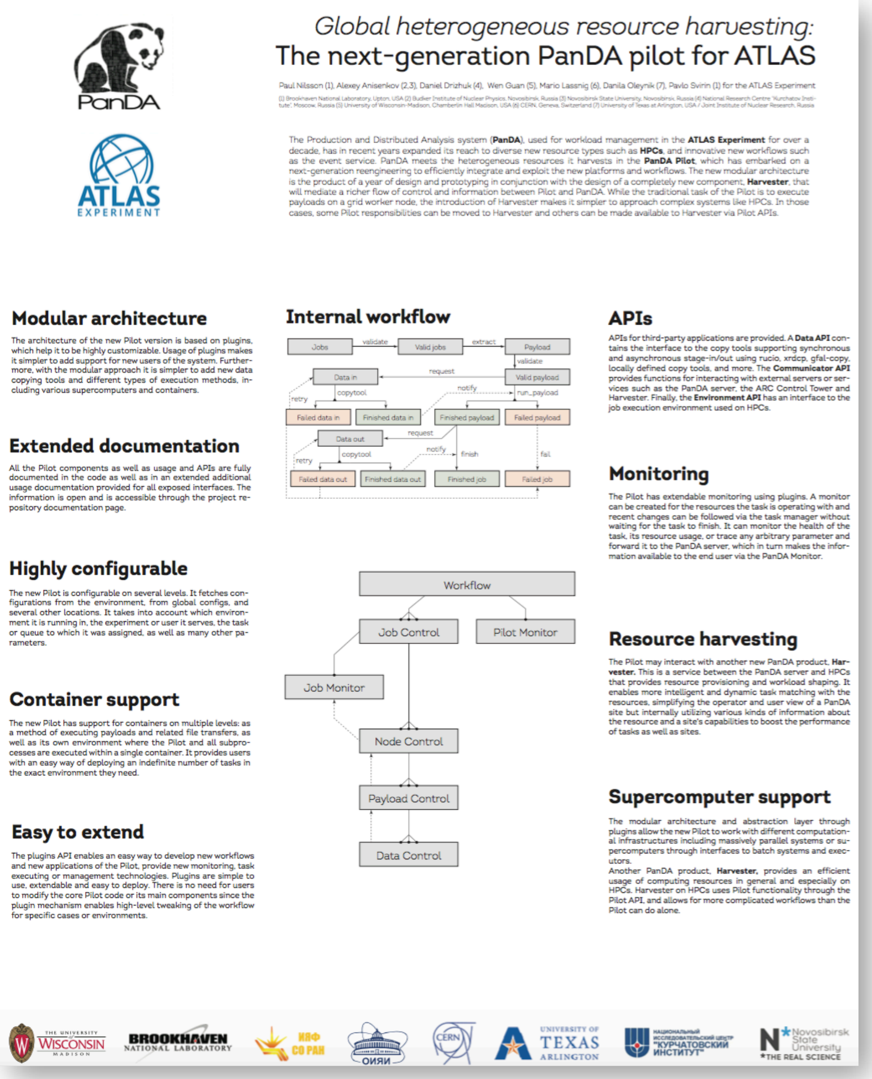
Sets up the context for future HTTPS requests:

1. Selects the certificate paths
2. Sets up *User-Agent*
3. Tries to create `ssl.SSLContext` for future use (falls back to **curl** if fails)

D. Drizhuk

Pilot 2 – ACAT 2017 Poster

- **“Global heterogeneous resource harvesting: the next-generation PanDA pilot for ATLAS”**
- Presented at ACAT 2017, Seattle US
- https://indico.cern.ch/event/567550/contributions/2627120/attachments/1516202/2366285/ACAT_Poster_copy.pdf
- Proceedings paper to be written shortly



Global heterogeneous resource harvesting:
The next-generation PanDA pilot for ATLAS

Paul Nilsson (1), Alexey Anisenkov (2,3), Daniel Drizhuk (4), Wen Guan (5), Mario Lassnig (6), Danila Oleynik (7), Pavlo Svirin (1) for the ATLAS Experiment

(1) Brookhaven National Laboratory, Upton, USA (2) Budker Institute of Nuclear Physics, Novosibirsk, Russia (3) Novosibirsk State University, Novosibirsk, Russia (4) National Research Centre "Kurchatov Institute", Moscow, Russia (5) University of Wisconsin-Madison, Chamberlin Hall Madison, USA (6) CERN, Geneva, Switzerland (7) University of Texas at Arlington, USA / Joint Institute of Nuclear Research, Russia

The Production and Distributed Analysis system (**PanDA**), used for workload management in the **ATLAS Experiment** for over a decade, has in recent years expanded its reach to diverse new resource types such as **HPCs**, and innovative new workflows such as the event service. PanDA meets the heterogeneous resources it harvests in the **PanDA Pilot**, which has embarked on a next-generation reengineering to efficiently integrate and exploit the new platforms and workflows. The new modular architecture is the product of a year of design and prototyping in conjunction with the design of a completely new component, **Harvester**, that will mediate a richer flow of control and information between Pilot and PanDA. While the traditional task of the Pilot is to execute payloads on a grid worker node, the introduction of Harvester makes it simpler to approach complex systems like HPCs. In those cases, some Pilot responsibilities can be moved to Harvester and others can be made available to Harvester via Pilot APIs.

Modular architecture

The architecture of the new Pilot version is based on plugins, which help it to be highly customizable. Usage of plugins makes it simpler to add support for new users of the system. Furthermore, with the modular approach it is simpler to add new data copying tools and different types of execution methods, including various supercomputers and containers.

Extended documentation

All the Pilot components as well as usage and APIs are fully documented in the code as well as in an extended additional usage documentation provided for all exposed interfaces. The information is open and is accessible through the project repository documentation page.

Highly configurable

The new Pilot is configurable on several levels. It fetches configurations from the environment, from global configs, and several other locations. It takes into account which environment it is running in, the experiment or user it serves, the task or queue to which it was assigned, as well as many other parameters.

Container support

The new Pilot has support for containers on multiple levels: as a method of executing payloads and related file transfers, as well as its own environment where the Pilot and all subprocesses are executed within a single container. It provides users with an easy way of deploying an indefinite number of tasks in the exact environment they need.

Easy to extend

The plugins API enables an easy way to develop new workflows and new applications of the Pilot, provide new monitoring, task executing or management technologies. Plugins are simple to use, extendable and easy to deploy. There is no need for users to modify the core Pilot code or its main components since the plugin mechanism enables high-level tweaking of the workflow for specific cases or environments.

Internal workflow

APIs

APIs for third-party applications are provided. A **Data API** contains the interface to the copy tools supporting synchronous and asynchronous stage-in/out using rucio, xrdcp, gfal-copy, locally defined copy tools, and more. The **Communicator API** provides functions for interacting with external servers or services such as the PanDA server, the ARC Control Tower and Harvester. Finally, the **Environment API** has an interface to the job execution environment used on HPCs.

Monitoring

The Pilot has extendable monitoring using plugins. A monitor can be created for the resources the task is operating with and recent changes can be followed via the task manager without waiting for the task to finish. It can monitor the health of the task, its resource usage, or trace any arbitrary parameter and forward it to the PanDA server, which in turn makes the information available to the end user via the PanDA Monitor.

Resource harvesting

The Pilot may interact with another new PanDA product, **Harvester.** This is a service between the PanDA server and HPCs that provides resource provisioning and workload shaping. It enables more intelligent and dynamic task matching with the resources, simplifying the operator and user view of a PanDA site but internally utilizing various kinds of information about the resource and a site's capabilities to boost the performance of tasks as well as sites.

Supercomputer support

The modular architecture and abstraction layer through plugins allow the new Pilot to work with different computational infrastructures including massively parallel systems or supercomputers through interfaces to batch systems and executors.
Another PanDA product, **Harvester,** provides an efficient usage of computing resources in general and especially on HPCs. Harvester on HPCs uses Pilot functionality through the Pilot API, and allows for more complicated workflows than the Pilot can do alone.

Current/Near Term Development

- Full chain of generic workflow running
  - Currently ironing out bugs
  - To be completed asap to facilitate for the more complicated AES workflow
- Event Service
  - In design stage to be implemented shortly
- APIs
  - Either already in design or implementation stage
- Job Client manager
  - Returns the proper job client object depending on the user
    - Needed by APIs
  - Code written, to be migrated into Pilot 2 within weeks
- Data Control extensions
  - xrdcp support recently added, but data module now needs some refactoring before additional copy tools will be added
  - Asynchronous stage-in/out requested (no estimate for delivery yet)
- Function development
  - Ongoing (bulk part of Pilot 2 development)

Time to Completion

- At least one more year of development until Pilot 1 can be fully retired
  - Pilot 1 feature requests are often a top priority and slows down other developments
- Focus now is on supporting existing workflows with more features (e.g. monitoring) and implement new ones (read: event service)
  - In particular, API development is a priority since it provides Pilot 2 functionality before the ‘final’ product is ready
  - Pilot 2 can gradually take responsibilities away from Pilot 1, e.g. event service

Pilot OTP – Jun-Dec (Projected)

* Developers involved with both Pilot 1 and Pilot 2

Name	Time	Task
Paul Nilsson (*)	90%	Project leader; core software and coordination
Danila Oleynik	60%	Pilot HPC, Pilot 2 payload specifics
Wen Guan (*)	30%	OS/Pilot general development
Daniil Drizhuk	25%	General pilot development
Alexey Anisenkov	15%	AGIS/Pilot interactions
Kaushik De	10%	Coordination

(*) Event service contributions not included here but counted separately
NB: Contributions from Mario Lassnig / Tobias Wegner counted as DDM