Design Notes

Prior documents

p3s stands for "ProtoDUNE Prompt Processing System". Supporting documents and an outline of the design can be found in the FNAL DocDB 1861 (authorization required for access).

The Pilot Framework

Similar to other distributed systems, there are the following problems that must be solved in p3s:

- "just-in-time" execution, i.e. minimal latency of job execution after it has been created. This could be a problem on most batch systems which don't guarantee that
- ability to run on diverse, heterogenious and ad-hoc resources (e.g. ad-hoc clusters at CERN, CERN Tier-0 and perhaps some other facilities)
- isolation of the execution environment from failures of worker nodes, and other incidents at job initialization time

The above issues are solved by using a pilot-based framework conceptually similar to what is used in PanDA, DIRAC and other systems. In it, a pilot job (agent) is deployed to the worker node and is registered on the system's server. After that, it is matched to a "payload job" which is then dispatched to the worker node and is managed by the pilot.

p3s vs other systems

The p3s is the computing platform for protoDUNE to support Data Quality Management (DQM). Its requirements and mode of operation are different from those of a typical Workload Management System (such as used in offline production). In particular,

- there are specific ETA requirements ranging from minutes to tens of minutes for various types of processing jobs since for DQM purposes the results become stale (not actionable) rather fast
- only a portion of the data (configurable) needs to be processed; it is assumed that the DAQ and its monitoring system provide stable data taking conditions so it is not necessary to sample most of the data continuously. Further, if necessary at run time, a portion of the data can be dropped (i.e. excluded from the DQM stream) in any stage of processing in order to optimize throughput for critical jobs

- there is no retry/recovery mechanism for failing jobs since any substantial delay in processing a unit of data makes the result less relevant. Instead, output and error logs are recorded and used to debug jobs
- processing jobs/streams are initiated automatically and are triggered by the data arriving from DAQ
- in p3s there is no distinct data handling system for two reasons. First, it is assumed that p3s can access data either locally via a POSIX-like interface, or via XRootD interface with minimal coding and integration effort. Second, a full-fledged data handling system would introduce additional complexity, latency and potentially failure modes. In summary, data handling capabilities are kept to an absolute minimum.

Due to above factors and to guarantee ease of operation, a simplified pilot-based system was created as the core of p3s which leverages Django web application framework and other existing tools, and includes a rather minimal amount of application-specific code.

Client/Server Architecture

The p3s server is a Web service i.e. its API is based on HTTP requests. The user interacts with the system by means of clients which send HTTP requests to the server. This is the only way to change the state of the server and the objects it contains it its database.

The p3s server provides the following set of functionality:

- creates objects e.g. job records in its database, according to messages received from clients which may be actuated by the user or run as a part of an automated script
- changes the state of objects - once again, according to messages coming from the clients
- monitoring of the state of various objects such as jobs, pilots, units of data etc

P3S Objects

Pilot

The pilot process can be started on a worker node via a variety of methods. Once it is live (and optionally performs a variety of possible checks) it contacts the server, which then creates a record in its database. The record reflects of state of the pilot as it goes through transitions.

An example of what states a pilot can go through during its lifecycle is given below:

- *active*: registered on the server, no attempt at brokerage yet
- *no jobs*: no jobs matched this pilot
- *dispatched*: got a job and preparing its execution (may still fail)
- *running*: running the payload job
- *finished*: job has completed
- *stopped*: stopped after exhausting all brokerage attempts.

Job

In the object sense, the p3s job is a record in its database which contains all information necessary to create an actual process on a worker node. This record also reflects the status of the object as it goes through transition.

Data

Currently p3s supports data units as individual files, in future development it may also support the notion of dataset. Data (datasets) are registered on the p3s server by client either actuated by the user or automated processes which create jobs and workflows when fresh data arrives.

Workflow

Workflow model, interface and the corresponding p3s client are described in a separate document (*WORKFLOW.pdf*).

Clients

Most important of the p3s clients are:

- The *pilot* - submission and management of pilot data on the server
- The *job* - submission of job definitions to the server and management of job data on the server
- The *dataset* - registration of data files in the system

Please see the dedicated document ("CLIENTS") for more detail.