Near term workflow for pre-operations with PanDA

William O'Mullane and Richard Dubois and Hsin Fang Chiang

2021-02-03

1 Introduction

We need a workflow system and production tools which can process DESC DC2 for DP0.2. Nominally the processing starts in June 2021. We have a milestone L3-MW-0050 in March (see Table 1) for Batch system installation and configurations on IDF and L3-MW-0060 for the DP0.2 production. The preferable way to do this would be with BPS in front of PanDA but there are potentially other solutions (see Section 5).

For PREOPS we should focus on PanDA in the near term get all the hooks in place and make it work for DP0.2. Getting this in place requires some leadership and decision making. We need a product owner and manager (see Section 3). This is separate from the construction side's HSC reprocessing at NCSA for development needs. The constructions team at NCSA can continue to use Condor-based BPS for the biweekly HSC reprocessing at NCSA.

How effective these tools are will determine how effort-intensive (and successful) the large-scale processing campaigns will be.

1.1 Milestones

General DP0 information is RTN-001. For simplicity some milestones are copied here in Table 1. Jira is the source of truth for dates on these though some may need revising.

Table 1: FY21 Middleware Milestones

Milestone	Jira ID	Rubin ID	Due Date	Level	Team
Read only Gen3 butler for DP0 at IDF	PREOPS-143	L3-MW-0030	2021-03-31	3	Science Users Middleware
Qserv installation on IDF	PREOPS-142	L3-MW-0010	2021-03-31	3	Science Users Middleware
PanDA based workflow system in place	PREOPS-154	L3-MW-0050	2021-03-31	3	Science Users Middleware
DP0.1 data loaded into Qserv on IDF	PREOPS-144	L3-MW-0020	2021-04-30	3	Science Users Middleware
Gen3 butler and pipeline task ready for DP0 production	PREOPS-156	L3-MW-0070	2021-06-10	3	Science Users Middleware
PanDA based workflow system with tooling (e.g. restart) added.	PREOPS-155	L3-MW-0060	2021-06-30	3	Science Users Middleware
Evaluate Batch Production System	PREOPS-153	L3-MW-0040	2021-07-31	3	Science Users Middleware

2 Requirements and priories

LDM-636 forms the formal requirements baseline.

Concisely we need the execution team to be able to run DP0.2 with minimum hand holding. Hence the top priorities for the near term would be:

1. Documentation: preferably on lsst.io, enough for the execution team to kick off pipelines, monitor and to first order troubleshoot them.
2. Workflow monitoring - some sort of web page which gives status (perhaps slightly customized)
3. Restart: Can resume an unfinished workflow. Can automatically retry jobs killed by preemption, DB connection, or other transient issues.
4. Logstash: On IDF this will be Google Logging. Any logging should end up in the same central logging system.
5. Troubleshooting failed jobs: Features to help understand non-transient failures, such as error messages aggregation and ways to reproduce failures. This kind of error usually is caused by pipeline failures and needs follow-up investigation.
6. This first version could be on Google only, though IN2P3 would be a good bonus. It is understood this may use DOMA @ CERN but we assume BNL clear that with CERN.

Longer term (which may not be for DP0.2)we need

- Installation at SLAC
- Multi site execution with France and eventually UK as well as SLAC.
- Campaign execution monitoring

2.1 Timeline

We have a milestone L3-MW-0050 in March (see Table 1) for Batch system installation, documentation and configurations on IDF and L3-MW-0060 for the DP0.2 production. We should

track these two milestones: L3-MW-0050 for an initial system and L3-MW-0060 to have the system to run DP0.2.

2.2 Evaluation

L3-MW-0060 will see the commencement of the processing run - we assume there may be some hiccups at that point. But at L3-MW-0060 + one month we should decide if this is the long term approach for Rubin Operations with DOE buy in. Hence L3-MW-0040 is approximately the evaluation date.

Should the evaluation be positive the next phase would include setting up the back end at SLAC.

3 Team

SLAC obviously have long term interest in this working and on a single track so it would be good to have some SLAC oversight on the topic. A product owner to shepherd requirements and priorities as well as a manager to guide resources must be identified. Currently (all at partial fractions) the team consists of:

- Brian Yanny and team at FERMILAB for execution
- Monica Adamow - Execution NCSA
- Michelle Gower, Mikolaj Kowalik - BPS and deployment
- Sergey Padolski and Shuwei Ye (starting in January)- PanDA

4 PanDA

The PanDA (“Processing and Data Analysis”) system was created by ATLAS at LHC to manage its massive processing efforts. In that capacity it handles around a million processing jobs per day across heterogeneous systems, supporting multiple parallel campaigns. Its main services (PanDA, Harvester, iDDS) are driven from a central database. The system can ingest DAGs,

handle the workflow and then the workload management. Currently PanDA cannot rerun parts of workflow, but the feature is being actively considered for addition.

PanDA satisfies a number of criteria:

- Multi-site authentication
- Multi-site processing - Harvester can be used to mitigate network traffic between sites and central workflow db; also handles site-specific submission properties allowing a range of different kinds of resources
- Manages workflow (via iDDS) as well as workload
- Good monitoring tools for the submitted workflow. Can be customized.

While support would be dependent on BNL expertise, several installations of PanDA have been undertaken outside of ATLAS, so there is experience in doing installs and of ongoing maintenance for other organizations.

In order to demonstrate the viability and customizability of PanDA for Rubin, BNL has set a target of doing processing with PanDA in the IDF by the March 2021 time frame. As a part of that demonstration, they will provide documentation of the PanDA system.

It would be additionally instructive to set up multi-site processing to include the French Data Facility and US Data Facility during 2021.

However, campaign management is outside PanDA's scope, so a layer on ctrl_bps would be needed to chunk up and keep track of elements of a campaign. Ctrl_bps would likely also need to handle resubmissions.

4.1 PanDA backend

PanDA began as a MySQL based system at BNL. It was switched to Oracle at ATLAS/CERN insistence when it was adopted by ATLAS and relocated to CERN in ≈2006. Since then PanDA has benefited from collaborative work with ATLAS Oracle experts to optimize and tune it as its usage and capabilities have grown to smoothly support DB-intensive tasks such as managing

>1M concurrent jobs and fine grained processing. PanDA's DB interfaces are agnostic to the RDBMS back end; it is able to work with other back ends, e.g. a MySQL instance has operated in Amazon EC2 for many years. The issues in using a non-Oracle back end

1. tuning and
2. production use of recent code/capability

Tuning: while not an issue for relatively small scale usage, if operational scale approaches that of ATLAS (as Rubin's will), DB tuning and optimization is important. Dedicated effort and expertise for the chosen back end will be required. Production use of recent code/capability: PanDA is under active development, and Rubin's use cases leverage recent/current developments, e.g. iDDS. That development takes place with Oracle as the back end. In order to promptly use new developments against another back end, an expert on that back end would need to work hand in hand with the developers to test, debug, tune and validate the new functionality.

4.1.1 Oracle features used by PanDA

SLAC, via Stanford, has free use of Oracle but with a given standard feature set. Oracle charges through the nose for additional features. Could you send along the list of features required for PanDA to use Oracle, please? Then we can confirm that our license conforms.

- Partitioning
  - Essential for scalability, particularly as the DB grows over time. Time-wise partitioning ensures that historical data can be accessed promptly and efficiently. This is the only extended functionality used by PanDA itself.
- Optimization tools
  - Compression, transportable namespaces used in DB optimization, administration
- Diagnostics, Tuning
  - Valuable for monitoring and maintaining the database.
- Oracle Data Guard

  - Used for backup. Without it, some alternative robust approach to DB backup would be needed.

CERN is also concerned about minimizing Oracle license costs. One approach they are taking is to use fewer, more powerful servers since the license cost scales with core count but not core power. The ATLAS PanDA production DB will shortly be moving from a 20 core server to a server with 8 slightly more powerful cores. The present server has 512GB memory. Large memory is important to avoid disk reads as much as possible. In the new 8 core server, memory will be increased to 768GB to compensate for the reduced core count. As mentioned below, this configuration provides the system with large headroom for load fluctuations and large spikes.

The ATLAS PanDA Oracle database uses about 75TB to serve PanDA's 15 years of ATLAS data.

4.2 Physical resources required for PanDA at Rubin's scale

ATLAS PanDA resource usage:

- PanDA server: 9 servers x (8 cores, 16GB RAM, 500GB disk)
- JEDI: 9 servers x (8 cores, 16GB RAM, 500GB disk)
  - PanDA/JEDI scale roughly with jobs per day (concurrent job count together with job duration)
  - Configuration is by design far from a performance ceiling, designed to handle large spikes (cf. below)
- PanDA monitoring: 8 servers x (8 cores, 16GB RAM, 80GB) + 1TB shared space for logs storage.
  - scales with operating scale (same as PanDA/JEDI servers) together with history (15 years of data, most of the monitoring usage looking at the last year of data)
- Harvester: 8 x (4 cores, 8GB RAM, 500GB disk)
  - Harvester scales roughly with number of sites
- Condor: 6 x (8 cores, 16GB RAM, 100GB disk)

  - Condor scales roughly with number of job slots (which scales roughly with concurrent core count)

- iDDS: 2 x (8 cores, 16GB RAM, 100GB disk)
  - iDDS is relatively new, its scalability in ATLAS is being exercised in the data carousel use case, in production. We'll learn how it scales in Rubin as we implement and exercise use cases.

We describe here the ATLAS PanDA resource usage. In estimating Rubin resource needs from this, these ATLAS PanDA metrics should be kept in mind:

- Steady state operation is currently 400-450k concurrent cores on average
- ATLAS PanDA scales smoothly to 1.4M+ concurrent cores, exercised during HPC spikes
- Each job communicates a heartbeat to the PanDA server every 15min
- Average job duration is 8 hours and ranges from minutes to a few days
  - Longer (within limits) is better for efficiency, system load and limiting DB growth. Ideal duration is a couple of hours. ATLAS has a lot of short O(10min) jobs, but it's preferable to avoid them if possible.
- About 1M jobs are processed per day on average, peaking at about 2M
- O(1000) users active in the last year, including individual analysis
- O(100) global processing sites
- About 150 PanDA queues globally

4.2.1 JEDI in ATLAS and Rubin

While JEDI plays a role in the ATLAS Production System Prodsys2 that Rubin will not use, JEDI also provides functionality that Rubin will use. (JEDI basically operates as an integral part of the PanDA core.) JEDI works with iDDS to provide task-based orchestration of jobs. A ForcedPhotCoaddTask in the processing pipeline may consist of thousands of jobs. Managing and monitoring them as a consolidated payload that is run with different input parameters makes use

of JEDI. A hundred-job processing pipeline consists of only 14 tasks (DeblendCoaddSourcesSingleTask, MergeDetectionsTask, DetectCoaddSourcesTask, ...), which makes it significantly easier to debug problems at a particular processing step and control the execution progress.

Graph generation is done using the BPS subsystem of the Rubin middleware. Once the graph is defined, the PanDA plugin in Rubin middleware sends the definition to the iDDS server, and at that point PanDA/JEDI takes care of its execution according to the relationship between jobs/tasks and inputs specified in the Butler repository.

4.2.2 Condor usage

Condor has since PanDA's inception been its most important workhorse for submitting pilot jobs and harvesting processing resources. The condor service machines handle the submission of PanDA pilots to sites (condor queues) around the world. In recent years Harvester has been developed to abstract PanDA itself away from the many kinds of execution back ends that are now supported, of which condor is one. Harvester manages the condor pilot job submission activities carried out by the condor service machines.

For ATLAS, with over 100 sites most of which use condor, the scale of the condor submission operation is large. For Rubin with far fewer sites it is a smaller task that will basically disappear inside the Harvester service serving Rubin sites.

4.2.3 Rome - cores and memory

Many ATLAS PanDA components are using 8 core VMs simply because that is the configuration preferred by CERN IT. It is the largest core count configuration CERN supports. From our point of view it would be preferable to have VMs 2x or 4x the size and hence fewer of them. But 128 core/512 GB is a big step in consolidating servers that we have never tried. Horizontal scalability is efficiently supported by having several servers and adding additional as needed. Using many servers also gives – crucially – redundancy and resiliency. You don't want to lose services if a huge server falls over. Apart from these considerations, we see no reason not to try 128 core/512 GB servers. The servers don't have heavy network traffic or disk I/O requirements. In terms of power, we estimate that a 128 core/512 GB server is roughly commensurate with the power Rubin will need in the near/mid term. We assume some standard virtualization layer will be used with this server and the whole setup will be available as

a number of VMs.

5 Potential solutions

Conceptually this is done in two steps: (a) workflow generation and (b) job execution. In step (a) the workflow generation defines executable jobs and job interdependency as a graph. In step (b) job execution includes workflow status monitoring, pausing/resuming/killing workflows, debugging/retrying failed jobs, resource usage monitoring, and relevant toolkits to facilitate execution management on a large scale.

1. ctrl_bps workflow generation + PanDA-plugin execution tools developed by BNL
2. ctrl_bps workflow generation + Condor-plugin execution tools developed by NCSA
3. ctrl_bps workflow generation + Pegasus as the execution tools
4. ctrl_bps workflow generation tools can't work on IDF, one can use customized scripts to generate workflow for any execution tools.

6 Risks and worries

1. Lack of documentation for PanDA: it is a complex system and will be the heart of processing.. operating for 12 years in this mode is unwise.
2. Dependence on an institution or individual because of 1, also suggests the need to spread the expertise more broadly across the team.
3. Having a LOT of scripting to make a production run of any size
4. dependence on Oracle: is an open source project would you not like to depend on commercial products furthermore some of us have had bad experience with Oracle.

7 Q&A

Could you send along the list of Oracle DB features required for PanDA to use Oracle, please?

- **Partitioning.** Essential for scalability, particularly as the DB grows over time. Time-wise partitioning ensures that historical data can be accessed promptly and efficiently. This is the only extended functionality used by PanDA itself.
- **Optimization tools.** Compression, transportable namespaces used in DB optimization, administration.
- **Diagnostics, Tuning.** Valuable for monitoring and maintaining the database.
- **Oracle Data Guard.** Used for backup. Without it, some alternative robust approach to DB backup would be needed.

CERN is also concerned about minimizing Oracle license costs. One approach they are taking is to use fewer, more powerful servers since the license cost scales with core count but not core power. The ATLAS PanDA production DB will shortly be moving from a 20 core server to a server with 8 slightly more powerful cores. The present server has 512GB memory. Large memory is important to avoid disk reads as much as possible. In the new 8 core server, memory will be increased to 768GB to compensate for the reduced core count. As mentioned below, this configuration provides the system with large headroom for load fluctuations and large spikes. The ATLAS PanDA Oracle database uses about 75TB to serve PanDA's 15 years of ATLAS data.

Could you give an idea of the physical resources (cores, storage...) needed to support PanDA at Rubin's scale? I suppose keeping in mind that the scale increases linearly over 10 years. A description of ATLAS's instance would be nice to have.
ATLAS PanDA resource usage:

- PanDA server: 9 servers x (8 cores, 16GB RAM, 500GB disk)
- JEDI: 9 servers x (8 cores, 16GB RAM, 500GB disk)
  - PanDA/JEDI scale roughly with jobs per day (concurrent job count together with job duration)
  - Configuration is by design far from a performance ceiling, designed to handle large spikes (cf. below)
- PanDA monitoring: 8 servers x (8 cores, 16GB RAM, 80GB) + 1TB shared space for logs storage. Monitoring scales with operating scale (same as PanDA/JEDI servers) together

with history (15 years of data, most of the monitoring usage looking at the last year of data)

- Harvester: 8 x (4 cores, 8GB RAM, 500GB disk). Harvester scales roughly with number of sites.
- Condor: 6 x (8 cores, 16GB RAM, 100GB disk). Condor scales roughly with number of job slots (which scales roughly with concurrent core count)
- iDDS: 2 x (8 cores, 16GB RAM, 100GB disk). iDDS is relatively new, its scalability in ATLAS is being exercised in the data carousel use case, in production. We'll learn how it scales in Rubin as we implement and exercise use cases.

We describe here the ATLAS PanDA resource usage. In estimating Rubin resource needs from this, these ATLAS PanDA metrics should be kept in mind

- Steady state operation is currently 400-450k concurrent cores on average
- ATLAS PanDA scales smoothly to 1.4M+ concurrent cores, exercised during HPC spikes
- Each job communicates a heartbeat to the PanDA server every 15min
- Average job duration is 8 hours and ranges from minutes to a few days. Longer (within limits) is better for efficiency, system load and limiting DB growth. Ideal duration is a couple of hours. ATLAS has a lot of short O(10min) jobs, but it's preferable to avoid them if possible.
- About 1M jobs are processed per day on average, peaking at about 2M
- O(1000) users active in the last year, including individual analysis
- O(100) global processing sites
- About 150 PanDA queues globally

Am I right that Rubin would not be using JEDI – that is ATLAS-specific and it's on us to provide the graph production etc to feed to PanDA?

While JEDI plays a role in the ATLAS Production System Prodsys2 that Rubin will not use, JEDI also provides functionality that Rubin will use. (JEDI basically operates as an integral part of the

PanDA core.) JEDI works with iDDS to provide task-based orchestration of jobs. A ForcedPhotCoaddTask in the processing pipeline may consist of thousands of jobs. Managing and monitoring them as a consolidated payload that is run with different input parameters makes use of JEDI. A hundred-job processing pipeline consists of only 14 tasks (DeblendCoaddSourcesSingleTask, MergeDetectionsTask, DetectCoaddSourcesTask, ...), which makes it significantly easier to debug problems at a particular processing step and control the execution progress. Graph generation is done using the BPS subsystem of the Rubin middleware. Once the graph is defined, the PanDA plugin in Rubin middleware sends the definition to the iDDS server, and at that point PanDA/JEDI takes care of its execution according to the relationship between jobs/tasks and inputs specified in the Butler repository.

What is the condor service doing in this application?
Condor has since PanDA's inception been its most important workhorse for submitting pilot jobs and harvesting processing resources. The condor service machines handle the submission of PanDA pilots to sites (condor queues) around the world. In recent years Harvester has been developed to abstract PanDA itself away from the many kinds of execution back ends that are now supported, of which condor is one. Harvester manages the condor pilot job submission activities carried out by the condor service machines. For ATLAS, with over 100 sites most of which use condor, the scale of the condor submission operation is large. For Rubin with far fewer sites it is a smaller task that will basically disappear inside the Harvester service serving Rubin sites.

For hardware, maybe outside of the actual Oracle servers, do you think it sensible to consider using Romes to cover the core/memory needs? We're looking at 128 core/512 GB nodes. Is there some traffic requirement that points to a certain number of individual nodes?
Many ATLAS PanDA components are using 8 core VMs simply because that is the configuration preferred by CERN IT. It is the largest core count configuration CERN supports. From our point of view it would be preferable to have VMs 2x or 4x the size and hence fewer of them. But 128 core/512 GB is a big step in consolidating servers that we have never tried. Horizontal scalability is efficiently supported by having several servers and adding additional as needed. Using many servers also gives – crucially – redundancy and resiliency. You don't want to lose services if a huge server falls over. Apart from these considerations, we see no reason not to try 128 core/512 GB servers. The servers don't have heavy network traffic or disk I/O requirements. In terms of power, we estimate that a 128 core/512 GB server is roughly commensurate with the power Rubin will need in the near/mid term. We assume some stan-

dard virtualization layer will be used with this server and the whole setup will be available as a number of VMs.

You don't need advanced compression? Basic is included for us.
Some tables are currently using advanced compression. The largest OLTP compressed tables however are archive tables, i.e. the rows are not updated and data is added by a job in bulk. We could probably get away with basic compression.

Are there globs in the db? I gather basic compression does not compress them.
There are LOBs in the DB, but they are not compressed.

They were curious what the transportable namespaces are used for (they thought it was tablespaces?)
It should be tablespaces, this is a typo.
In ATLAS there are 2 databases for PanDA: the "active" database and the "archival" database.

- The active database holds the last 3 years of data and is based on fast storage. The DBA creates yearly tablespaces.
- The archival database holds jobs older than 3 years. In practice the data in the archival database is never used and kept only for exceptional cases (e.g. investigation of a very old issue). Storage can be less performant, since the DB is used extremely rarely.

The tablespaces are copied from the "active" DB to the "archival" DB manually once per year. Alternative strategies for archival could be possible.

We have (and use Data Guard); do you use advanced data guard?
From an application perspective there is no dependency at all on Data Guard or on RAC. These are setups available at CERN to provide highly available services.
Active Data Guard is used for physical standby in case the primary would fail. PanDA only relies on the primary database.

Just checking: the Oracle server is the 20 core machine with 512 GB memory and a 75 TB database (for 15 yrs, so far). Soon to go to 8 core, 768 GB mem. Is this a single machine or a RAC cluster?
At CERN there is a RAC cluster with three fat nodes. EACH node has 20 core+512GB mem,

moving to 8core+768GB mem. Applications are clearly delimited across the nodes so that they don't overwrite the memory of each other or affect each other. Applications could failover to other nodes in case of node unavailability, but this never happens during normal operations. The assignment during normal operation is:

- One fat node for PanDA
- One fat node for Rucio (data management)
- One fat node for monitoring applications and various side services

Old CPU: 2 x Intel Xeon E5-2630 v4 (broadwell, total of 20 physical cores), 3.10 GHz max
New CPU: 2 x Intel Xeon Gold 5222 (cascade lake, total of 8 physical cores), 3.90 GHz max

PanDA developers also use a small clone of the production DB, called INTR (integration) DB, for development. I think it would be TBD by the developers whether there should be an INTR at SLAC or Rubin-focused developers use also the INTR at CERN.

A References

[LDM-636], Kowalik, M., Gower, M., Kooper, R., 2019, *Batch Production Service Requirements*, LDM-636, URL `https://ls.st/LDM-636`

[RTN-001], O'Mullane, W., 2020, *Data Preview 0: Definition and planning.*, RTN-001, URL `http://RTN-001.lsst.io`

B Acronyms

Acronym	Description
ATLAS	A Toroidal LHC Apparatus
BNL	Brookhaven National Laboratory

BPS	Batch Production Service
CERN	European Organization for Nuclear Research
DB	DataBase
DC2	Data Challenge 2 (DESC)
DESC	Dark Energy Science Collaboration
DM	Data Management
DOE	Department of Energy
DP0	Data Preview 0
FY21	Financial Year 21
GB	Gigabyte
HPC	High Performance Computing
HSC	Hyper Suprime-Cam
IDF	Interim Data Facility
IN2P3	Institut National de Physique Nucléaire et de Physique des Particules
IT	Information Technology
JEDI	Job Execution and Definition Interface
L3	Lens 3
LDM	LSST Data Management (Document Handle)
LHC	Large Hadron Collider (at CERN)
NCSA	National Center for Supercomputing Applications
PanDA	Production ANd Distributed Analysis system
RAM	Random Access Memory
RDBMS	Relational Database Management System
RTN	Rubin Technical Note
SLAC	SLAC National Accelerator Laboratory
UK	United Kingdom
US	United States