OSG Workload Management --
Generalizing & Extending Panda

Torre Wenaus, BNL

OSG workload management planning meeting
Fermilab
Sep 1, 2006
Panda

- **PanDA** - Production and Distributed Analysis system
- Started August 2005 in US ATLAS as a full redesign to achieve performance, scalability, ease of operation needed for ATLAS datataking (up to 100-200k jobs/day)
  - Leverages past production experience
  - Designed to inherently support analysis
  - ‘One stop shopping’ for distributed processing
- In production since Dec 2005
  - Ambitious development milestones met
  - Thanks to productive development team
  - Still in rapid development, esp. analysis
Core components

- **Job Interface** – allows injection of jobs into the system
- **Executor Interface** – translation layer for ATLAS prodsys/prodDB
- **Task Buffer** – keeps track of all active jobs (job state is kept in MySQL)
- **Brokerage** – initiates subscriptions for a block of input files required by jobs (preferentially choose sites where data is already available)
- **Dispatcher** – sends actual job payload to a site, on demand, if all conditions (input data, space and other requirements are met)
- **Data Service** – interface to DQ2 Data Management system
- **Job Scheduler** – send pilot jobs to remote sites
- **Pilot Jobs** – lightweight execution environment to prepare CE, request actual payload, execute payload, and clean up
- **Logging and Monitoring systems** – http and web based
- **All communications through REST style HTTPS services** (via mod_python and Apache servers)
Key Panda Features, and Status

• Designed from beginning to support both managed production and individual users (analysis) via **flexible job spec/injection**
  • interactive analysis, user-level job submission, regional group production
    • first two implemented; regional group production not yet
  • grid-based or farm-based resources
    • currently supported: grid: CondorG, batch farm: PBS
• **Dataset** based organization of Panda matches the DDM system and the **analysis work model** (implemented, based on DQ2)
• Use of DDM to **pre-stage input data** and **immediately return outputs**, all asynchronously, minimizes data transport latencies and delivers earliest possible first results (implemented)
• Management/optimization of workload via **job queue** with **late binding** of jobs to worker nodes gives **dynamic and flexible** system response to highly variable DA work (implemented)
• Use of grid and/or farm batch queues to **pre-stage job wrappers** to worker nodes (pilot jobs) and **directly deliver workloads** from Panda allows **fast injection of DA work** (implemented but a work in progress!)
Key Panda Features (2)

• Support for packaging, deploying, running **arbitrary user code/jobs**
  • Implemented - arbitrary scripts can be specified by job and loaded (via http retrieval) for execution

• **Comprehensible system view** offered to users: **heterogeneous** distributed resources appear as one **uniform** resource accessed through standard interface (implemented)

• Easy to **integrate your own local resources**: site requirements are pilot delivery (via local batch queue or grid), outbound http, and access to a DQ2-enabled SE (locally or ‘nearby’) (only Tier 2s so far)

• Easy-to-use **client interface** makes integration with diverse analysis/interactive front ends easy (implemented)

• **User ID** built into Panda DB; **monitoring and metadata** extensible to **user level** (User ID (DN) is recorded, user-level extensions not in yet)

• **User-level controls, quotas** directly implementable in Panda’s brokerage rules (not implemented yet)

• Extensive **monitoring & browsing** (some specialization for DA, more to come)
Panda Server

- Apache-based
- Communication via HTTP/HTTPS
- Multi-process
- Global info in the memory resident database

**System configuration**

- **Components**
  - Apache 2.0.55
    - mod_python doesn't run with Apache 2.2.X yet
  - Python 2.4.2
  - mod_python 3.1.8
  - mod_gridsite 1.1.15
  - MySQL-python 1.1.8
- The panda server doesn't have dependence on special grid-middleware
  - can run even at CERN if needed
  - needs to be deployed near the database

**Diagram**

- Apache
  - child process
    - Python interpreter
  - MySQL API
- DB
- Job info, etc
- DQ2
- HTTP/HTTPS
- HTTP/HTTPS

T. Maeno
Job Flow

Production system

ProdDB

Job

Panda Server

Submitter

User

Pilots

1. Each pilot runs on a worker node
2. 1. send a request
3. 2. receives a job
4. 3. runs the job

T. Maeno
Client Interface

- Pickle module of Python and native curl
  - External component is not needed
  - Pickle is faster than XML parser
- Clients require python 2.3 or higher, curl and a grid-proxy, but not grid middleware
- Simple, light-weight, independent from grid-flavor
DA Submission to Panda: pathena

• Job submission to Panda is very simple and easy via python client interface
  • Job and file specs serialized by pickle and sent to Panda via http POST
  • Only client dependency is on curl (http library)
• pathena (Tadashi Maeno) uses this interface to submit Athena jobs to Panda (see PhysicsAnalysis/AnalysisCommon/UserAnalysis/share)
  • an adaptation of the ‘athena’ command line invocation familiar to Athena users
• pathena processes user submission in two steps:
  • Build step (buildJob): gather up user code, store it (webdav, svn) and ship it to processing site for (http) retrieval by the pilot and site installation/build in user area
  • Run step (runAthena): run N Athena jobs with user-designated input and output datasets
• User retrieves output dataset via DDM tool dq2_get
**buildJob**

- Compiles sources/jobOs and install libraries/jobOs

  - Archive of 'work dir'
  - could contain many packages
  - keep directory-structure

Usage:

when running athena, e.g.,

```
$ athena -c "OutputLevel=DEBUG; DetDescrVersion = 'Rome-Initial' opt1.py myTop.py opt2.py
```

All you need to do is

```
$ pathena -c "OutputLevel=DEBUG; DetDescrVersion = 'Rome-Initial' opt1.py myTop.py opt2.py --inDS dataset1 --outDS dataet2 --split 10
```

- Archive of InstallArea
- Symlinks are changed to be relative → 'portable'

T. Maeno
buildJob (2)

HTTP-reachable location

sources.tgz

wget

DDM may be used but a bit slow. Panda server provides putFile method to upload files.

buildJob

1. wget https://.../source.tgz
2. tar xvfz source.tgz
3. cmt broadcast make

T. Maeno
runAthena

- run Athena with arbitrary configuration
  - arbitrary job O
  - arbitrary inputs and outputs
  - arbitrary package configuration

libraries.tgz

inputs

runAthena

outputs

Pool, Coll

Pool, Ntuple, Hbook

T. Maeno
Can include a third merge step

Partial results delivery

T. Maeno

Full Scenario

Local

Panda

buildJob x 1

runAthena x N

source.tgz

buildJob

libraries.tgz

Storage

outputs

runAthena

runAthena

outputs

outputs

outputs

inputs

inputs

output dataset

input dataset

runAthena

runAthena

merge

outputs

outputs

outputs

Not yet
Multitasking Pilot

- Can offer a large pool of analysis resources (the production farm) without machines standing idle
  - Good for accommodating highly fluctuating analysis loads
- Analysis can either run concurrently with production job (current approach) or production job can be suspended (still to try)
- Concerns:
  - Resource contention, esp. memory, when running concurrently
  - Are we ‘cheating’? Not if we own the resource
Workflow of Panda jobs scheduler and pilot job
User Access to Data

- **DQ2 end-user tools**
  - Documentation
    https://uimon.cern.ch/twiki/bin/view/Atlas/AccessPandaData

- **dq2_get**
  - copy files over the grid
  - Files need to be registered in DQ2 catalog
    - OSG/Panda: done
    - LCG: still using LFC
    - NG: on going
  - Working with OSG, LCG (temporary hack).
    Once NG prod system is integrated with DQ2, NG datasets are available automatically

- **Todo**
  - 3rd party transfer
  - Parallel transfer
  - md5 sum check

---

**Work-flow**

1. Use dq2_ls to look for interesting datasets
2. Submit jobs using pathena
3. Jobs runs on LCG/OSG/NG production system
4. Output datasets are registered to DQ2 catalog automatically
5. Get output by using dq2_get

- **dq2_ls**
  - List datasets and files in DQ2 catalog
  - Allow users to search for datasets
  - Wildcard search
    > dq2_ls mc11.*RDO*

- **dq2_put**
  - not yet
  - Register files in DQ2 catalog
  - Files produced on production system are registered to DQ2 catalog automatically
    - regional production

Tadashi Maeno, Dietrich Liko et al
DQ2 Architecture

Dataset repository

Dataset hierarchy (to be completed)

Dataset location catalog (Site SEs)

Dataset content catalog (LFNs)

Dataset catalog services

Registration, lookup

Subscription services

Registration, lookup

Claim loading

Subscription management

Production bookkeeping

End user interfaces

Client applications

Space manager

Claims catalog (File usage, lifetime)

Dataset subscription queue

Local replica catalog (LFN -> PFN)

Claims info

Not part of DDM
Dataset-based Data Flow in Panda

Production dataset

Dispatch datasets

Destination dataset

BNL

Remote site

T. Maeno

T. Maeno
DQ2 Based Data Handling in Panda

Objective: extensive automation for minimal operations manpower and maximal flexibility/quickness in reacting to transfer/storage service problems

Operational: BNL, BU, UTA, UC, OU, IU, SLAC
Dataset Browser

DQ2 dataset browser, category csc, datasets at BNL

Click for help
Dataset lists last updated 03-07 02:18 (updates are currently nightly)

Select a dataset category Category counts are totals, exclusive of selections or site restrictions

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>16079</td>
<td>All datasets</td>
</tr>
<tr>
<td>csc</td>
<td>333</td>
<td>Computing system commissioning production</td>
</tr>
<tr>
<td>cto</td>
<td>8</td>
<td>Combined testbeam production</td>
</tr>
<tr>
<td>dc2</td>
<td>6</td>
<td>Data Challenge 2 production</td>
</tr>
<tr>
<td>destination</td>
<td>9271</td>
<td>Panda destination sub-blocks</td>
</tr>
<tr>
<td>dial</td>
<td>14</td>
<td>DIAL analysis</td>
</tr>
<tr>
<td>dispatch</td>
<td>4307</td>
<td>Panda dispatch blocks</td>
</tr>
<tr>
<td>larg</td>
<td>5</td>
<td>LAr commissioning</td>
</tr>
<tr>
<td>mc</td>
<td>416</td>
<td>MC production</td>
</tr>
<tr>
<td>other</td>
<td>168</td>
<td>Everything else</td>
</tr>
<tr>
<td>pone</td>
<td>77</td>
<td>Home physics workshop production</td>
</tr>
<tr>
<td>testpanda</td>
<td>553</td>
<td>Panda test datasets</td>
</tr>
<tr>
<td>tile</td>
<td>9</td>
<td>Ticaal commissioning</td>
</tr>
</tbody>
</table>

Restricted to BNL resident datasets (clear) Choose another site: ASCC BNL BNL-SC BU CERN ONAF FZK INP3D OU PIC TRIUMF UC UTA

Selected category: csc (clear category)

csc datasets matching selection and present at BNL (122):
csc11.005001.pythia.mmbias.recon.AOD.v11004103
csc11.005001.pythia.mmbias.recon.CBNV.v11004103
csc11.005001.pythia.mmbias.recon.ESD.v11004103
Panda Security

- System is https based, authenticated by grid certificate
- Any user of the system (job injection from client, pilot requesting work) must hold a valid proxy
- DN of user is carried as part of metadata of the job
- Execution on pilot, and file management, currently rely on production-wide certificate
- Expect to (need to) introduce gLexec based identity change on WN to submitter identity
- Planning to introduce means of securing the workload specification (transformation) from tampering in the Panda DB (encryption using RSA key pair)
Panda status and near term plans

• Moved to local pilot submission at BNL, in particular for analysis pilots, much more robust than Condor-G (blame the G)
• DDM enhancements to enable opportunistic sites
• Multitasking pilot being tested/deployed
• User accounting/quotas operational
• Error handling, robustness improvements (never ending)
• Near term goals:
  • Add more sites, keep working on lowering the barrier for new sites
    • Including LCG sites -- in progress
  • Support for ‘standalone’ production -- managed fully by Panda, not through ATLAS prodsys
    • Analysis usage is already ‘standalone’
  • More usage flexibility, support more analysis use cases
• NB all these near term goals are all in the direction of generalizing Panda (thanks to its success), albeit still within ATLAS scope
Panda Team

- Project leaders: K. De (UTA), T. Wenaus (BNL)
- Lead developer: T. Maeno (BNL)
- Panda team
  - BNL: P. Chiao (UTA student), W. Deng, A. Klimentov, S. Reddy (UTA), Y. Smirnov, P. Thilagar (UTA student), T. Wlodek, Xin Zhao
  - UTA: N. Ozturk, M. Sosebee, P. McGuigan
  - Oklahoma U: K. Arunachalam, Horst Severini
  - U Chicago: M. Mambelli
  - CERN: Paul Neilssen (UTA) + (we hope) soon 50% of a DQ2/DDM expert