Panda/DDM Integration

Torre Wenaus (BNL), Tadashi Maeno (BNL)

BNL DDM Workshop

BNL

September 28, 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
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)
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)
Datasets in Panda

- From a point of view of DDM/PANDA there is no essential difference between production jobs and analysis jobs.
- User dataset can be accessed via DDM (e.g., using DQ2 end-user tools).

T. Maeno
Datasets in Panda (2)

$ pathena -c "OutputLevel=DEBUG; DetDescrVersion = 'Rome-Initial' opt1.py myTop.py opt2.py --inDS dataset1 --outDS dataset2

Pre-existing

Created by Panda

User owned dataset

‘tid’ dataset Subtask/grid specific

Torre Wenaus, BNL

T. Maeno
Dataset-based Data Flow in Panda

Production dataset ----> Dispatch datasets ----> Destination dataset

BNL

Remote site

input

output

Sub-destination dataset

T. Maeno
**DQ2 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

Possibility, not practice today
Present Panda DDM Operating Scale

- 7 facilities: BNL, BU, UTA, UC, OU, IU, SLAC
- ~15 DQ2 site installations (site services + LRC)
- ~900 active subscriptions, recent 12 hour period
- ~8k Panda data management datasets (25k total)
- 1.3M LFNs in BNL LRC (POOL FC MySQL)

D. Liko, Sep software week:

<table>
<thead>
<tr>
<th>CSC11 AOD Data at Tier-1</th>
<th>CSC11 ESD Data at Tier-1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Datasets</strong></td>
<td><strong>Complete</strong></td>
</tr>
<tr>
<td>ASGC</td>
<td>96</td>
</tr>
<tr>
<td>BNL</td>
<td>226</td>
</tr>
<tr>
<td>CERN</td>
<td>253</td>
</tr>
<tr>
<td>CNAF</td>
<td>1</td>
</tr>
<tr>
<td>FZK</td>
<td>16</td>
</tr>
<tr>
<td>LYON</td>
<td>72</td>
</tr>
<tr>
<td>RAL</td>
<td>10</td>
</tr>
<tr>
<td>SARA</td>
<td>2</td>
</tr>
<tr>
<td>PIC</td>
<td>7</td>
</tr>
<tr>
<td>TRIUMF</td>
<td>7</td>
</tr>
</tbody>
</table>
DQ2 Callbacks

- Implemented in DQ2 at US request for Panda
  - Now extended and heavily used eg. for monitoring
- Useful in implementing data-driven workflow
  - Completion of input dataset dispatch to trigger job release
  - Completion of output dataset archiving to
    - signal data availability
      - data not available (=usable) until replicated to Tier 1 (empirical fact)
    - trigger release of downstream job
  - Automation of reprocessing
    - If the mass store system can tell us when a file has been staged in, daemon can deliver callback to trigger associated job release
- Depends on performant DQ2 subscription system
Pilot-Level Data Handling

• Pilot data handling responsibilities:
  • Input data
    • Get input data from local SE (where it’s been placed by the dispatch block subscription) to the WN
  • Output data
    • Transfer outputs to local SE
    • Validate the transfer
    • Register the local SE instances with local LRC
    • Inform Panda server of outputs
      • XML FC transmitted to Panda dispatcher, which registers them with output (destination) dataset
      • Destination dataset subscription to BNL then takes care of archival replication to BNL
  • Also pilot provides info to Panda (via local DDM http service)
    • Space remaining on local SE
Pilot-Level Data Handling Implementation

- `panda/pilot2/DQ2ProdClient2.py`
  - `get_data`, `put_data` functions
  - Site-dependent SiteMover implementations depending on nature and organization of the SE

- Sites currently in Panda production all have site-local SEs

- High current priority: add sites without site-local SEs (‘opportunistic sites’)
  - Implemented, based on use of `uberftp` for grid transfer with remote SE
    - `uberftp` because it supports remote md5sum check, subdirectory placement
    - Not yet in production

- New effort in generic pilot/scheduler ‘TestPilot’ beginning
  - Intended for non-US ATLAS, and non-ATLAS (OSG) Panda usage -- highly generic and customizable, and low install/deployment threshold
  - Requires DDM plug-ins by which a site, region, application, or VO can insert its own data handling
    - Currently working on LCG ATLAS version based on DQ2
    - To come: generic OSG version (have an OSG customer, CHARMM)
    - Will draw on `DQ2ProdClient2.py` for US ATLAS site support; explore merging
DQ2 Architecture

- Dataset repository
  - DQ2 dataset information
  - Dataset location catalog (Site SEs)
- Dataset content catalog (LFNs)
  - Dataset hierarchy (to be completed)
- Dataset catalog services
  - Registration, lookup
- Subscription services
  - Registration, lookup
- End user interfaces
  - Data access
  - Subscription services
    - Subscription management
  - Dataset management
  - Dataset lookup
- Client applications
- Production bookkeeping
  - Not part of DDM
- Claim loading
- Claim info
- Global
- Local
- Space manager
- Claims catalog (File usage, lifetime)
- Dataset subscription queue
- Local replica catalog (LFN -> PFN)
Replica Catalog

- MySQL implementation of POOL FC
- Simple implementation of basic functionality, vintage 2002
- Generally stable, scalable, efficient SQL access by bypassing POOL interface (which we do a lot)
- http web service front end provides lightweight client access with no client dependencies beyond http
- Weakness: Authentication. Currently based on standard MySQL user/pass
  - Possible to google user/pass in ToA; even write accounts until recently
  - Solution: grid certificate based authentication, implemented by Sasha Vaniachine et al, Wensheng et al working on production deployment
- Very amenable to wider ATLAS use?! Would (likely) require centralized LRCs at CERN serving Tier 1 clouds (we know how to run MySQL services @CERN)
  - Implement first as LFC backup, used as LFC mirror?
Grid Middleware Fallbacks

An important requirement in Panda and its DDM is support for *fallbacks* for external components, in particular immature unproven middleware.

- FTS is in my view still immature unproven middleware.
- Panda/DDM uses FTS for all data transfers at the moment (BNL<->Tier 2s) but this will not necessarily always be the case.
- Panda/DQ2 has the capability now to swap out FTS for an alternative, and this has been exercised during periods of FTS problems.
- If FTS (or any other middleware) fails and as a result DDM fails, it’s *our* fault, if there was a potential fallback that we couldn’t use.
- DQ2 in the new version still supports fallbacks, and we (OSG) want to keep it that way.

Slide untouched from April ATLAS grid workshop at CERN
(Today I’d mention LFC! And note Miguel comment: Don’t use SRM if you don’t have to!)
Data Access Documentation

CscGetFiles
Guidance for locating and accessing CSC data on the different grids

UsingDQ2
Documentation for setting up and using the dq2_* end user data handling tools
Dataset Browser
Tasks define production tasks and record their associated metadata.

Datasets define and organize the task inputs and outputs.
DQ2 Datasets In Action (Panda)

DQ2 catalogs datasets...

...provides user-level access: dq2_get, dq2_ls, (dq2_put coming)...

...provides automated management/movement tools for use in production...

...organizes validation...

...and ultimately lets you get at the files!
DDM Wish List

• Dataset catalog system OK but still some holes to fill
  • More bulk ops -- eg. efficient dataset metadata retrieval
  • Versioning system, closed/frozen datasets now in place but, from usage, not sufficient
    • Very few datasets frozen; need more awareness in system of partial datasets, %complete, empty datasets

• Subscription system much more problematic
  • Extremely long subscription servicing cycle
  • All the issues we’re discussing here of replication robustness and performance

• Local replica catalog
  • US MySQL LRC generally working well, LFC is a problem (for US as well)
    • Need production deployment of grid certificate based authentication for MySQL LRC (expected very soon?)
  • We need scalable robust LRCs ATLAS-wide
DDM Wish List (2)

• Clean, simple, documented packaging/installation for OSG
  • As much coherence as possible between LCG and OSG
  • cf Miguel’s comments, we should use distutils directly? Cleanly split off LRC installation?
• Eventually: system partitioning. US DQ2 system publishing to ATLAS
• DQ2 monitoring much better; next is monitoring of the monitoring
  • ‘Intelligent layer’ watching for prevalent errors, raising alarms
• ‘Claims system’ that became a couple more LRC attributes
  • Need a real plan for dataset- as well as file- level space management
• More end-user support
  • DDM down to the laptop
  • User subscriptions, upgraded dq2_get (copy to local SE and register)
  • Opening up FTS site-to-site access -- Dan’s proposal
• More developers! Particularly tool-making: consistency checking, site data management, etc. Have had very strong contributions from site people, more welcome!
DDM Wish List (3-10)

• Plus all the things I’ve forgotten ;-)
Current/Future Activities

- Top priority: extending Panda to opportunistic US ATLAS sites, which is mainly a DDM issue
  - And extending Panda yet further -- LCG ATLAS in particular
- Top priority: robustness, stability, fault tolerance
  - DDM failures both in production and analysis way too common. Partly a facility issue, partly fault tolerance (retry, timeouts etc.) in the DDM system
- User-level support
  - Today neither dq2_get nor asking users to use subscriptions is adequate
- Many efforts/improvements coupled closely to ATLAS level program and decisions
  - LRC, grid authentication, http interface
  - Subscription implementation, monitoring, quality of service
  - DDM monitoring
  - System partitioning (better to do it before we urgently need it); partitioning by region (OSG production) and function (user catalogs)
- Site services
  - Space management, consistency between storage and LRC, uniform SRM/xxxx deployment, SRM 2.2, stable ToA endpoints
From Aug Tier 2 Workshop
http://www.usatlas.bnl.gov/twiki/bin/view/Admins/TierTwoStorageDataServices

- Tier2 DDM responsibilities, tasks
  - Connect to Tier 1 via FTS
  - Connect to all other US Tier 2s via FTS
  - Provide SRM based storage. Current baseline: SRM/dCache
  - Investigate xrootd for possible role (esp. SLAC)
  - Provide DQ2 site service
    - Site should have single, stable endpoint; ToA config should be stable
    - Site (re)configuration should be internal, not exposed at the site service address
  - Provide US standard LRC and associated http service
    - Separate MySQL from site service strongly recommended
  - Support agreed, US supported space usage controls
  - Support agreed, US standard & supported end user data access tools
    - Today, requirement includes LCG UI if direct LCG data access is desired (available from BNL AFS)
    - But data aggregation at BNL should make this mostly unnecessary