Possible Beyond Pledge Resource Scenarios

Torre Wenaus, BNL
ADC Facilities Jamboree
Dec 3, 2015

This presentation/discussion

- Beyond pledge resources and Tier 3s were big topics of discussion at a US ATLAS S&C meeting in August (they always are)
- We agreed there (Simone kindly attended) that there are general issues that could usefully be discussed at the Jamboree
- Because unpledged resources are not under ADC control, ADC doesn't often discuss them
- But even if sites do whatever they want with unpledged resources (as is their right), it would be good for ADC to discuss and agree on recommended scenarios that people are free to adopt (or not)
- ADC would aim to provide a few tools for recommended scenarios
  - With a focus on analysis; usage of unpledged for production is well established and supported in our standard systems

Precepts

- Supporting beyond-pledge usage in some 'recommended way' should require minimal incremental effort over and above what a site is already doing
  - For an on-grid site, beyond-pledge should look much like on-grid
  - For an off-grid site using batch and local disk, beyond-pledge should look much like that
- The two extremes for a site's ADC-awareness: support All, support Zero
  - A: If your resources are well integrated with ATLAS distributed processing (CE, SE, PanDA, DDM), you want beyond-pledge to be supported within ADC services
  - Z: If your resources are integrated not at all or minimally – local batch and local disk – you want lightweight low maintenance tools to enable local analysis with the necessary connections to the wider ATLAS analysis world
  - Some services like cvmfs are essential and common to all scenarios
- And sites may be somewhere in between (Lightweight integration)

A sites: Well integrated

- Many sites all over ATLAS
- Storage and data access:
  - LOCALGROUPDISK provides for locally managed, DDM-integrated storage
  - Import what you need to LOCALGROUPDISK with Rucio and run analysis on it
- Processing:
  - If PanDA is supported, beyond-pledge can be supported either via...
    - A dedicated PanDA site with managed throughput controls and access controls
    - A site shared with general usage, with the over-pledge fraction of the throughput preferentially allocated to locals – PanDA's 'localpool' mechanism
      - Room for improvement on the algorithm – greater preferential weight on local jobs
  - Or sites may use local batch served by DDM

Z sites: Minimally integrated

- Many sites in the US
- Storage and data access:
  - Not grid storage, no SE endpoint
  - Want lightweight, shrink-wrapped tools to reach outside for data
    - dq2-get, gridftp, FAX, xrootd, xrootd cache, Rucio cache, ...
    - … a complicated zoology covering a range of use cases and an evolving set of technologies (AleDG has summarized the state of the protocols, we'll be seeing and discussing it this week)
- Processing:
  - If PanDA is supported, good for you, you understand that PanDA isn't just a grid processing system ;-)
  - If PanDA isn't supported, you're like most of the US Tier 3s today, using local batch, probably happily, and probably for some understandable reasons
    - Will get to that later

L sites: Lightweight integration

- Storage and data access:
  - Can we focus the Z zoology down to a lightweight, uniform way to provide a small site with good data access?
- Processing:
  - Can we make PanDA lightweight and fast enough on local resources that small sites adopt it because their analysis users want it?
  - The pilot and PanDA embody a great deal of knowledge and capability for running analysis jobs and managing their data – there's much benefit to be had
  - Resolve the disadvantages to gain the advantages
- If we can offer ~painless ways to move sites from Z to L we would all be the better for it
  - More capability available to analyzers and more uniformity in tools and operations
- Tried in the US over years – “T3 in a box”, “Virtual T3”, etc – but never got there

Do emerging tools fit in ADC?

- Tools supporting beyond-pledge/Tier 3 should be consistent and coherent with overall ADC strategy, if they are to receive endorsement or support from ADC
- Do tools now on the table meet that test? Do they integrate coherently with ADC services overall?
- Two examples that are questionable:

Rucio Cache

- “Goal is to provide a lightweight way for small sites to get data from their PanDA jobs” – Doug Benjamin
- Basically an API by which an external file manager can use the Rucio file catalog to manage a file cache as a 'volatile Rucio storage element'
  - An external file manager client (like xrootd/FAX) is responsible for declaring to Rucio file insertions, deletions
- But has no notion of datasets, only files
  - How can it integrate coherently with ATLAS DDM without dataset support?
  - Datasets may be a secondary notion for Rucio but they are a primary tool for ATLAS
- Bears some resemblance to a Rucio feature requested long ago by PanDA, except what PanDA asked for was a dataset-level cache registry service (which doesn't exist)
- Does this fit? Can we make it fit? Should we?

ATLAS Connect

- Users authenticate to their institutional computing and use a job submission tool that transparently ssh's to a machine capable of submission to distributed resources (via Condor and 'flocking' to connect to remote pools)
- Does it extend coherent ATLAS computing the 'last mile' to the user's institutional computing (a good thing)?
  - By serving as a mechanism for “off-grid” environments without grid authentication to use local authentication to access the ATLAS distributed computing environment and inject PanDA jobs
- Or does it do the opposite, propagating the unmanaged, unaccounted, unmonitored environment of local batch into the ATLAS distributed computing environment (a bad thing)?
  - By 'overflowing' Tier 3 batch jobs into T2/T1 resources, still as mere batch jobs
- It clearly does the latter; it less clearly promises to do the former

An ADC lightweight toolkit

- If tools of questionable coherence with ADC are emerging from beyond-pledge communities, how can we better focus on coherence?
- How can we make sure that a new ATLAS Supertool actually is a considered product of ADC planning?
- What tools would best serve a lightweight environment and be coherent?
- First let's take advantage of an exercise now underway in the US to look at key requirements on lightweight tools
  - Currently a **draft** document gathering comments, I am paraphrasing it
  - I take full responsibility for the paraphrase:

Requirements on lightweight tools: Processing

Abbreviated/paraphrased from “Requirements for accessing external resources from Tier 3” ***draft*** US doc

- Support familiar analysis workflows with minimal change
- Monitoring allowing users to understand the state, progress and problems in their submissions
- Fault tolerant, support for auto retry at functional sites, robust bookkeeping of input file processing
- Robust against detectable user error: validating inputs, configuration, site capability, run ability, etc.
- Support fair share
- Remote resources should be continuously validated

Requirements on lightweight tools: Data

Abbreviated/paraphrased from “Requirements for accessing external resources from Tier 3” ***draft*** US doc

- Outputs should be accessible without explicit local copy
- Easy file movement between local/remote storage
- Support for team storage, access rights, fair share
- Make data originating at small sites available to remote jobs

Meeting the Requirements

- How can we meet the requirements with tools that are coherent with ADC?
- For processing, PanDA clearly meets the requirements on their face already, except perhaps for an unstated one, that people have to like it better than the (local batch job) alternative in the Z site context
- For storage and data access, how can we make the zoo as simple and transparent as possible for users? How much can we hide behind shiny smooth stable interfaces? Not a new strategy but an incomplete one
  - Easy common interface that abstracts away the details underneath
    - dq2-get + rucio-get + fax-get + … = ddm-get
  - CDN-like direct data access service as discussed in the last talk
    - When you hit play on a Netflix movie, you don't care what caching and data federation mechanisms are at play behind the scenes

PanDA for lightweight sites

- Is today's PanDA well suited to a small lightweight analysis site? Not in some important respects – speed and simplicity
- PanDA is tarred with being the gateway to overloaded grid resources with slow turnaround, so “PanDA is slow”
  - Partly unfair, but only partly
- We need an operational mode tailored to a user's wholly owned resource
  - The 200 batch jobs they're able to launch instantly on their own T3 (or in their own cloud allocation, or other opportunistic resource...)
- And these local beyond-pledge T3 resources do look like opportunistic resources...

Tier 3 as an Opportunistic Resource

- A Tier 3 shares characteristics with the opportunistic resources we've just been discussing as the target of the Event Service
  - We made the ES to harvest dynamically/transiently available resources in a quick agile way, which is what users want to get out of exploiting their T3
- When slots become free, they're available to be grabbed for immediate exploitation
  - Delay means an unhappy analyzer and slower analysis turnaround
  - A user acquiring T3 batch slots is similar to acquiring an HPC scheduling hole or a spot market cluster
- Transparent data access, local if possible but remote if necessary, is valuable to Tier 3s
  - Payloads should function as efficiently as possible if data is remote (while still preferring local)

Improving PanDA for lightweight sites

- Look at the 'just-in-time' pilot factory/job dispatch mechanism
  - It was designed to *remove* latencies and failure modes in grid based production, and does so effectively on the grid
  - But compared to local batch at a small site, it *adds* latency and failure modes
    - With batch, the user can inject work instantly and see it start running instantly: having an 'empty' resource available for instant exploitation is not uncommon in local Tier 3s
    - With PanDA today, a pilot factory has to provision the pilots, which typically takes minutes
      - Fixable? Sure, but not trivially across dozens of sites, and not a well-motivated approach in this environment anyway
- Why deal with supporting (or being dependent upon) a pilot factory and pilot provisioning latencies when you can just submit batch jobs?
- The locals 'own' these resources – it is understandable they expect complete, instantaneous, unmediated control

PanDA/JEDI analysis tasks + Event Service + Tier 3 = a better way

- We now support PanDA/JEDI analysis **tasks;** jobs are an implementation detail of getting tasks done
- A user now declares to PanDA the task they want done: process these inputs with this transformation
- PanDA applies available capable resources to getting the task done
- Can we usefully extend this by including T3s in the resources PanDA can apply to completing the task?
  - If the user has T3 cycles at their command, apply them
- Can we do so without importing the latencies and complications of grid-based PanDA to T3s?

From pathena to prun to... psub

- On the grid, 'just-in-time' has been PanDA's catch-phrase: holding work in reserve in PanDA's queue until it can be dispatched to a worker assured of being able to execute the work
  - Avoid grid latencies that can make old decisions on brokerage etc. obsolete when the work is actually dispatched
- On Tier 3s, the fact that the actual work is held in reserve in a just-in-time scenario becomes a disadvantage; it only delays applying available resources to the work
  - A resource is available *now*, we want to leverage it *now*
  - Delay in applying that resource to real work is undesirable, and serious if waiting humans are involved
  - Pilot factories introduce delay, add complexity, complicate ownership/accountability
- So instead, in environments where they are able to, have users submit batch jobs themselves, to acquire resources in the way they know and trust, with zero latency and complication from higher level infrastructure
- Provide a user CLI **psub** that submits batch jobs to serve as workers for the user's pre-declared tasks

psub (2)

- The self-submitted psub batch jobs are essentially the all-familiar PanDA pilot, in ES mode
  - But no pilot factory intermediary, no brokerage and dispatch intermediaries
  - No ownership/authorization/traceability issues – the user owns the job and the outputs it produces
  - While preserving all the knowledge and capability the pilot has, together with PanDA, in validation, execution, monitoring, data management etc
- The psub jobs declare themselves to PanDA/JEDI as being workers for a particular task or user, and pick up work from the pre-defined (via prun, pathena, or even prodsys) tasks of the user
- **You** the user own the T3/BP resource, **you** directly submit batch jobs that you own, they immediately launch and ask PanDA for work, which PanDA immediately delivers, **no questions asked**
  - Fair share on T3/beyond pledge resources is their business – PanDA makes mechanisms available but they're optional
  - All the controls might be at the batch system level, or the corridor conversation level

psub (3)

- The user can submit however many jobs they can cram into their T3; they all dynamically become active event consumers for the task
- PanDA manages them as event service workers, spreading the work dynamically over available workers, which can appear and disappear
- Inputs are accessed via EDS, delivering from local copy if available, else transparently remotely, leveraging data marshalling by EDS and fully asynchronous pre-fetch
- Outputs streamed in near real time as ES output event clusters, available for early inspection at your neighborhood object store, automatically merged into conventional outputs when PanDA detects task completion
- Ditto for incremental logs – available for early inspection via http

psub benefits

- A user task can be shared across beyond-pledge T3 resources *and grid resources, other resources*
  - No 'ATLAS Connect' necessary to harness remote grid resources to expedite a user's T3 task
  - Users can (optionally) apply their PanDA grid quotas to their tasks concurrently running on T3/BP
- Could also use your own laptop, running a pilot daemon
- And/or dedicated ATLAS@Home on all the desktops along your corridor
- PanDA is able to respond with its full capability – ie *instantly* – to the user's declaration that resources are available *now* to execute their work
  - No middleman latencies
- What remains 'just-in-time' in this scenario is the workload brokerage at the fine grained level of event ranges – PanDA can assign what work goes where on the basis of the data locality landscape in the present moment
  - Maximize opportunities to dispatch where data is local, minimize use of remote access
- Local resources are truly owned and fully controlled locally; resource allotment is exactly as for simply-batch submission

psub benefits (2)

- No impedance mismatch between grid and off-grid submissions, and without importing gridworld disadvantages
  - The user submits the batch jobs they love and trust, while PanDA provides its well developed mechanisms for handling bookkeeping, data access and delivery, work allocation, insulation from heterogeneity, monitoring, log access, etc.
- A site could choose to permit **both** standard pilots and user psub
  - Users don't have to use psub to see their tasks attended to (psub requires attentiveness, just as local batch does)
  - But if they care to, they can, to accelerate throughput
    - Regulated by the site's (not PanDA's) fair share mechanisms

Why does ES mate well with the psub approach?

- The resources a user acquires via psub become their execution pool, that can take advantage of PanDA's ES capability to apply efficiently towards the user's tasks
- Batch slots of any duration can be used fully; work assignment and lifetime is not coupled to a pre-defined job
- psub ES jobs become dynamic agents contributing to the higher task of completing a task, in concert with other resources, including possibly remote resources (grid, cloud, HPC, laptop, ...)
- Event data service abstracts away the efficient delivery of inputs, aynchronously
  - No job/file level data colocation requirement
- Near real time output streaming to safe high-availability output stores
  - No long latencies in output transfers... outputs incrementally and concurrently move to their destination, with less chance of trapped outputs
- Outputs aggregate at standard, trusted, web-accessible storage points
  - No matter where the workers contributing to them are running
  - Output aggregation and merging doesn't constrain which sites participate
- *All that said – an initial early target could be psub with conventional jobs, not ES – needs study*

And briefly back to data access...

- Near term...
  - The Zoology
- Longer term...
  - Can we clarify a strategy?
  - Which creatures belong and which don't
  - “ddm-get” and what all is under the hood
  - Start some R&D on a EDS/CDN like approach?

Summary thoughts

- We should have an ADC view on what services can effectively support beyond-pledge/T3/local resources while integrating well in ADC
- The short term is pretty clear, we use what we have, what BP/T3 folks have put together
- For the longer term, new possibilities are opening with e.g. the ES work
- If we agree on a strategy it could very helpfully focus/filter work that would otherwise proceed with less direction towards optimally useful tools
- If we agree on long term directions we could then when making a real plan, look for some medium term objectives
  - e.g. if we go for psub, does a medium term 'conventional job' based psub make sense as a precursor to ES based psub