BaBar -Overview of a running (hybrid) system

Peter Elmer

Princeton University

5 June, 2002

Overview

- BaBar uses Objectivity for the eventstore, conditions, detector configurations, ambient data and for temporary storage of information used in calibrations
- BaBar is primarily interested in following the LCG persistency framework area as part of developing and maintaining an Objectivity contingency plan
- I'll try to describe here the BaBar production and analysis system and some of the things that shaped it
- I'm not going to discuss the Objectivity based eventstore in detail. That has been done before and I'm not the right person for that anyway
- I'm not going to show any class diagrams

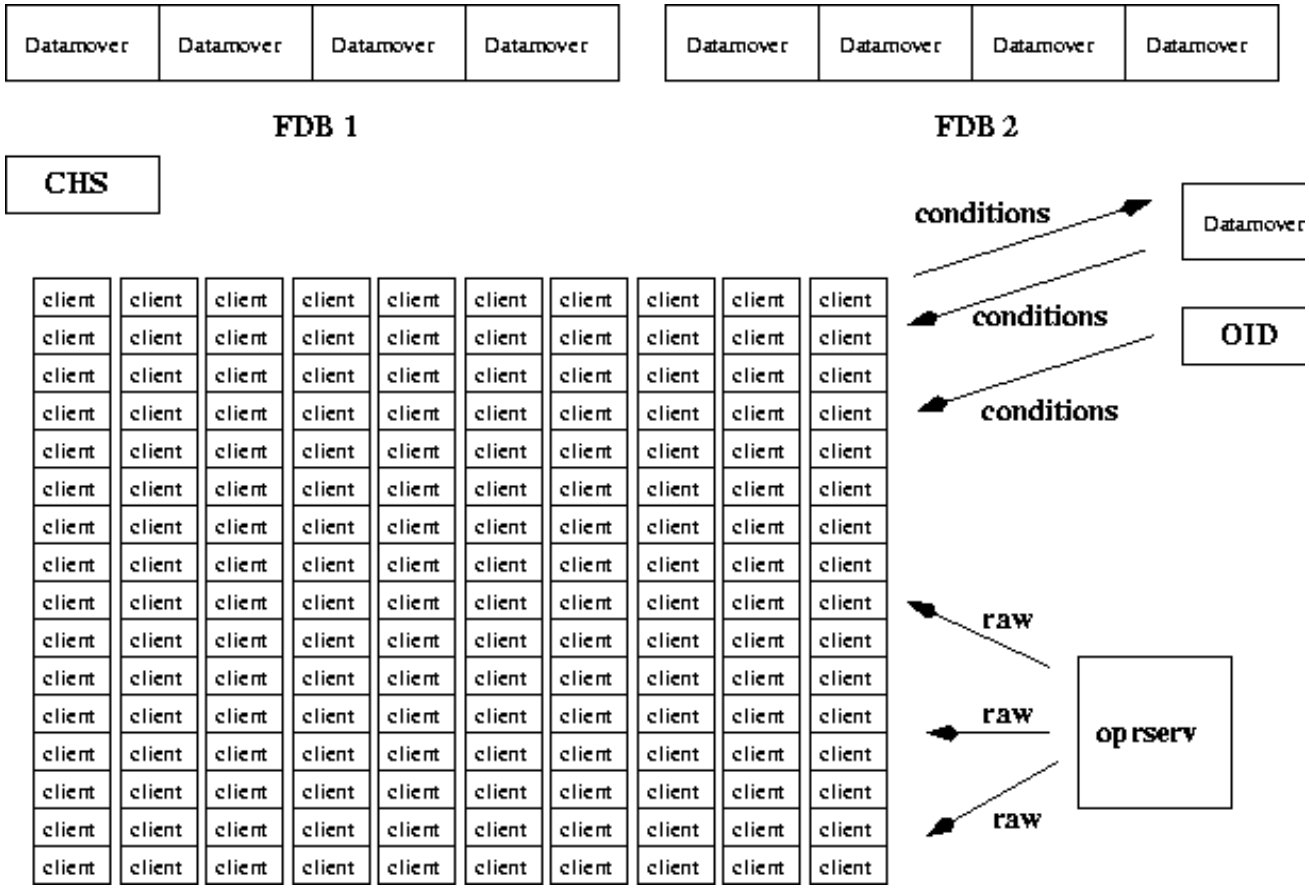
Experiment deployment model

- Timescale: in production/contingency
- Volume: 575TB (Objectivity) + 15TB (ROOT) + 100TB (raw data, "xtc" files)
- Files: 350k (Objectivity), a lot (ROOT), ~20k (xtc/raw)
- Distribution: 4 ``Tier A", 2 "Tier B", 15-20 simulation production sites, ~15 ``Tier C"
- Recovery from job failures: yes (method varies)
- Number of population jobs: 20M(?) so far
- Use of Refs: yes
- (the rest of this talk is actually details about our current ``deployment model")

Trigger/Online

- L1 Trigger -hardware
- L3 Trigger -software
  - running on farm of unix processors
  - Solaris/sparc from beginning of data taking, recently demonstrated running on linux/intel
  - Design output was 100Hz, now surpassed
  - Writes raw data to a custom flat file ("xtc") as a buffer, this is saved to mass storage and is the input to Prompt Reconstruction (PR). This is the canonical raw data format for BaBar.

Prompt Reconstruction (PR) Farm

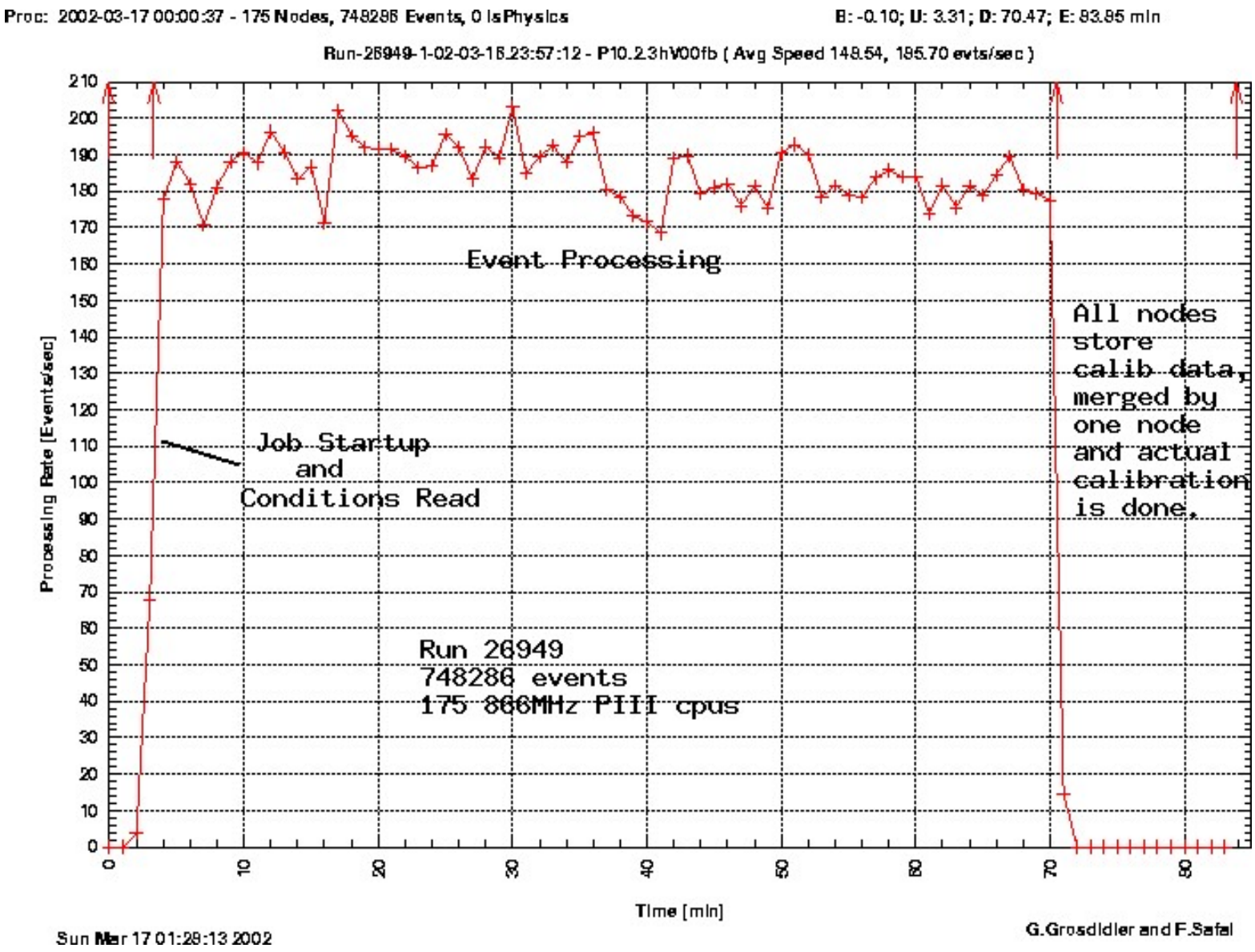


- Runs processed sequentially in the order they are taken, raw data taken from ``xtc'' file (retrieved from disk cache or from HPSS)
- ``Rolling'' calibrations from one run used by next

Event Data written to Objy in PR

- TAG - tag data, used for analysis (1kB/event)
- AOD - ``micro data'', widely used for analysis (5kB/event)
- ESD - ``mini data'', used currently mostly for calibrations, new more complete version going into production will likely be more interesting for analysis (may replace AOD, 10kB/event)
- REC -reco data (deprecated, 150kB/event)
- RAW - Objy copy of raw data (may now be turned off, 50kB/event)

Typical run in PR

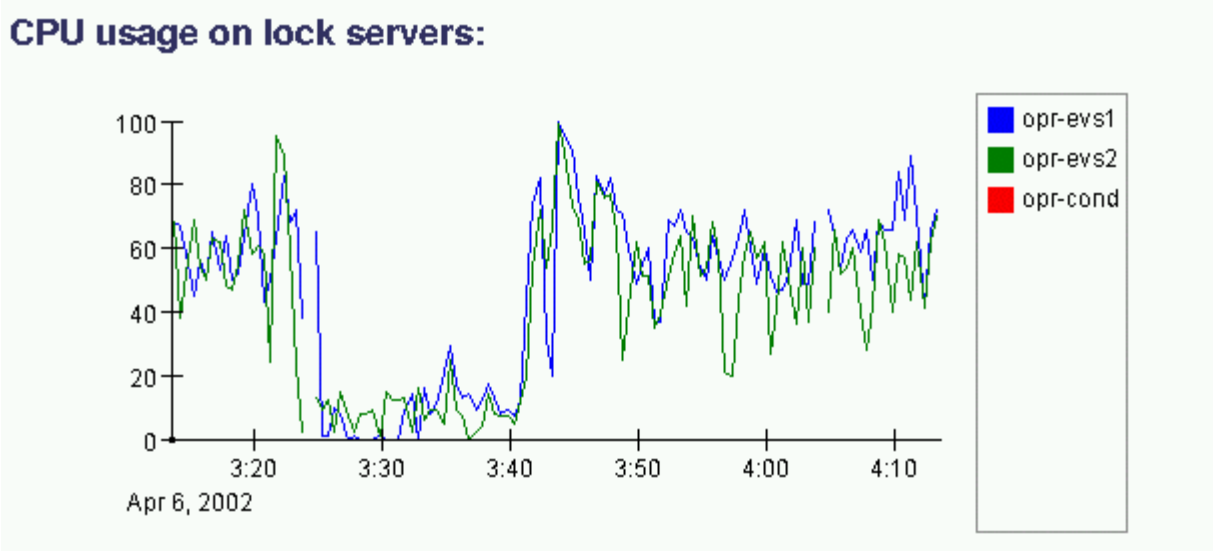


PR Hardware today

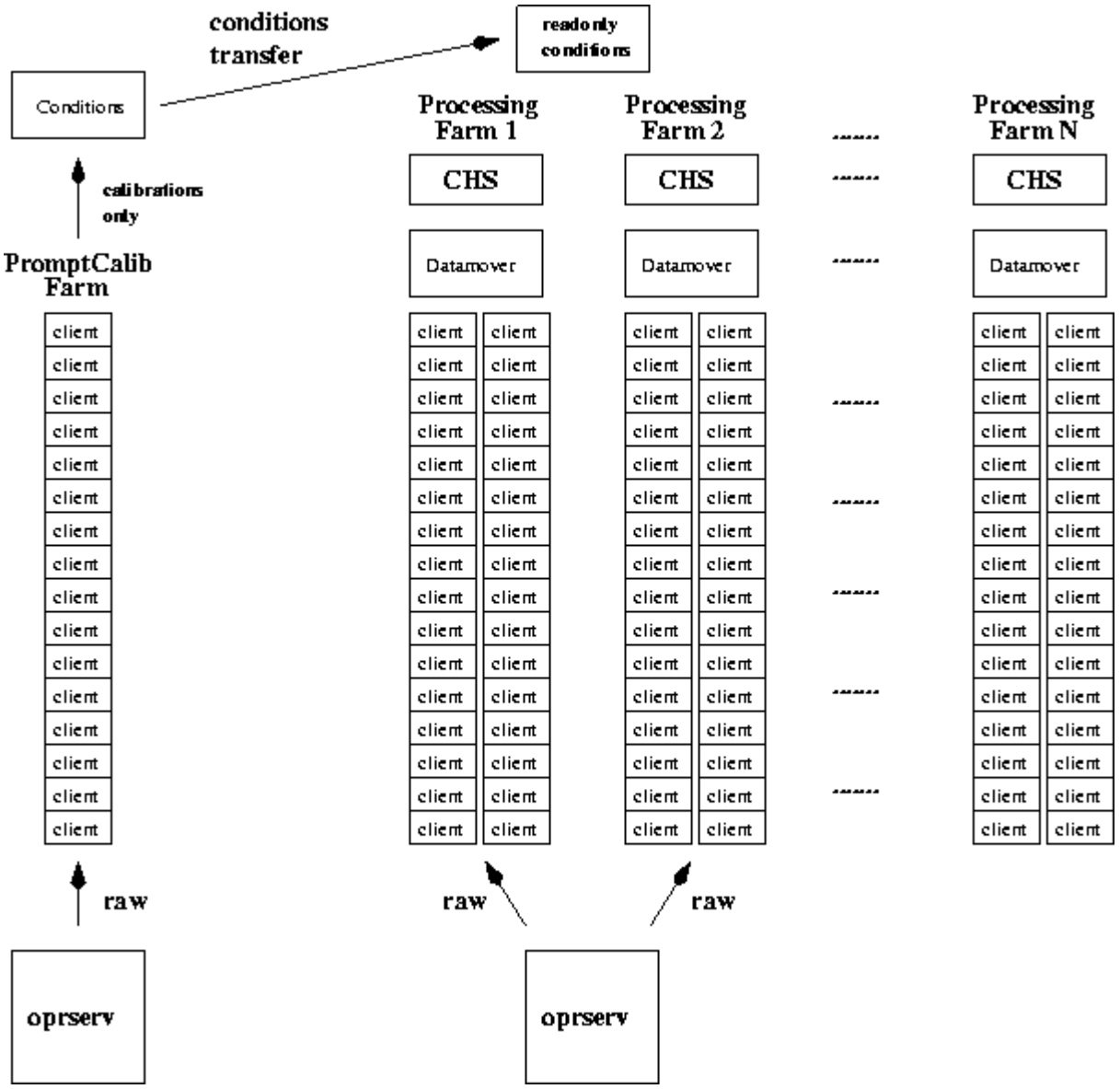
- 8 datamovers (4 cpu, 450MHz sparc), each with 1 1TB of disk space attached (perhaps overkill)
- 175 client cpus - 866 MHz PIII cpus, 1 GB (Linux)
- 3 lockserver machines - 1 cpu 450MHz sparc
- 1 CHS/OID server - 2 cpu 450MHz sparc
- 1 catalog server - 2 cpu 450MHz sparc
- 1 journal server - 2 cpu 450MHz sparc
- 2 oprserv machines -2 cpu, 400MHz sparc, 400 GB of disk (2 machines perhaps overkill)

Scaling limitations

- Data servers: limitations can be dealt with by simply adding more data servers (optimization done with corba clustering hint server)
- Lockserver limitation: partition farm such that half of nodes write to one FDB and the other half to a separate FDB and piece the resultant collections back together in a ``bridge federation'' for use in analysis



New PR architecture



- Two passes
- PromptCalib: process fixed rate of some event types, runs processed in the order taken, write calibrations only
- Bulk reco: parallel farms for event reco. (in limit, it is batch processing)

Simulation Production (SP)

- Simpler system: N (independent) batch clients writing data to one server, conditions are read only. Only the Objectivity federation ties the output of jobs together.
- Currently do production in 3 separate steps, each a separate batch job
  - Geant simulation
  - Detector response simulation
  - Reconstruction
- This was useful when the rate of change of the code base was higher, now moving to single monolithic executable which does all three ``steps'' in one job

Simulation Production (2)

- SP done at 15-20 sites involving something like 700 cpus
- We were shipping all data (analysis data as well as ``bulk'' data) back to SLAC, was heading towards 1 TB/day. We no longer ship the bulk data databases back to SLAC, but delete it instead locally at the production site. (i.e. it doesn't go out over the WAN)
- Single pass monolithic executable will go into production soon, allowing us to avoid also sending the (intermediate) bulk data over the LAN

Kanga/Root format

- Due to initial difficulties in using Objectivity for both production and (in particular) analysis away from SLAC, in 1999/2000 an alternative ROOT I/O based analysis data format was developed (Kanga)
- Limited to ``micro data'' (TAG and AOD)
- Emphasis on ease of import and use at University sites (to date Objectivity is only used for analysis at SLAC and In2p3)
- Constructed such that a given analysis job could switch back and forth between the Objy and ROOT based eventstores very easily (transient-persistent separation)

Kanga/Root format

- A collection of TAG/AOD data is stored as a single tree per file, one file output per job, one job per data/MC run.
- Simple integer based references using a registry implemented, but not used in TAG/AOD data
- Data is read from Objectivity eventstore and written out in ``Kanga'' format by dedicated converter jobs
- As used in production, redoing the tag bits due to new selections means rewriting all TAG/AOD data
- The system supports both streaming of multiple files as well as simple pointer collections, but neither of these is used (yet) in production system.

Analysis jobs: notes

- Users queries a collection database in RDBMS (for both Objectivity and Kanga eventstores) with a standalone tool and receives a list of collections matching the specified criteria (skim, release, run number, ...). The Framework job then runs on the eventstore (Objy or Kanga) and does not interact with the RDBMS.
- Analysis data resides on servers which are physically separate from those from production, so data becomes available in a weekly sweep, requiring an outage of both the production and analysis federations.

Production reskimming

- Approximately once/year since data taking began, the physics selections are revisited (new ones added, existing selections are changed, etc.)
- This seems to happen asynchronously from our ``production releases'' for use in PR and SP
- When these selections converge, they are put into production but existing real and simulated data must be reskimmed, i.e. tag bits are recalculated and physics selection collections are redone
- In Objectivity eventstore, TAG is rewritten and the rest of the event is borrowed from original copy. In Kanga, TAG/AOD is rewritten.

Streaming strategy

- Initially 4 data streams were output from PR (isMultihadron, isCalibEvent, ...). This was mostly useful for staging, but was not well matched to export of small data samples.
- In early 2001, switched to writing 20 output streams (with event duplication between them), such that a site could import a single ``stream'' as a data subset. ``Solved'' problem of useful deep copy due to DBID limitation of Objectivity.
- This strategy was to bootstrap use of Objectivity for analysis at ``Tier C'' (i.e. University) sites

Streaming Strategy (2)

- Problems with 20 stream strategy:
  - duplication at sites with all streams on disk requires a factor 2.5 more disk space,
  - a ``stream'' contained several physics selections, more than a small site really wanted (100+ selections are grouped into the 20 streams to minimize duplication between streams)
  - Increased significantly the number of open files in PR and thus impacted performance
- Due to large size of disk resident data in Objectivity eventstore, we never actually turned on the event duplication between streams (wrote 20 streams with ``first hit, owns the event'' placement)

Streaming strategy (3)

- In the past 1.5 years, the technology to use multiple federations within a single job was put into use.
- We are now switching back to a scheme in which we write 4 streams and 100+ skim (pointer) collections for physics selections. Data can be exported by running a deep copy job over one of the physics selections and writing the output to a separate federation.
- Not yet 100% clear if data duplication is needed anyway for performance reasons or not....

Analysis use

- I've described the production systems which produce terabytes of analysis level data for use in analysis and some analysis related activities.
- What does the average physicist doing analysis do once the data is available in one of the two (relatively sophisticated) eventstores?
- He/she immediately runs over it and dumps the data out to ntuples/root files in his/her own custom format (and asks for terabytes of disks to store it)
- To some extent this has crystallized into ``ntuple productions'' by each analysis working group
- Work on standalone ntuples/root files interactively and then pass through data a second time creating reduced collections

Analysis use (2)

- We provide standardized classes for persisting ``composite'' candidates (e.g. a reconstructed B meson in an analysis program) as well as ``attributes'' for user defined information in both the Objectivity and Kanga/ROOT eventstores.
- Not widely used, people use ntuples and custom root files, but interest seems to be increasing a bit.
- Mini (ESD) date is significantly more sophisticated and may replace some of this as analyses are becoming more sophisticated

Objectivity contingency plan

- As a first pass, we are interested in exploring if we could extend our existing ``Kanga'' ROOT based persistency framework as a contingency plan for our eventstore, rather than importing something new lock, stock and barrel.
- However, as this project becomes more mature, it will clearly become more interesting as the ``contingency plan''
- Currently we have no plan to migrate away from Objectivity.

General issues for Kanga extension

- Well adapted to tag/micro analyses, but not everything we would need from a RDBMS/ROOT replacement for Objectivity eventstore, some considerations are:
- All components in one file (no tag, aod, esd, raw, sim, tru separation). no esd, raw, sim, tru...
- No means to borrow components from an event
- File size is poor match to HPSS (too small)
- No automatic staging mechanism
- Production pointer collections (in progress)
- How do we insure that every event is written once and only once in PR? (checkpointing/transactions)