Microsoft Research

Building Reliable Data Pipelines for Managing Community Data using Scientific Workflows

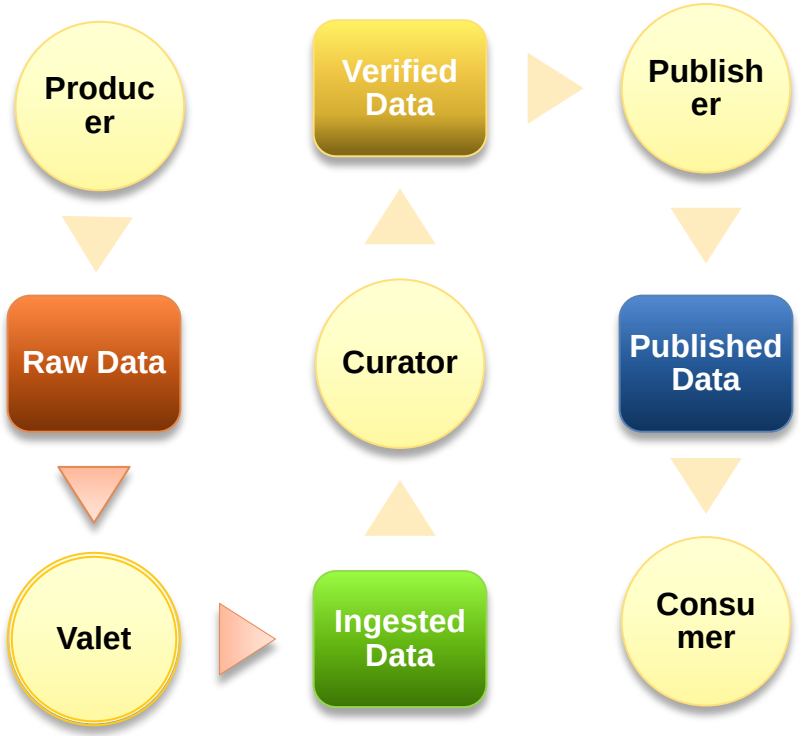
Yogesh Simmhan
eScience Group | Microsoft Research

Catharine van Ingen, Roger Barga, Microsoft Research
Alex Szalay, Johns Hopkins University
Jim Heasley, University of Hawaii

External Research
eScience

Science Data Lifecycle

- Science is producing *vast quantities* of data
- Data needs to be *ingested & published* for community use
- Different *user roles*, different domain/IT *expertise*
- *Provenance*, *reliability*, *semantics* are key!
  - Commodity hardware

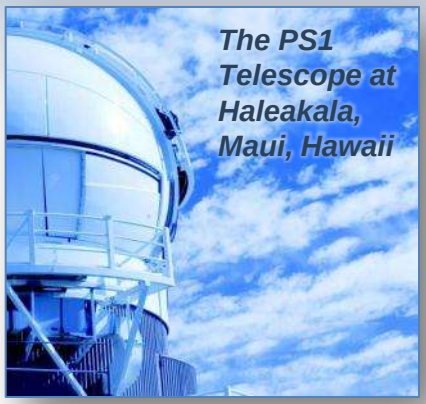


Workflows for Data Management

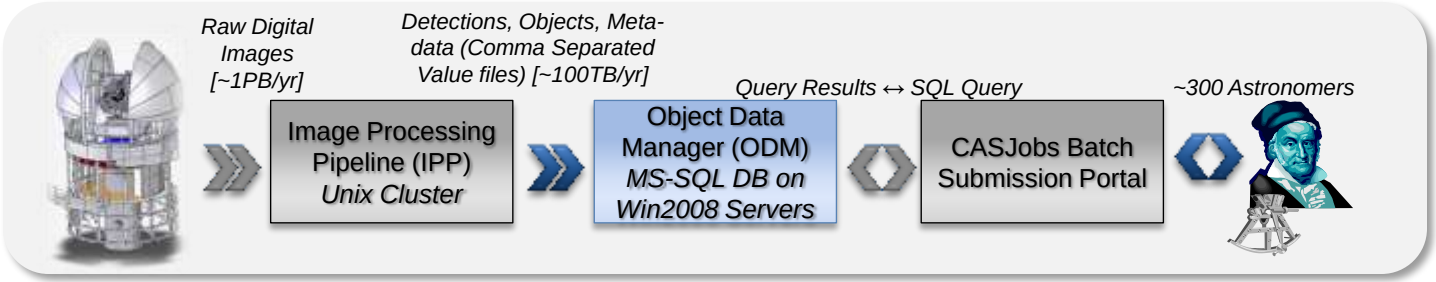
- *Scientific workflows* are popular with scientists
- *Composable* & *reusable* pipelines supporting *data flows*
- Diverse execution environments, *tracking* & monitoring
- Workflows are suitable for *data ingest*
  - But *reliability* is important for "valet" workflows
- **Workflow design models** for reliability in distributed environment
- **Workflow framework features** for fault tolerance

Pan-STARRS eScience Application

- **Pan**oramic **S**urvey **T**elescope **A**nd **R**apid Response **S**ystem
  - Discover dangerous asteroids & comets.
  - Use data for Solar System, cosmology study

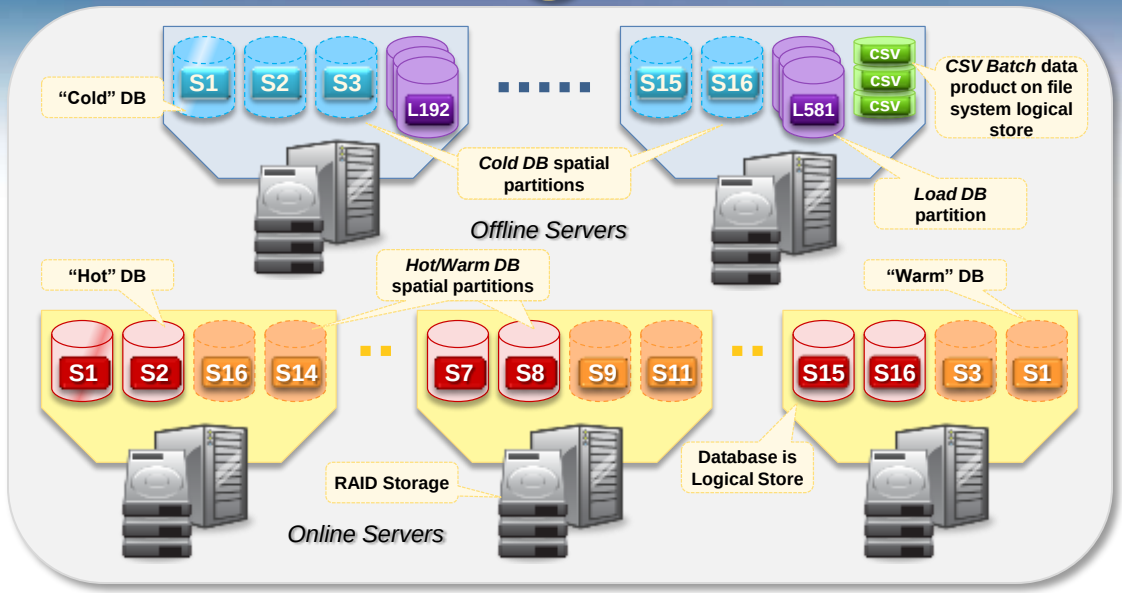


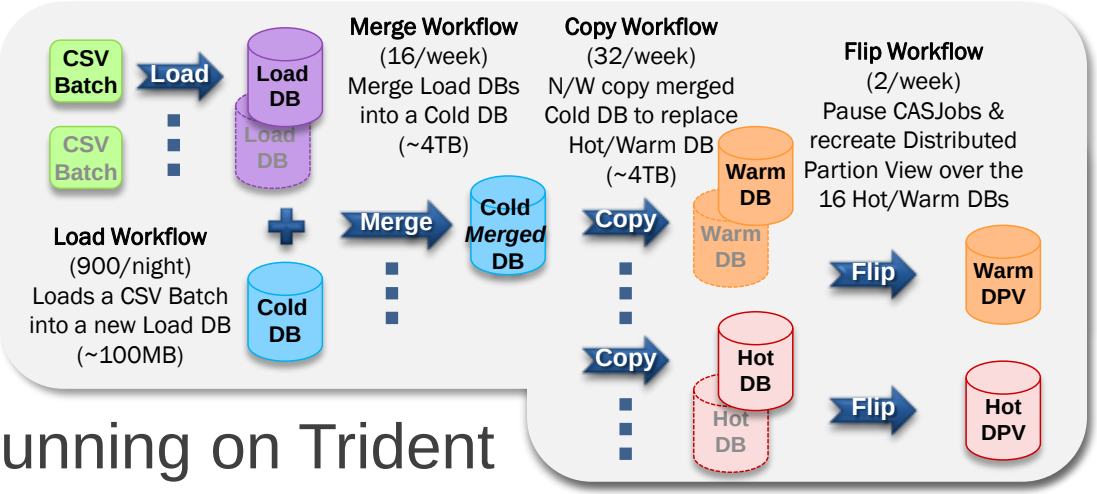
- One of the *largest visible light telescopes*
  - $^{2}/_{3}^{rds}$ *of sky* scanned *thrice a month*
- Large, growing data collection
  - Annual *1PB* of raw images | *30TB* of community data | 5.5B objects | 350B detections
  - Daily *2.5TB* image data | *100GB* of shared data | 1000 images | 150M detections



Data Layout & Loading in PS

- *Data partitioned* as dist-ributed DB with views
- WinHPC Cluster,MSSQL
- 3 copies: hot/cold/warm
- *CSV files* arrive from Image Pipeline: 1k daily
- Each file *loaded* into one *'Load' database*
- Load DBs from each week *merged* with prior data in *'Cold' database*
- Copies of new Cold DB surfaced to users
- Workflows for data ops: *Load, Merge, Copy, Flip* running on Trident





Reliable Workflow Design

Well defined, Well tested workflows

- Valet workflows run *repeatedly*, their impact is *cumulative*
- Workflows and activities *do not change* often
- Changes need to be *synchronized* with repository schedule
- *Testing* is crucial
- E.g. Load workflow run 900 times/day. Faults can compound.

Granular, Reusable workflows

- *Avoid custom activities* for ad hoc tasks
- Easier to test and maintain *library of core activities*
- Separate *policy* from *mechanism* E.g. static n/w files
- E.g. Copy & Flip are separate workflows. Preamble for Copy is separate from actual copy action.

Reliable Workflow Design…

Workflows as Data State Machines

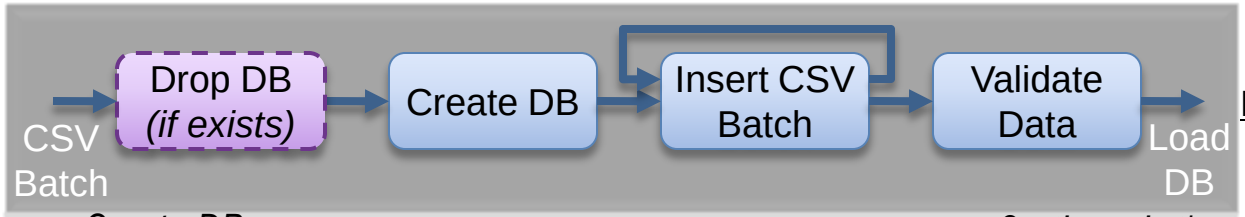
- *State of repository* depends on state of its constituent resources
- Workflows operate on *state machines*
  - *Data* containers have *states*
  - *Workflows* and activities cause *state transitions*
- Instantly determine state of system
  - Easier to *interpret workflow progress*, complex interactions
  - *Impact of workflows* on states are different. E.g. Load vs. Merge workflow failures
- Easily define *fault paths* based on data state
  - Goal: Recovery from fault state to clean state

Reliable Workflow Design…

Workflows as Data State Machines…

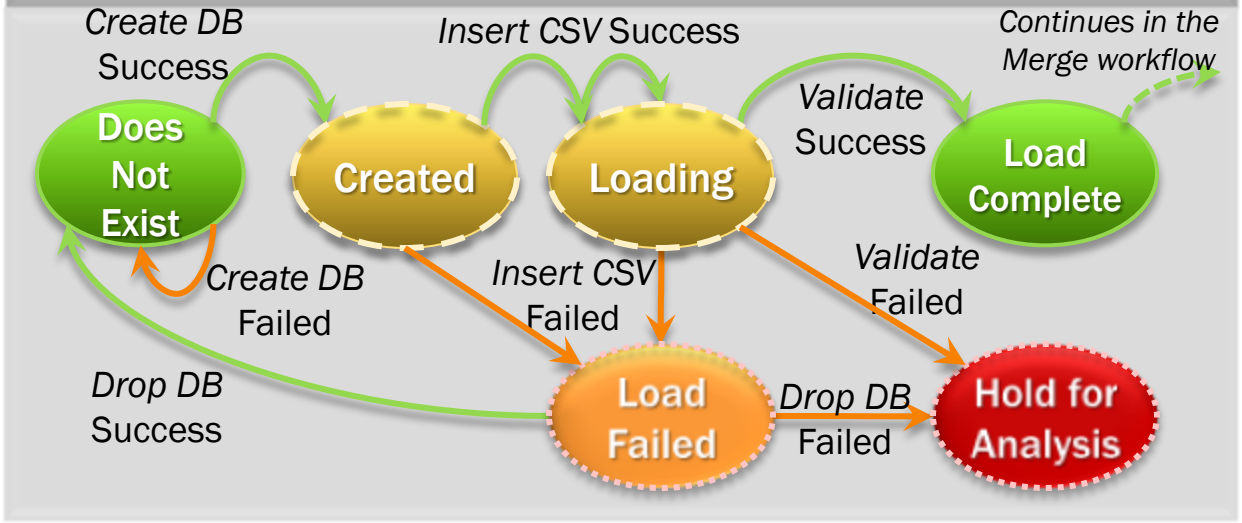
- *Simple state definitions* depending on stage of loading data, recovery model used
- Activities should cause *at most one* state change
- States used for recovery workflow, policy workflow & display

Load Workflow Activities



State Machine for Load DB

State transitions take place when activities execute.



Activity Type
Blue/Solid : Mainline Code
Purple/Dashed: Recovery Preamble

Success States
Green/Solid : Clean
Yellow/Dashed: In Flight

Failed States
Orange/Dotted: Recoverable
Red/Dotted: Unrecoverable

Reliable Workflow Design…

Workflow Recovery Baked into Design

- Faults are a *fact of life* in distributed systems: hardware loss, transient failure, data errors
- Recovery as part of normal workflow makes handling faults a *routine action*
- Coarser (simpler) approach to recovery
- Different recovery designs
  - Re-Execute Idempotent
  - Resume Idempotent
  - Recover-Resume Idempotent
  - Independent Recovery

Reliable Workflow Design…

Workflow Recovery Baked into Design…

- Re-Execute Idempotent Recovery
  - Idempotent *workflows* that can be rerun without side-effects
  - Input data states valid, no in-place update
  - Retry limits
  - E.g. Flip workflow for DPV reconstruction
- Resume Idempotent Recovery
  - Idempotent *activities* allow “goto” at start
  - *Cost*, *complexity* of resume vs. re-execute
  - E.g. Copy workflow for parallel file copies

Reliable Workflow Design…

Workflow Recovery Baked into Design…

- Recover & Resume
  - Separate activity(s) to *rollback* to initial state
  - *Reduce* the problem to resume/re-execute
  - Passive Recovery: *Rollback activity(s)* at start of workflow. E.g. Load workflow drop database
  - Active Recovery: *Rollback workflow* captures operations to undo dirty state. E.g. Merge workflow on machine failure
  - Fail fast vs. Fault tolerant activities
- Independent Recovery
  - *Complex faults* requiring global synchronization
  - Sanity check, cleanup, consistent states
  - Manually coordinated. E.g. Loss of merged Cold DB

Reliability Features in Workflow Systems

- Specific requirements to support data valet tasks. E.g. tracking more imortant than *ad hoc* composition

Provenance Collection

- *Track operations* across data, activities, workflows, distributed *temporally & spatially*
- Record inputs and outputs to activities
- Valet workflows need *fine grained*, system level provenance (logging)
- Reliable, scalable provenance collection

Reliability Features in Workflow Systems…

Provenance Collection for Data State Tracking

- Provenance provides record of state transitions
- Provenance can be mined for current state of data resources. Recovery based on it.
- *Direct State Recording*: Activities expose the state transition they effect as in/outputs
  - Pro: Easily queried; Con: Reusability, local view
- *Indirect State Recording*: External mapping from activity execution to state transition
  - Pro: Reusable, WF level; Con: Complexity
- PS uses indirect model

Reliability Features in Workflow Systems…

Provenance Collection for Forensics

- *Investigate* cause of distributed, repetitive faults
- Low level, system provenance ~monitoring, logging, tracing: disk, machine, services, I/O

Scalable & Resilient Provenance Collection

- *Long running, continuous* workflow execution
- Fine-grained collection
- Stored till data release, even *instrument lifetime*. E.g. Status of 30,000 Load workflows
- *Efficient querying* for recovery, current state
- Provenance loss → *Doubtful repository state*
- PS has realtime SQL replication of provenance

Reliability Features in Workflow Systems…

Support for Workflow Recovery Models

- Re-run previous workflow with same inputs
- Well defined exceptions & recovery execution paths
  - E.g Recover & Resume workflows
- Identify exception source. Transmit it, reliably, to initiate recovery.
- Low level, system provenance ~monitoring, logging, tracing: disk, machine, services, I/O

Fail-fast guarantees

- Workflows, activities must fail fast
- Framework must halt on errors: timeout, liveliness, network connectivity
  - Early, accurate detection → Early recovery
- Trident Workflow configured to fail fast for PS
  - Synchronous provenance logging to Registry
  - HPC fault event monitoring

Related Work

- Reliability of repository data management less studied compared to other aspects
- Data reliability
  - Replication in Grids, Clouds – replicated file system, database
  - Transparency vs. Update, recovery overhead
  - Specialized for repositories operate only by workflows, not general purpose applications
- Application reliability
  - Over-provisioning workflows. Infeasible.
  - Checkpoint – restart. Insufficient.
  - Transactional workflows. Do not support data states.

Summary

- Data management in eScience is complex
- Large shared data respositories on commodity clusters more common
- Data valets have special needs from workflows
- Goal driven approach to workflow design using data state machines
- Simple models for reliability & recovery go a long way
- Trident workflow provides tools to support both scientist & valet users

Microsoft Research

Questions

Thank you!

Acknowledgements

Maria Nieto-Santisteban, Richard Wilton, Sue Werner, *Johns Hopkins University*

Conrad Holmberg, *University of Hawaii*

Dean Guo, Nelson Araujo, Jared Jackson, *Microsoft Research Trident Workflow Team*

www.ps1sc.org

http://research.microsoft.com/en-us/collaboration/tools/trident.aspx