Active Data: Data Life Cycle Management Across Heterogeneous Systems and Infrastructures

Anthony Simonet, Gilles Fedak (INRIA)
Matei Ripeanu, Samer Al-Kiswany (UCB)
Kyle Chard, Ian Foster (ANL/UC)

Hot Topics in High-Performance Distributed Computing Workshop
IBM Almaden Research Center
San Jose, California
March 12, 2015
Huge and growing volume of information originating from multiple sources.

... or Big Bottlenecks?
- how to scale the infrastructure?
  - end-to-end performance improvement, inter-system optimization.
- how to improve productivity of data-intensive scientist?
  - data-oriented programming language, data quality, improve automation and errors recovery.
**Data Life Cycle**

**Definition**

*Data Life Cycle* (DLC) is the course of operational stages through which data pass from the time when they enter a set of systems to the time when they leave it.

**Challenges**:

- Expose high level view DLC across distributed systems and infrastructures
- Expose interactions between the infrastructure and the DLC (e.g., failures)
Active Data:

- Allow to reason about data sets handled by heterogeneous software and infrastructures.
- A **formal model** that captures the essential life cycle stages and properties: creation, deletion, faults, replication, error checking . . .
- **programming model** to develop easily data life cycle management applications.
- Allows legacy systems to expose their intrinsic data life cycle.
System programmers expose their system’s internal data life cycle with a model based on Petri Nets.

A *Life Cycle Model* is made of

- **Places**: data states
- **Transitions**: data operations

Each token has a unique identifier, corresponding to the actual data item’s.
System programmers expose their system’s internal data life cycle with a model based on Petri Nets.

A **Life Cycle Model** is made of:

- **Places**: data states
- **Transitions**: data operations

A transition is fired whenever a data state changes.
System programmers expose their system’s internal data life cycle with a model based on Petri Nets.

A *Life Cycle Model* is made of:

- **Places**: data states
- **Transitions**: data operations

```java
public void handler() {
    computeMD5();
}
```

Code may be plugged by clients to transitions. It is executed whenever the transition is fired.
Framework features:

- Captures data events in legacy systems
- High-level *life cycle-centered* view of data
  - Single namespace for all the files, datasets and metadata
- Powerful filters based on **Data Tags**
  - Install *Taggers* on Transitions
  - *Guarded Transitions*: only executes on token which have specific tags.
- Publish/subscribe transitions
- Custom user reaction to data progress
  - Custom code execution
  - Custom notifications (twitter, email, gdoc, ifttt . . . )
3 to 5 TB of data per week on this detector

- Raw data are pre-processed and registered in the Globus Catalog:
- Data are curated by several applications
- Data are shared amongst scientific user
4 goals (that would otherwise require a lot of scripting and hacking):

- Monitoring Data Set Progress
- Better Automation
- Sharing & Notification
- Error Discovery & Recovery
Data life cycle model composed of 6 systems.
Example scenario

Recover from system-wide errors: faulty acquired files are detected only after Swift fails to process them.

In this situation, the user manually:

- Drops the whole dataset
- Removes any associated file and metadata
- Re-acquire the dataset using the same parameters
Data life cycle model composed of 6 systems.
TransitionHandler handler = new TransitionHandler() {
    public void handler(Transition t, boolean isLocal, Token[] inTokens, Token[] outTokens) {
        // Get the dataset identifier
        LifeCycle lc = ad.getLifeCycle(inTokens[0]);
        datasetId = lc.getTokens("Shared storage.Created")[0].getUid();

        // Remove the dataset annotations from the catalog
        String url = "https://catalog.globus.org/dataset/" + datasetId;
        Runtime r = Runtime.getRuntime();
        Process p = r.exec("catalog_client.py remove " + url);
        p.waitFor();

        // Locally, remove the datasets
        String path = "~/aps/" + datasetId;
        FileUtils.deleteDirectory(new File(path));

        // Publish the "Detector.End"
        Token root = lc.getTokens("Detector.Created")[0];
        ad.publishTransition("Detector.End", lc);

        // Notify the user
        sendEmail("user@server.com", "APS - Corrupted dataset " + datasetId);
    }
};

HandlerGuard guard = new HandlerGuard() {
    public boolean accept(Transition t, Token[] inTokens, Token[] outTokens) {
        return input[0].hasTag("failure corrupted");
    }
}

ad.subscribeTo("Swift.Failure", handler, guard);
Active Data

- allows to expose Data Life Cycle across heterogeneous systems and infrastructures
- *transition-based* programming model for DLC management application
  - Monitoring, automation, error detection & recovery
  - X-systems optimizations: incremental computing, data staging, caching, throttling etc... 

Perspectives:

- Use AD to deploy data management software stack on IaaS (Asma Ben Cheick, Heithem Abbes, Univ. Tunis)
- Big Data Apache stack X-optimization (H. He, CAS, Beijing)
- Volunteer & crowd computing (M. Moca, BBU, Romania)
Thank you!

Questions?