Active Data

A Data-Centric Approach to Data Life-Cycle Management

Anthony Simonet[1] Gilles Fedak[1]
Matei Ripeanu[2] Samer Al-Kiswany[2]

[1]Inria, ENS Lyon, University of Lyon [2]University of British Columbia

November 26th, 2013

Outline

Introduction
- Data Life Cycle Management
- Use-case
- Requirements

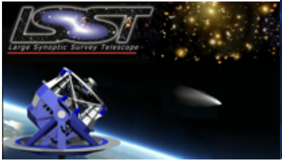
Active Data
- Active Data: principles & features

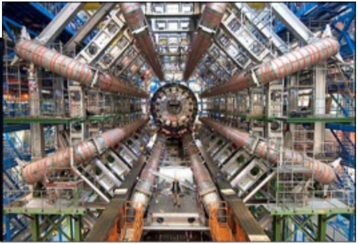
Discussion
- Advantages
- Limitations

Conclusion
- Related works
- Conclusion

Big Data

- Science and Industry have become data-intensive
  - Volume of data produced by science and industry grows exponentially
  - How to store this *deluge* of data?
  - How to extract knowledge and sense?
  - How to make data valuable?
- Some examples
  - CERN's Large Hadron Collider: 1.5PB/week
  - Large Synoptic Survey Telescope, Chile: 30 TB/night
  - Billion edge social network graphs
  - Searching and mining the Web



Data Life Cycle

Data Life Cycle

- Creation/Acquisition
- Transfer
- Replication
- Disposal/Archiving

Definition

The life cycle is the course of operational stages through which data pass from the time when they enter a system to the time when they leave it.

Data Life Cycle Management

Complicated scenarios

- Execution of workflow
- Complex interactions between software
- Need to quickly react to operational events

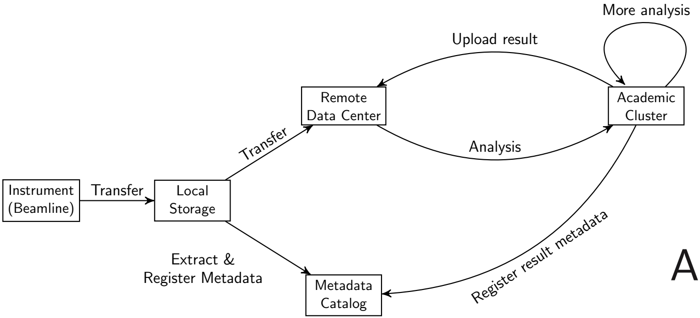
Ad-hoc task-centric approaches

- Hard to program, maintain and debug
- No formal specification
- Complicates interactions between systems

Data Life Cycle Use-case

Example: the Advanced Photon Source at Argonne National Lab

- 100TB of raw data per day
- Raw data are preprocessed and registered in a Globus dataset catalog
- Data are analyzed by various applications
- Results are stored in the dataset catalog and shared



Use-case

Task Centric

- Independent scripts
- Hard to program, maintain, verify
- Coarse granularity

Vs

Data Centric

- Express data-dependancies
- Cross data-center coordination
- User-level fault-tolerance
- Incremental processing

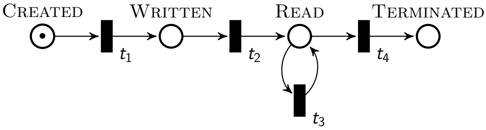
Requirements

Challenges: a perfect system should...

- Simply represent the life cycle of data distributed across different data centers and systems
- Simplify DLM modeling and reasoning
- Hide the complexity resulting from using different infrastructures and systems
- Be easy to integrate with existing systems

Active Data principles

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** and **Transitions**

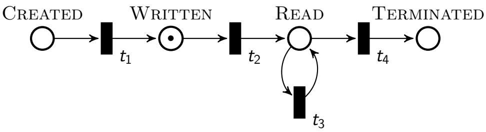


Each token has a unique identifier, corresponding to the actual data item's.

Active Data principles

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** and **Transitions**

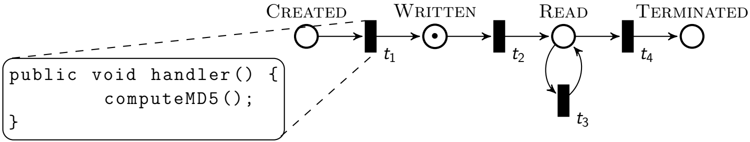


A transition is fired whenever a data state changes.

Active Data principles

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** and **Transitions**



Code may be plugged by clients to transitions.

It is executed whenever the transition is fired.

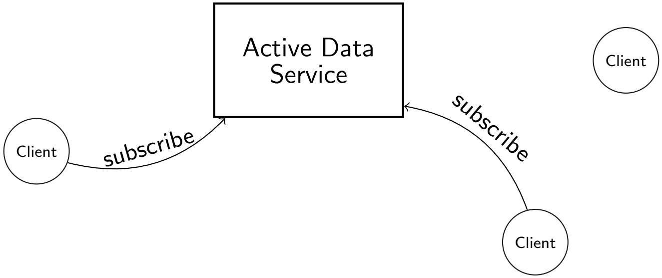
Active Data features

The Active Data programming model and runtime environment:

- Allows to react to life cycle progression
- Exposes transparently distributed data sets
- Can be integrated with existing systems
- Has scalable performance and minimum overhead over existing systems

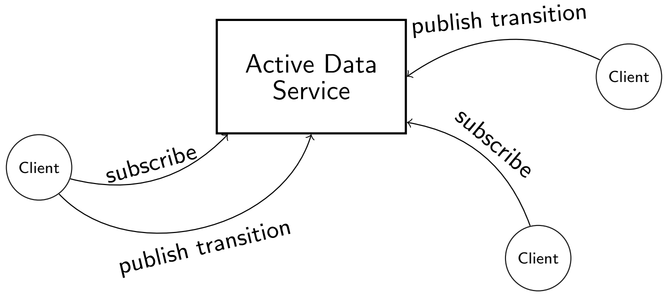
Implementation

- Prototype implemented in Java ($\simeq$ 2,800 LOC)
- Client/Service communication is Publish/Subscribe
- 2 types of subscription:
  - Every transitions for a given data item
  - Every data items for a given transition



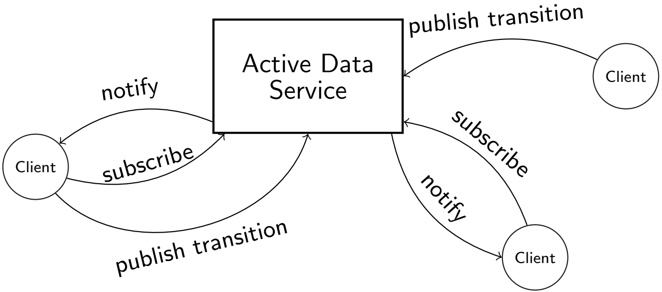
Implementation

- Several ways to publish transitions
  - Instrument the code
  - Read the logs
  - Rely on an existing notification system
- The service orders transitions by time of arrival



Implementation

- Clients run transition handler code locally
- Transition handlers are executed
  - Serially
  - In a blocking way
  - In the order transitions were published



Performance evaluation: Throughput

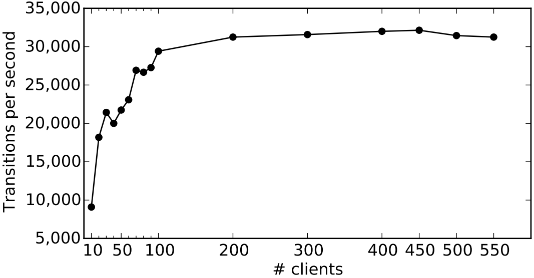


Figure: Average number of transitions per second handled by the Active Data Service

Clients publish 10,000 transitions in a row without pausing.

Performance evaluation: Throughput

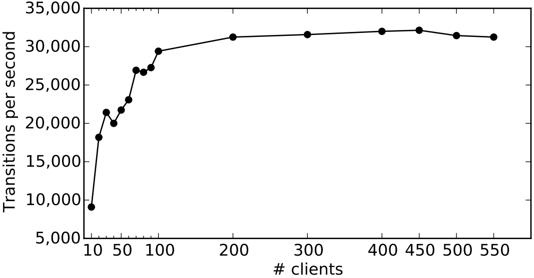


Figure: Average number of transitions per second handled by the Active Data Service

The prototype scales up to 30,000 transitions per seconds.

Supported Systems

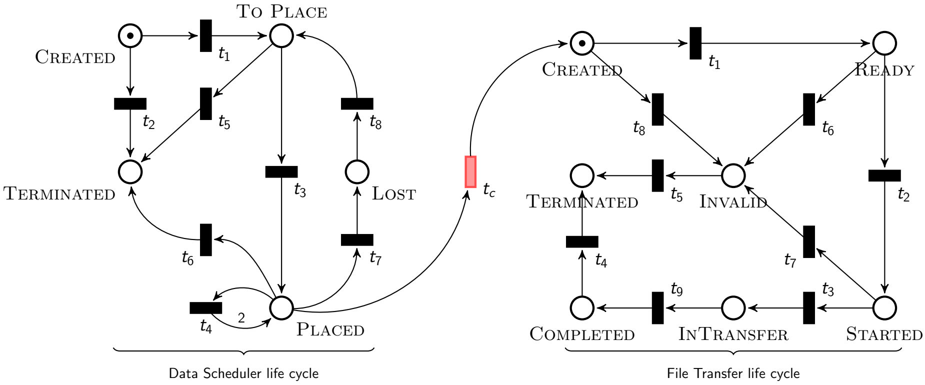
- 4 systems
  - BitDew
  - inotify
  - Globus Online
  - iRODS
- 4 use-cases
  - Storage cache for Amazon S3
  - Distributed data throttling
  - Incremental MapReduce
  - Data provenance

Supported Systems

- 4 systems
  - **BitDew**
  - inotify
  - Globus Online
  - iRODS
- 4 use-cases
  - Storage cache for Amazon S3
  - Distributed data throttling
  - Incremental MapReduce
  - Data provenance

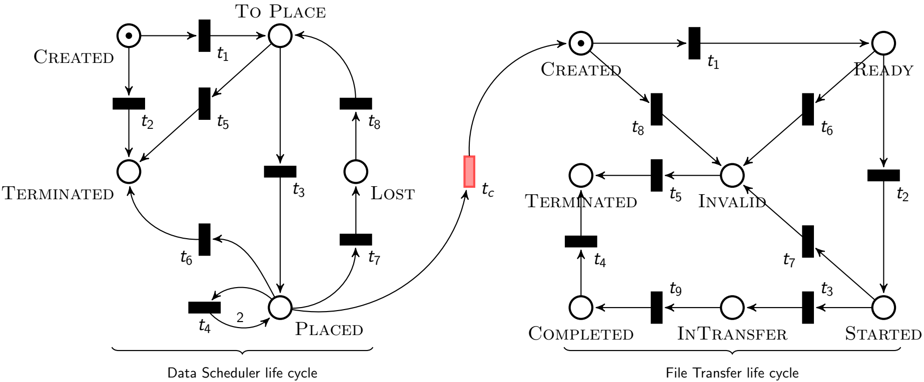
Exemple: BitDew

Complete life cycle models inferred from the code.



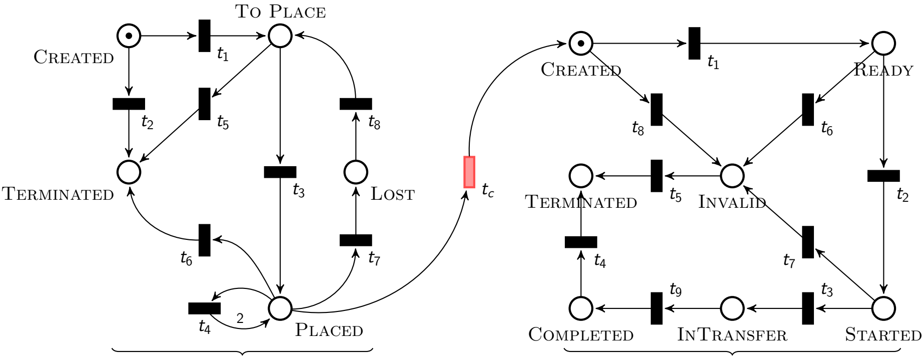
Exemple: BitDew

Composition of the life cycle for data management and the life cycle for data transfers.



Exemple: BitDew

Replication and fault tolerance are included in the model.



Data Scheduler life cycle

File Transfer life cycle

Performance evaluation: Overhead

		med	90^{th} centile	std dev
Latency	Local	0.77 *ms*	0.81 *ms*	18.68 *ms*
	Eth.	1.25 *ms*	1.45 *ms*	12.97 *ms*
Overhead	Eth.	w/o AD 38.04 *s*	with AD 40.6 *s* (4.6%)	

Table: Latency in milliseconds for life cycle creation and transition publication and overhead measured using BitDew file transfers with and without Active Data.

A single node transfers 1,000 1KB files.
More than 6,000 transitions published.

Supported Systems

- 4 systems
  - BitDew
  - inotify
  - **Globus Online**
  - **iRODS**
- 4 use-cases
  - Storage cache for Amazon S3
  - Distributed data throttling
  - Incremental MapReduce
  - **Data provenance**

Exemple: Data Provenance

Definition

The complete history of data life cycle derivations and operations.

- Assess the quality of data
- Keep track of the origin of data over time
- Specialized Provenance Aware Storage Systems

Exemple: Data Provenance

Definition

The complete history of data life cycle derivations and operations.

- Assess the quality of data
- Keep track of the origin of data over time
- Specialized Provenance Aware Storage Systems
  $\longrightarrow$ What about heterogeneous systems?

Exemple: Data Provenance

Definition

The complete history of data life cycle derivations and operations.

- Assess the quality of data
- Keep track of the origin of data over time
- Specialized Provenance Aware Storage Systems
  $\longrightarrow$ What about heterogeneous systems?

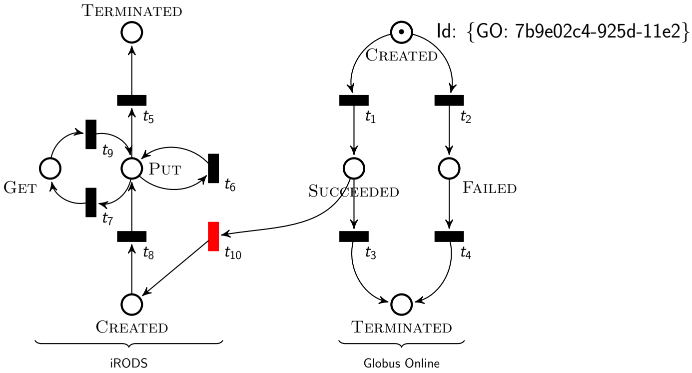
Example with **Globus Online** and **iRODS**

File transfer service

Data store and metadata catalog

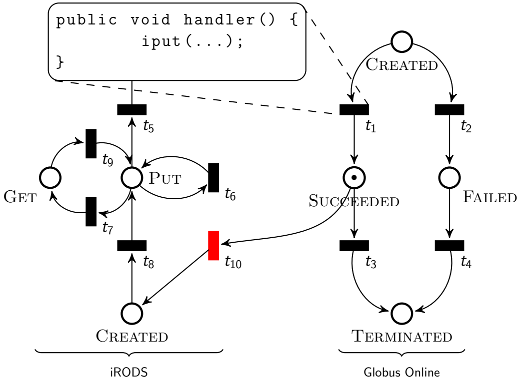
Active Data: principles & features

Data events coming from Globus Online and iRODS



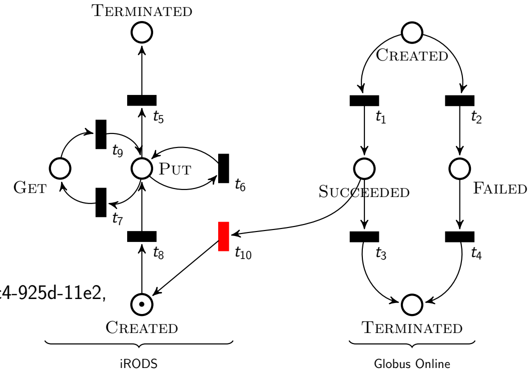
Active Data: principles & features

Data events coming from Globus Online and iRODS



Active Data: principles & features

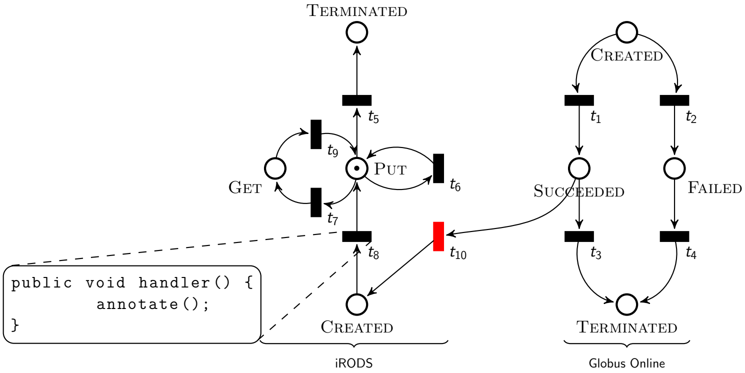
Data events coming from Globus Online and iRODS



Id: {GO: 7b9e02c4-925d-11e2, iRODS: 10032}

Active Data: principles & features

Data events coming from Globus Online and iRODS



Exemple: Data Provenance

```
$ imeta ls -d test/out_test_4628
AVUs defined for dataObj test/out_test_4628:
attribute: GO_FAULTS
value: 0
----
attribute: GO_COMPLETION_TIME
value: 2013-03-21 19:28:41Z
----
attribute: GO_REQUEST_TIME
value: 2013-03-21 19:28:17Z
----
attribute: GO_TASK_ID
value: 7b9e02c4-925d-11e2-97ce-123139404f2e
----
attribute: GO_SOURCE
value: go#ep1/~/test
----
attribute: GO_DESTINATION
value: asimonet#fraise/~/out_test_4628
```

Advantages

- Simple and graphical way to program DLM operations
- Allows to formally verify some properties of data life cycles
- Easy coordination between systems
- Easy to scale
- Easy to debug
- Easy fault tolerance
- Fine-grain interaction with data life cycle

Limitations

- Complexity to reason in terms of life cycle events
- Lack of standard

Related works

- Data-centric parallel programming languages (MapReduce, Dryad, Allpairs, Twister, PigLatin. . .)
- Runtime execution environments for dynamic data : incremental processing (Percolator), parallel stream processing (Nephele, MapReduce Online), workflow (Chimera)
- Event based processing (Mace, libasync, Incontext)
- Data Provenance addresses the issue of representation of data-set derivation (PASS, Open Provenance Model)
- Data Management Software (BitDew, Chirp, MosaStore, Globus Online, DCache, iRODS and many more)

Conclusion

Active Data is. . .

- Data-centric & Event-driven
- System-level data integration

What's next?

- Advanced representation of operations that consume and produce data: represent data derivation
- Data collection abilities
- Distributed implementation of the Publish/Subscribe layer

Thank you!

Questions?