



pomsets

Workflow management for your cloud

Michael Pan

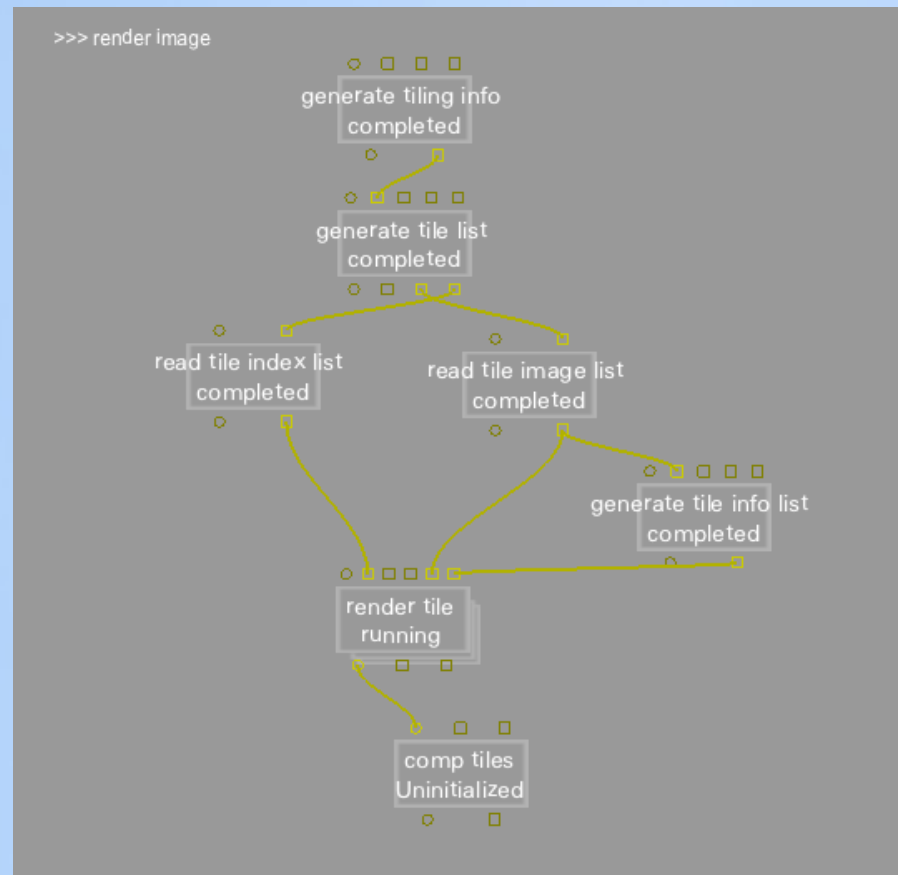
neposity

In the future, the rapidity with which any given discipline advances is likely to depend on how well the community acquires the necessary expertise in database, **workflow management**, visualization, and **cloud computing** technologies.

“Beyond the Data Deluge”, *Science*, Vol. 323. no. 5919, pp. 1297-1298, 2009.

Workflow management is...

the design,
specification,
coordination of
the execution of
tasks and task
dependencies.



Why workflow management + cloud computing?

- Cloud computing provides the ability to scale compute resources with the work that needs to be done
- Better than what has been available, i.e. WFM+grid
- WFM is critical to a successful long-term cloud computing strategy
 - A critical component of the cloud computing software stack
 - Growing recognition of the need for workflow management

Issues with WFM+grid

- Jobs submitted to grids queue up behind jobs of other users, reduces operational efficiencies provided by WFMS
- Heterogeneous comput environments may result in different task results
- Grids are not easily federated, limiting burst computing
- Available only to institutions with the resources to deploy their own grid and implement their own WFMS

Components of a cloud computing software stack

- Virtual machines (VMWare, Xen, Virtuzzo, KVM)
- Dynamic provisioning (Amazon EC2, Eucalyptus)
- Task partitioning (MapReduce, Hadoop, Disco, Sphere)
- Data distribution (GFS, HDFS, Ceph, Sector, MongoDB, CouchDB)
- Unified messaging (Qpid, RabbitMQ, ZeroMQ)
- Workflow management (Azkaban, Kepler, Oozie, Pipeline, Pegasus, Taverna, Triana, pomsets)
- Analytics (Rightscale, Nagios, Ganglia, Graphite)


Growing recognition of the need for workflow management

(screencap 2009-12-04, currently 59 watchers)

Home About Community Training Support Distribution Blog

Hadoop Development Status

Report generated on 2009-12-04 19:43:21 by cron



JIRA Issues Mailing List Contributors

Most Watched Issues

Hadoop uses [JIRA](#) to track bugs and tasks. Anyone can sign up for a JIRA account and watch issues; [sign up here](#) to get involved in the community! The following table shows the most watched open tickets in Hadoop core.

[Download CSV](#)

Name	Type	Status	Component	Watchers
[HADOOP-5303] Oozie, Hadoop Workflow System	New Feature	Patch Available	None	47
[HADOOP-5670] Hadoop configurations should be read from a distributed system	New Feature	Open	conf	25
[HADOOP-3421] Requirements for a Resource Manager for Hadoop	New Feature	Open	None	25
[HADOOP-4487] Security features for Hadoop	New Feature	Open	security	24
[HADOOP-2560] Processing multiple input splits per mapper task	Bug	Open	None	23

Why pomsets?

- Other existing workflow management systems are made for programmers
- Non-programmers in enterprises need an easier way to manage their data-intensive computational workflows

Oozie

```
<action name="myPigjob" type="pig">
  <task name="pig" >
    <inputs>
      <dataset url="s3://traffic/oystercard/london/2009" />
      <dataset url="s3://traffic/anpr/london/2009/" />
      <file path="/users/steve/pig/car-vs-tube.pig" />
    </inputs>
    <option name="limit" value="40000" />
    <option name="pigfile" value="/users/steve/pig/car-vs-tube.pig" />
    <option name="startdate" value="2009-01-01" />
    <option name="enddate" value="2009-02-28" />
    <outputs>
      <dataset url="hdfs://traffic/anpr/london" />
    </outputs>
  </task>
</action>
```

Cascading

Bind multiple sources and sinks to a Flow

```
Pipe headLeft = new Pipe( "headLeft" );
// do something interesting

Pipe headRight = new Pipe( "headRight" );
// do something interesting

// merge the two input streams
Pipe merged = new GroupBy( headLeft, headRight, new Fields( "common" ) );
// ...

// branch the merged stream
Pipe tailLeft = new Pipe( "tailLeft", merged );
// filter out values to the left
tailLeft = new Each( tailLeft, new SomeFilter() );

Pipe tailRight = new Pipe( "tailRight", merged );
// filter out values to the right
tailRight = new Each( tailRight, new SomeFilter() );

// source taps
Tap sourceLeft = new Hfs( new Fields( "some-fields" ), "some/path" );
Tap sourceRight = new Hfs( new Fields( "some-fields" ), "some/path" );

Pipe[] pipesArray = Pipe.pipes( headLeft, headRight );
Tap[] tapsArray = Tap.taps( sourceLeft, sourceRight );

// a convenience function for creating branch names to tap maps
Map<String, Tap> sources = Cascades.tapsMap( pipesArray, tapsArray );

// sink taps
Tap sinkLeft = new Hfs( new Fields( "some-fields" ), "some/path" );
Tap sinkRight = new Hfs( new Fields( "some-fields" ), "some/path" );

pipesArray = Pipe.pipes( tailLeft, tailRight );
tapsArray = Tap.taps( sinkLeft, sinkRight );

// or create the Map manually
Map<String, Tap> sinks = new HashMap<String, Tap>();
sinks.put( tailLeft.getName(), sinkLeft );
sinks.put( tailRight.getName(), sinkRight );

// set property on Flow
FlowConnector flowConnector = new FlowConnector();

Flow flow = flowConnector.connect( "flow-name". sources. sinks. tailLeft. tailRight );
```

Pig

Here's the entirety of Pig Latin code that achieves this:

Load:

```
t1 = LOAD 'texts/alls_well.txt' USING TextLoader() AS (string:chararray);
t1 = FOREACH t1 GENERATE 'alls_well.txt' as fname, string;
t2 = LOAD 'texts/cymbeline.txt' USING TextLoader() as (string:chararray);
t2 = FOREACH t2 GENERATE 'cymbeline.txt' as fname, string;
text = UNION t1, t2;
```

Process:

```
words = FOREACH text GENERATE fname, FLATTEN( TOKENIZE(string) );
-- thanks to reader 'jakeo' for catching the typo in the line below...
word_groups = GROUP words BY $1;
index = FOREACH word_groups {
files = DISTINCT $1.$0;
cnt = COUNT(files);
GENERATE $0, cnt, files;
}.
```

Shell script

```
#!/bin/bash

for i in {1..5}
do
    /bin/wordcount /tmp/pomsets/text$i /tmp/pomsets/count$i
done

/bin/wordcount_reduce -input \  
    /tmp/pomsets/count1 \  
    /tmp/pomsets/count2 \  
    /tmp/pomsets/count3 \  
    /tmp/pomsets/count4 \  
    /tmp/pomsets/count5 \  
-output /tmp/pomsets/count_all
```

pomsets is ...

- A mathematical model- first used in 1985 by Vaughn Pratt- to describe concurrent processes
- An application that implements the mathematical model as the data structures that represent workflow complements, facilitates the design and specification of workflows, and coordinates the execution of workflow tasks on cloud deployments

The mathematical definition

A labelled partial order is a 4 tuple $(V, \Sigma, \preceq, \mu)$ where

- ▶ V is a set of vertices
- ▶ Σ is the alphabet
- ▶ \preceq is the partial order on the vertices
- ▶ μ is the labelling function $\mu: V \rightarrow \Sigma$

A pomset (partially ordered multiset) is a LPO up to isomorphism.

The workflow management system

- 2 components
 - pomsets-core is the backend and provides an API
 - pomsets-gui is the front end and interacts with the user

Features

- Parallel computing
- Data flow
- Flow control
- Workflow reusability
- Compute cloud agnosticism
- Execute environment agnosticism
- Task partitioning
- Shell commands, Hadoop, Python functions, etc
- Intuitive GUI
- Simple API

Demo

How to create the following script in pomsets

```
#!/bin/bash

for i in {1..5}
do
    /bin/wordcount /tmp/pomsets/text$i /tmp/pomsets/count$i
done

/bin/wordcount_reduce -input \  
    /tmp/pomsets/count1 \  
    /tmp/pomsets/count2 \  
    /tmp/pomsets/count3 \  
    /tmp/pomsets/count4 \  
    /tmp/pomsets/count5 \  
-output /tmp/pomsets/count_all
```

Demo



Growing recognition

- nephosity was showcased at Structure 2010 as one of the 11 most promising startups, due to its focus on workflow management in the cloud for non-programmers

nephosity.com
enable the cloud



@nephosity

Michael Pan
mjpan@nephosity.com