About me

• Apache Hadoop Committer, PMC Member, Apache Member
• Engineer at Cloudera working on core Hadoop
• Founder of Apache Whirr
• Author of “Hadoop: The Definitive Guide”
  • [http://hadoopbook.com](http://hadoopbook.com)
Agenda

- What is Whirr?
- How to use Whirr
- How to write a Whirr Service
- Future work
What is Whirr?
Whirr is an easy way to run services in the cloud
Two aspects

▪ Make it easy for service writers to “Whirr-enable” their service

▪ Make it easy for users to consume Whirr services
Whirr in 5 minutes

% curl http://www.apache.org/dist/incubator/whirr/whirr-0.5.0-incubating/whirr-0.5.0-incubating.tar.gz | tar zxf -
% cd whirr-0.5.0-incubating
% ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa_whirr

% bin/whirr launch-cluster \
   --config recipes/zookeeper-ec2.properties \
   --private-key-file ~/.ssh/id_rsa_whirr \
   --identity=$AWS_ACCESS_KEY_ID \
   --credential=$AWS_SECRET_ACCESS_KEY

% echo "ruok" | nc $(awk '{print $3}' ~/.whirr/zookeeper/instances | head -1) 2181; echo
1. Install

% curl http://www.apache.org/dist/incubator/whirr/whirr-0.5.0-incubating/whirr-0.5.0-incubating.tar.gz | tar zxf -
% cd whirr-0.5.0-incubating
% ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa_whirr

% bin/whirr launch-cluster \
  --config recipes/zookeeper-ec2.properties \
  --private-key-file ~/.ssh/id_rsa_whirr \
  --identity=$AWS_ACCESS_KEY_ID \
  --credential=$AWS_SECRET_ACCESS_KEY

% echo "ruok" | nc $(awk '{print $3}' ~/.whirr/zookeeper/instances | head -1) 2181; echo
2. Run

% curl http://www.apache.org/dist/incubator/whirr/whirr-0.5.0-incubating/whirr-0.5.0-incubating.tar.gz | tar zxf -
% cd whirr-0.5.0-incubating
% ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa_whirr

% bin/whirr launch-cluster \ 
   --config recipes/zookeeper-ec2.properties \ 
   --private-key-file ~/.ssh/id_rsa_whirr \ 
   --identity=$AWS_ACCESS_KEY_ID \ 
   --credential=$AWS_SECRET_ACCESS_KEY

% echo "ruok" | nc $(awk '{print $3}' ~/.whirr/zookeeper/instances | head -1) 2181; echo
3. Use bit.ly/whirr5

% curl http://www.apache.org/dist/incubator/whirr/whirr-0.5.0-incubating/whirr-0.5.0-incubating.tar.gz | tar zxf -
% cd whirr-0.5.0-incubating
% ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa_whirr

% bin/whirr launch-cluster \
   --config recipes/zookeeper-ec2.properties \
   --private-key-file ~/.ssh/id_rsa_whirr \
   --identity=$AWS_ACCESS_KEY_ID \
   --credential=$AWS_SECRET_ACCESS_KEY

% echo "ruok" | nc $(awk '{print $3}掸 /./whirr/zookeeper/instances | head -1) 2181; echo

imok
Configuration

• zookeeper-ec2.properties:

  whirr.cluster-name= zookeeper
  whirr.instance-templates= 3 zookeeper
  whirr.provider= aws-ec2
  whirr.identity= ${env: AWS_ACCESS_KEY_ID}
  whirr.credential= ${env: AWS_SECRET_ACCESS_KEY}
What did it do?

controller
machine

Whirr
JVM

Cluster
state

1. Bootstrap
   provision (jclouds)
   install (ssh)

2. Configure
   configure (ssh)
   start (ssh)

ZooKeeper

ZooKeeper

ZooKeeper
The Big Picture

Whirr CLI

Whirr services
Hadoop  ZooKeeper  HBase  ...

Whirr core

jclouds
EC2  Cloud Servers  vcloud  BYON  ...

Friday, July 29, 2011
jclouds is awesome

- ComputeService API for managing machines
  - Uniform API across ~20 providers
- BlobStore API for using key-value stores
  - Uniform API across ~10 providers
- Optionally use provider-specific APIs to use non-portable features
  - E.g. EC2 spot pricing
- Emphasis on testing and performance
- Vibrant, responsive community
The Big Picture

Whirr CLI

Whirr services
- Hadoop
- ZooKeeper
- HBase
- ...

Whirr core

jclouds
- EC2
- Cloud Servers
- vcloud
- BYON
- ...

Friday, July 29, 2011
The Whirr Community

- Apache Whirr is currently undergoing Incubation at the Apache Software Foundation
- Over 1 year old
- 5 releases
- People: 10 committers (6 orgs), more contributors and users
- The Whirr community shares recipes
  - Cloud best practice (e.g. good images, hardware types)
  - Service configuration
4. Don’t forget to shutdown!  

% bin/whirr destroy-cluster --config recipes/zookeeper-ec2.properties
How to use Whirr
Using Whirr from Java

```java
Configuration conf = new PropertiesConfiguration("recipes/zookeeper-ec2.properties"); //1
ClusterSpec spec = new ClusterSpec(conf); //2
ClusterController cc = new ClusterController(); //3
Cluster cluster = cc.launchCluster(spec); //4

String hosts = ZooKeeperCluster.getHosts(cluster); //5
ZooKeeper zookeeper = new ZooKeeper(hosts, ...); //6
// interact with ZooKeeper cluster

cc.destroyCluster(spec); //7
```
A Lifecycle API

- Very simple API
- ClusterController
  - `Cluster launchCluster(ClusterSpec spec)`
  - `void destroyCluster(ClusterSpec spec)`
  - `Set<Instance> getInstances(ClusterSpec spec)`

- Whirr is not dependent on service libraries (e.g. ZooKeeper)
- Version independent
Whirr is very customizable

- Version
  - Specify the version (e.g. whirr.hadoop.version)
  - Or the tarball to install (e.g. whirr.hadoop.tarball.url)

- Dev workflow:
  - Build tarball – e.g. Hadoop with a patch you want to test
  - Start a cluster that uses this tarball specified as a file:// URI
  - Whirr will push tarball to a blob store and then download onto cloud instances
Customizing services

• Configuration
  • Set service properties
    • E.g. `hadoop-common.fs.trash.interval=1440`
      • Sets `fs.trash.interval` in the Hadoop cluster configuration
      • Whirr will generate the service configuration file for the cluster

• Customize nodes
  • E.g. install extra software on nodes simply by editing scripts
Characteristics of Whirr Clusters

- Short lived clusters with a small number of users
- Testing, manual or automated (e.g. Jenkins)
- Evaluation of services
- Ad hoc data exploration
  - Example: data POC
    - Load data from e.g. S3 into temporary cluster (Hadoop, HBase) for analysis
- Reproducibility
  - A way to share analysis. Can share datasets easily already, but Whirr makes it easy to reproduce results.
Whirr Use Cases

▪ Cloudera
  ▪ Provides Whirr in CDH to make it easy to try out Hadoop

▪ Omixon
  ▪ Uses Whirr to run human exome analysis
  ▪ Regular job uses 10 machines
  ▪ 80 gigabases exome pipeline runs in 4 hours

▪ Outerthought
  ▪ Will use Whirr to do Lily cluster installs
  ▪ Lily combines HBase and Solr to provide large-scale storage with indexing and search

▪ https://cwiki.apache.org/confluence/display/WHIRR/Powered+By
Steps in writing a Whirr service

- 1. Identify service roles
- 2. Write a ClusterActionHandler for each role
- 3. Write scripts that run on cloud nodes
- 4. Package and install
- 5. Run
1. Identify service roles

- Flume, a service for collecting and moving large amounts of data
- Flume Master
  - The head node, for coordination
  - Whirr role name: flumedemo-master
- Flume Node
  - Runs agents (generate logs) or collectors (aggregate logs)
  - Whirr role name: flumedemo-node

https://github.com/cloudera/flume
2. Write a ClusterActionHandler for each role

```java
public class FlumeNodeHandler extends ClusterActionHandlerSupport {

    public static final String ROLE = "flumemon-node";

    @Override
    public String getRole() { return ROLE; }

    @Override
    protected void beforeBootstrap(ClusterActionEvent event)
        throws IOException, InterruptedException {
        addStatement(event, call("install_java"));
        addStatement(event, call("install_flumemon"));
    }

    // more ...
}
```
public class FlumeNodeHandler extends ClusterActionHandlerSupport {

    // continued ...

@override
protected void beforeConfigure(ClusterActionEvent event) throws IOException, InterruptedException {
    // firewall ingress authorization omitted

    Cluster cluster = event.getCluster();
    Instance master = cluster.getInstanceMatching(role(FlumeMasterHandler.ROLE));
    String masterAddress = master.getPrivateAddress().getHostAddress();
    addStatement(event, call("configure_flumedemo_node", masterAddress));
}
}
3. Write scripts that run on cloud nodes

- install_java is built in
- Other functions are specified in individual files

```bash
function install_flum demos() {
    curl -O http://cloud.github.com/downloads/cloudera/flume/flume-0.9.3.tar.gz
    tar -C /usr/local/ -zxf flume-0.9.3.tar.gz
    echo "export FLUME_CONF_DIR=/usr/local/flume-0.9.3/conf" >> /etc/profile
}
```
You can run as many scripts as you want

- This script takes an argument to specify the master

```bash
function configure_flumedemo_node() {
    MASTER_HOST=$1
    cat > /usr/local/flume-0.9.3/conf/flume-site.xml <<EOF
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>flume.master.servers</name>
        <value>$MASTER_HOST</value>
    </property>
</configuration>
EOF
    FLUME_CONF_DIR=/usr/local/flume-0.9.3/conf \
    nohup /usr/local/flume-0.9.3/bin/flume node > /var/log/flume.log 2>&1 &
}
```
4. Package and install

- Each service is a self-contained JAR:
  
  functions/configure_flumemo示_master.sh  
  functions/configure_flumemo示_node.sh  
  functions/install_flumemo示.sh  
  META-INF/services/org.apache.whirr.service.ClusterActionHandler  
  org/apache/whirr/service/example/FlumeMasterHandler.class  
  org/apache/whirr/service/example/FlumeNodeHandler.class

- Discovered using java.util.ServiceLoader facility
  
  - META-INF/services/org.apache.whirr.service.ClusterActionHandler:
    
    org.apache.whirr.service.example.FlumeMasterHandler  
    org.apache.whirr.service.example.FlumeNodeHandler

- Place JAR in Whirr’s lib directory
5. Run

▪ Create a cluster spec file

    whirr.cluster-name=flumedemo
    whirr.instance-templates=1 flumedemo-master,1 flumedemo-node
    whirr.provider=aws-ec2
    whirr.identity=${env:AWS_ACCESS_KEY_ID}
    whirr.credential=${env:AWS_SECRET_ACCESS_KEY}

▪ Then launch from the CLI

    % whirr launch-cluster --config flumedemo.properties

▪ or Java

    Configuration conf = new PropertiesConfiguration("flumedemo.properties");
    ClusterSpec spec = new ClusterSpec(conf);
    ClusterController cc = new ClusterController();
    Cluster cluster = cc.launchCluster(spec);
    // interact with Flume cluster
    cc.destroyCluster(spec);
Orchestration

- Instance templates are acted on independently in parallel
- Bootstrap phase
  - start 1 instance for the `flumedemo-master` role and run its bootstrap script
  - start 1 instance for the `flumedemo-node` role and run its bootstrap script
- Configure phase
  - run the configure script on the `flumedemo-master` instance
  - run the configure script on the `flumedemo-node` instance
- Note there is a barrier between the two phases, so nodes can get the master address in the configure phase
Future Work
Challenges

▪ Complexity
▪ Degrees of freedom

\[
\#\text{clouds} \times \#\text{OS} \times \#\text{hardware} \times \\
\#\text{images} \times \#\text{locations} \times \\
\#\text{services} \times \#\text{configs} = \text{a big number!}
\]

▪ Known good configurations, recipes
▪ Regular automated testing
▪ Move common patterns into core
▪ Debugging – what to do when the service hasn’t come up?
▪ Logs
What’s next?

▪ Add more services
  ▪ Use Bigtop – packaging and testing for the Hadoop ecosystem
    ▪ http://incubator.apache.org/projects/bigtop.html
▪ Support more cloud providers
▪ Support for configuration management systems like Puppet and Chef

▪ https://cwiki.apache.org/confluence/display/WHIRR/RoadMap
Questions

▪ Find out more at
  ▪ http://incubator.apache.org/whirr
  ▪ https://github.com/tomwhite/whirr-service-example
▪ IRC: #whirr on freenode
▪ Twitter: @tom_e_white