IdeasOnLdapConfiguration

This is for HADOOP:5670.

First, a bit about LDAP (extremely simplified).

All objects in LDAP are defined by one or more object classes. These object classes define the attributes that a given object can utilize. The attributes in turn have definitions that determine what kinds of values can be used. This includes things as string, integer, etc, but also whether or not the attribute can hold more than one value. Object classes, attributes, and values are all defined in a schema definition.

One key feature around LDAP is the ability to search objects using a simple, RPN-style system. Let's say we have an object class that has this definition:

```
objectclass: node
hostname: string
domain: string
```

and in our LDAP server, we have placed the following objects:

```
hostname=myhost1
objectclass=node
domain=example.com

hostname=myhost2
objectclass=node
domain=example.com
```

We can now do an LDAP search with (&(objectclass=node)(hostname=myhost1)) to find the 'myhost1' object. Similarly, we can (&(objectless)(domain=example.com)) to find both myhost1 and myhost2 objects.

Let's apply these ideas to Hadoop. Here are some rough objectclasses that we can use for demonstration purposes:

```
generic properties: hadoopGlobalConfig
hadoop.tmp.dir: string
fs.default.name: string
dfs.block.size: integer
dfs.replication: integer
clusterName: string

For datanodes: hadoopDataNode
hostname:  multi-string
dfs.data.dir: multi-string
dfs.datanode.du.reserved: integer
commonname: string

hadoopTaskTracker
commonname: string
hostname: multi-string
mapred.job.tracker: string
mapred.local.dir: multi-string
mapred.tasktracker.map.tasks.maximum: integer
mapred.tasktracker.reduce.tasks.maximum: integer

hadoopJobTracker
hostname:
mapred.reduce.tasks: integer
mapred.reduce.slowstart.completed.maps: numeric
mapred.queue.names: multi-string
mapred.jobtracker.taskScheduler: string
mapred.system.dir: string

For the namenode: hadoopNameNode
commonname: string
dfs.http.address: string
hostname: string
dfs.name.dir: multi-string
```

Let's define a simple grid:

```
clusterName=red
objectclass=hadoopGlobalConfig
hadoop.tmp.dir=/tmp
fs.default.name: hdfs://namenode:9000/
dfs.block.size: 128
dfs.replication: 3

commonname=master,cluster=red
objectclass=hadoopNameNode,hadoopJobTracker
dfs.http.address: http://masternode:50070/
hostname: masternode
dfs.name.dir: /nn1,/nn2
mapred.reduce.tasks: 1
mapred.reduce.slowstart.completed.maps: .55
mapred.queue.names: big,small
mapred.jobtracker.taskScheduler: capacity
mapred.system.dir: /system/mapred

commonname=simplecomputenode,cluster=red
objectclass=hadoopDataNode,hadoopTaskTracker
hostname:  node1,node2,node3
dfs.data.dir: /hdfs1, /hdfs2, /hdfs3
dfs.datanode.du.reserved: 10
mapred.job.tracker: commonname=jobtracker,cluster=red
mapred.local.dir: /mr1,/mr2,/mr3
mapred.tasktracker.map.tasks.maximum: 4
mapred.tasktracker.reduce.tasks.maximum: 4
```

Let's say we fire up node1. The local config would say what ldap server, necessary creds to talk to the ldap server, etc. It might also say that it is part of the red cluster in order to speed up the startup. From there, it would do the following:

Get all the global config for the red cluster: search scope: (&(objectclass=hadoopGlobalConfig)(clusterName=red)). We now know hadoop.tmp.dir, fs. default.name, etc.

Are we a namenode? (&(objectclass=hadoopNameNode)(hostname=node1)). Empty. Drats!

Are we a datanode? (&(objectclass=hadoopDataNode)(commonname=node1)). We got an object back! Grab that info and can now start up the datanode process.

Are we a jobtracker? (&(objectclass=hadoopJobTracker)(hostname=node1)). Empty. Drats!

Are we a tasktracker? (&(objectclass=hadoopTaskTracker)(hostname=node1)): We got an object back! Fire up the task tracker with that object's info.

Let's do a more complex example. What happens if we have more than one type of compute node? We just have multiple definitions of our compute nodes:

```
commonname=computenode1,cluster=red
objectclass=hadoopDataNode,hadoopTaskTracker
hostname:  node1,node2,node3
dfs.data.dir: /hdfs1, /hdfs2, /hdfs3
dfs.datanode.du.reserved: 10
mapred.job.tracker: commonname=jobtracker,cluster=red
mapred.local.dir: /mr1,/mr2,/mr3
mapred.tasktracker.map.tasks.maximum: 4
mapred.tasktracker.reduce.tasks.maximum: 4

commonname=computenode2,cluster=red
objectclass=hadoopDataNode,hadoopTaskTracker
hostname:  node4,node5,node6
dfs.data.dir: /hdfs1, /hdfs2, /hdfs3, /hdfs4
dfs.datanode.du.reserved: 10
mapred.job.tracker: commonname=jobtracker,cluster=red
mapred.local.dir: /mr1,/mr2,/mr3,/mr4
mapred.tasktracker.map.tasks.maximum: 8
mapred.tasktracker.reduce.tasks.maximum: 4
```

What if we want more than one job tracker talking to the same HDFS? LDAP makes defining this easy:

```
commonname=masternn,cluster=red
objectclass=hadoopNameNode
dfs.http.address: http://masternn:50070/
hostname: masternn
dfs.name.dir: /nn1,/nn2

commonname=jt1,cluster=red
mapred.reduce.tasks: 1
mapred.reduce.slowstart.completed.maps: .55
mapred.queue.names: big,small
mapred.jobtracker.taskScheduler: capacity
mapred.system.dir: /system/mapred
hostname=jt1

commonname=jt2,cluster=red
mapred.reduce.tasks: 1
mapred.reduce.slowstart.completed.maps: .55
mapred.queue.names: etl
mapred.jobtracker.taskScheduler: capacity
mapred.system.dir: /system/mapred
hostname=jt2

commonname=computenode1,cluster=red
objectclass=hadoopDataNode,hadoopTaskTracker
hostname:  node1,node2,node3
dfs.data.dir: /hdfs1, /hdfs2, /hdfs3
dfs.datanode.du.reserved: 10
mapred.job.tracker: commonname=jt1,cluster=red
mapred.local.dir: /mr1,/mr2,/mr3
mapred.tasktracker.map.tasks.maximum: 4
mapred.tasktracker.reduce.tasks.maximum: 4

commonname=computenode2,cluster=red
objectclass=hadoopDataNode,hadoopTaskTracker
hostname:  node4,node5,node6
dfs.data.dir: /hdfs1, /hdfs2, /hdfs3, /hdfs4
dfs.datanode.du.reserved: 10
mapred.job.tracker: commonname=jt2,cluster=red
mapred.local.dir: /mr1,/mr2,/mr3,/mr4
mapred.tasktracker.map.tasks.maximum: 8
mapred.tasktracker.reduce.tasks.maximum: 4
```

This is important when you consider that small-medium sized grids are likely to have a mix of nodes. For example, some nodes may have 8 cores with four disks and some nodes may have 6 cores with eight disks. If they are part of the same cluster, they will need different mapred-site.xml settings in order to maximize the hardware purchase.

From the client side, this is a huge win. We can do things like:

```
$ hadoop listgrids
red
green
```

In LDAP terms, this would be fetching the (objectclass=hadoopGlobalConfig) and reporting all clusternames.

I can also submit a job without knowing any particulars or having a bunch of config files to manage:

```
$ hadop job -grid red -jt jt1 -jar ....
```

Because we have access too all grid definitions, we could also do:

```
$ hadoop distcp red:/my/dir  green:/my/dir
```

With distcp able to get the necessary grid config from LDAP with no (real) local config necessary.