Building and Administering Hadoop Clusters

21 April 2011
Jordan Boyd-Graber

UNIVERSITY OF MARYLAND 18 56

COLLEGE OF
INFORMATION
STUDIES

Administrivia

- Homework 5 graded
- Homework 6 due soon
- Keep working on projects!
- Final next week (will take better of midterm of final)
- Will have food for final class, May 5 (RSVP)
- Project writeup due May 10

Roadmap

- Choosing hardware / platform
- Getting a single node up and running
- Managing a running cluster
  - Caches, Buffers, and Backups
  - Scheduling Policies
- Adding nodes

Caveats and Context

- Why talk about this now?
- Even if you never have to worry about it, it helps you understand the underlying process
- I am not an expert in running Hadoop clusters
- However ...
  - Have seen multiple clusters in operation
  - Involved in setting up Maryland's
  - Suggestions culled from multiple sources
  - Have run these tips by people who do admin (but too shy / lazy to talk to you)
- Your mileage may vary ... be sure to vet tweaks

What Machines to Buy

- Get beefy consumer-grade machines
- Get components that you can replace for the next 4-8 years
- If you want homogenous hardware, buy expensive now, and have costs descend as you scale out over time
- UMIACS Bespin cluster:
  - Data nodes: HCGI/Ingram-Micro SuperMicro 2U quad-server enclosure with each server equipped with 2 quad-core 2.4Ghz Opteron Processors, 24GB of memory, and three 2TB SATA Drives.
  - Name nodes: PowerEdge R610 with dual 2.66 Ghz processors, 48GB of memory (6x4GB) , two mirrored 500GB 7200 rpm 2.5inch sata drives, and redundant power supplies with an idrac enterprise.

Do you even want to buy machines?

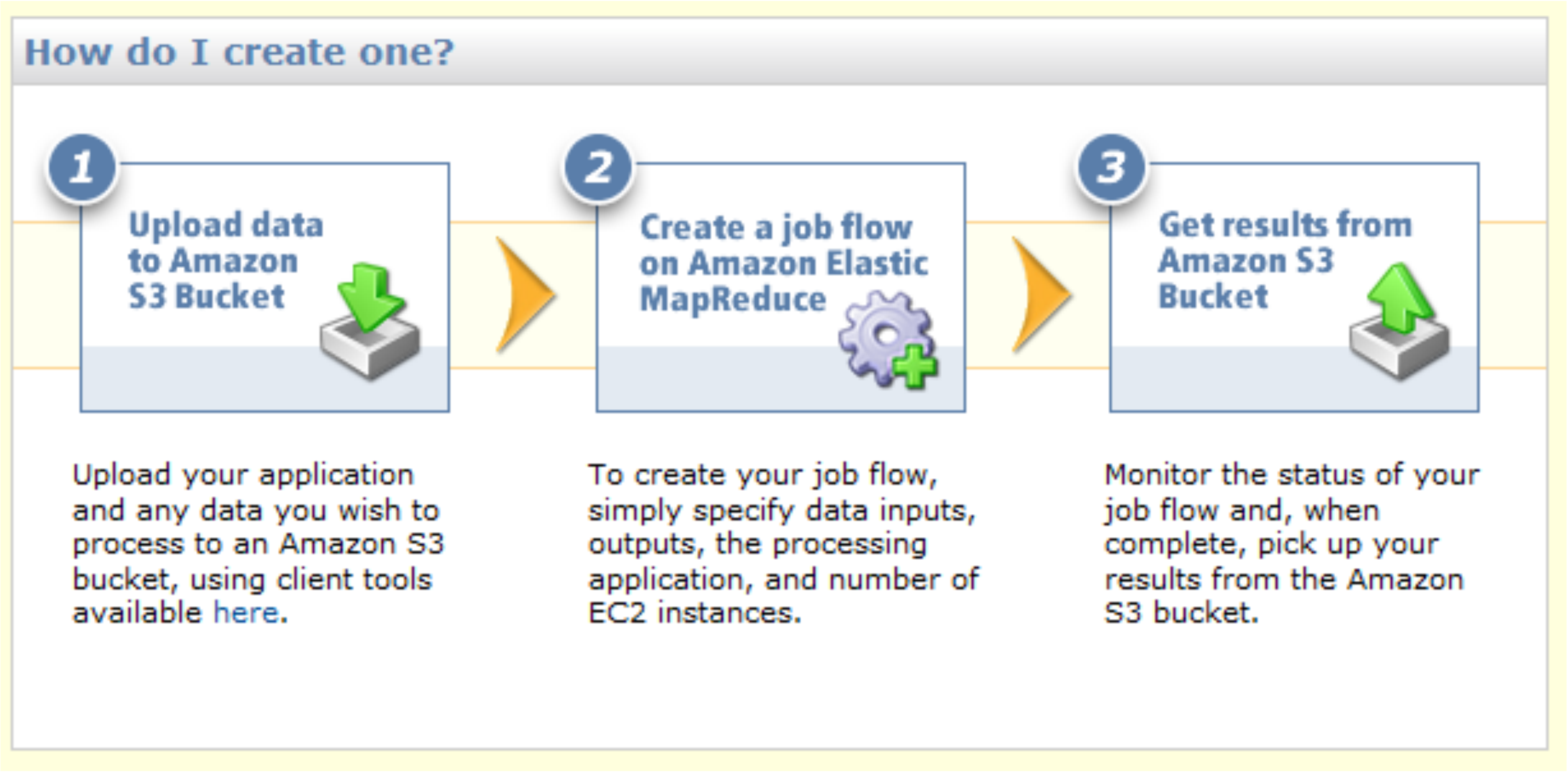
- Amazon Elastic Compute Cloud (Amazon EC2)
- Part of Amazon Web Services (AWS)
- Rent machines for $0.10 / machine hour to $2 / machine hour (depending on CPU / memory)
- Who's using it
  - Autodesk, Washington Post, Reddit
  - Foursquare, Quora, Amazon
- Pros
  - Don't pay for support, electricity
  - Seamless "upgrades"
- Cost
  - Not as cost-effective as running your own cluster 24/7
    - Who does?
  - Less control

Creating and Using a Hadoop Cluster on EC2

- Install Hadoop on a local machine
- Edit *hadoop/src/contrib/ec2/bin/hadoop-ec2-env.sh*□
    - Add AWS account, key
    - Size of machines
    - Architecture
- Hadoop installation provides a script to create cluster
    - bin/hadoop-ec2 launch-cluster test-cluster 2
    - Starts running a TaskTracker, command returns IP
- Can then either log in
- Or run remotely (just like we're doing)
    - Caution, IO is metered (cent per minute)

Do you even want to bother with virtual machines?

- Amazon offers "Elastic Map Reduce"



How do I create one?

1 Upload data to Amazon S3 Bucket

Upload your application and any data you wish to process to an Amazon S3 bucket, using client tools available here.

2 Create a job flow on Amazon Elastic MapReduce

To create your job flow, simply specify data inputs, outputs, the processing application, and number of EC2 instances.

3 Get results from Amazon S3 Bucket

Monitor the status of your job flow and, when complete, pick up your results from the Amazon S3 bucket.

Elastic MapReduce

- Uses S3 for Input and Output
- Very little configuration (web-based)
- Can use most of the techniques discussed in class
    - Streaming
    - Custom jar files
    - Chaining jobs
- Cannot use
    - Local data
    - Hadoop pipes
- API or CLI for automation of creating environments / jobs

Complications of Using AWS

- There are outages (beyond your control)
  - E.g. today (April 21, 2011), Reddit, Foursquare, and Quora were down
- While there are SLAs, it's only a refund of what you've paid
- What's the answer?
  - As before, it's almost always redundancy
- Amazon offers four zones
  - US-East (Norcal), US-West (Virginia), Europe (Ireland), Asia (Singapore)
  - Hardware relatively independent across zones
  - Multiple instances increase probability continuity, cost
  - What about software?

No, I really want to build my own

How to put together a new cluster

- Installing software
- Letting computers talk to each other
- Configuring the network
- Setting up storage
- Changing options
-

Installing Software

- Do it yourself
    - Java
    - Hadoop
    - Anything else you need ...
- Use Cloudera
    - Maintains internally consistent packages
    - Play well together
    - Provides
        - Packages
            - Different
              for namenode, datanode, secondarynamenode, jobtracker, tasktracker
        - Virtual Machine Images
        - Whirr (image + setup) for use on EC2

SSH Key Distribution

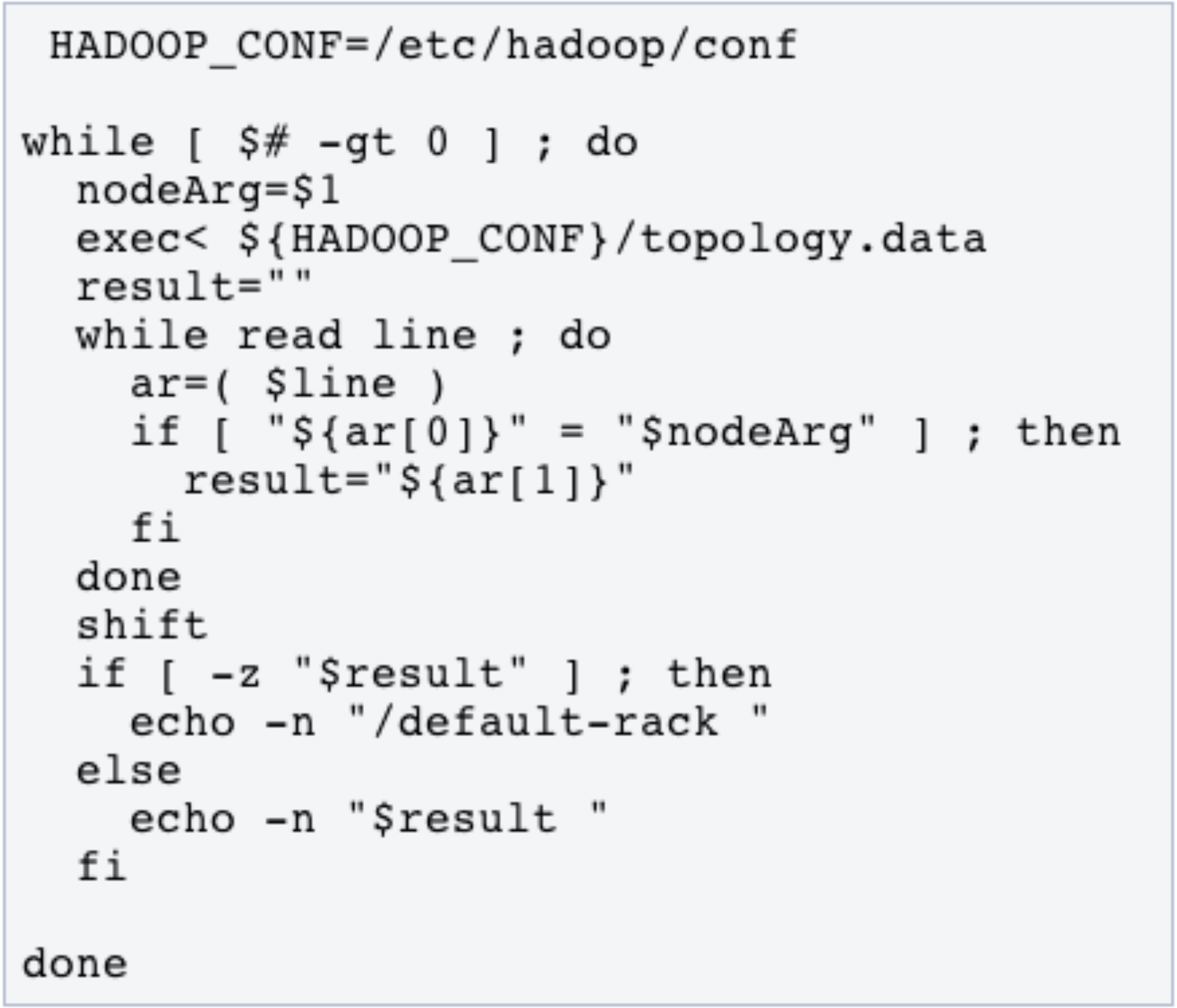
- NameNode and JobTracker must be able to connect to all slave machines (e.g. to start up processes when the cluster starts)
- SSH works on private and public keys
  - Keep private key
  - Distribute public key to the systems you connect to
- Typically done with a script on NameNode and JobTracker that copies public key to many computers
- Do this with "hadoop" user

Specifying Network Topology

- Default configuration puts nodes on the same rack
- For small clusters, this is fine
- Large clusters have more complicated topology
  - Throughput much larger within a rack
  - Tasks will complete faster if jobs are localized to racks
- Goes beyond racks
  - switch, data unit, building, datacenter

Configuring Topology

- The parameter **topology.script.file.name** should point to a script that takes IP addresses or host names and returns the rack location
- You can also do this in Java



```
 HADOOP_CONF=/etc/hadoop/conf

while [ $# -gt 0 ] ; do
  nodeArg=$1
  exec< ${HADOOP_CONF}/topology.data
  result=""
  while read line ; do
    ar=( $line )
    if [ "${ar[0]}" = "$nodeArg" ] ; then
      result="${ar[1]}"
    fi
  done
  shift
  if [ -z "$result" ] ; then
    echo -n "/default-rack "
  else
    echo -n "$result "
  fi

done
```

hadoopdata1.ec.com /dc1/rack1
hadoopdata1 /dc1/rack1
10.1.1.1 /dc1/rack1

Setting up HDFS

- NameNode - Hold metadata for the blocks of data on cluster
- Secondary NameNode - Merges EditList with FsImage
  - Identical memory requirement as NameNode
  - Reconciles edits
  - Not (just) a backup (changes in 0.21)
- Default
  - Nodes are identical
  - EditList is reconciled only on initialization
- NameNode often is the weakest link
  - Good idea to have separate machine, less strain on NameNode
- User-level Trash (not on by default)

Making NameNodes Resilient

- Save NameNode information on multiple hard drives
- Also save NameNode information on NFS (metadata)
- What if NameNode fails?
  - If it's just a HD, replace the disk and continue
  - If the metadata are backed up, then any machine with access to the data can take over
  - Hadoop 0.21 is moving toward hot-swappable NameNodes

Using a Secondary NameNode

- Adding it to the network
  - Add its entry to the *masters*

Update **dfs.http.address** so it knows where to get edits

- What if the NameNode fails?
  - Change the IP address of secondary NameNode to that of old NameNode
    - Cannot just be host, as DNS is cached
  - Remove its entry from masters, add new secondary
  - Start the NameNode on what was the secondary

What does a DataNode look like?

```
${dfs.data.dir}
/current/VERSION
/blk_<id_1>
/blk_<id_1>.meta
/blk_<id_2>
/blk_<id_2>.meta
/...
/blk_<id_64>
/blk_<id_64>.meta
/subdir0/
/subdir1/
/...
/subdir63/
```

- Unlike NameNode **dfs.data.dir** is not replicated (RR)
- meta file contains version information and checksums
- subdirs don't correspond to structure in HDFS; prevent single directory from having too many files (**dfs.datanode.numblocks**)

Getting Ready to Run

- Create a hadoop user that own appropriate directories
  - E.g. temporary processing files
  - DataNode blocks
- Distribute configuration files
- Decide which nodes are going to take on which roles
  - masters - list of secondary name nodes
  - slaves - data nodes
- Run start-dfs.sh on the NameNode (SSH keys)
  - Starts all of the data nodes
  - Starts the SecondaryNameNode
  - Enters safe mode
- Run start-mapred.sh on the JobTracker
  - Starts TaskTracker on all of the slave nodes
  - Starts JobTracker on current node

Options

- Live in the conf directory
  - core-site.xml, mapred-site.xml, hdfs-site.xml
- Written as

```
<property> <name>dfs.client.buffer.dir</name>
<value>/tmp/hadoop/dfs/client</value> <final>true</final>
</property>
```

- Default options
  - designed to be idiotproof
  - somewhat optimized for standalone mode
  - won't fail miserably for larger clusters

Map Options

- **mapred.local.dir** (/tmp/) - Where spills are written
- **min.num.spills.for.combine** (3) - When a combiner is called
- **io.sort.mb** (100) - Buffer used in sorting map output
- **io.sort.spill.percent** (0.8) - How much of the memory needs to be used before spilling to disk
- **tasktracker.http.threads** (40) - How many threads copy data to reducer

MapReduce Options

- **mapred.reduce.max.attempts** (2) - Number of times to try a job before declaring it failed
- **mapred.max.{map|reduce}.failures.percent** (0) - How many failures are possible.
- **mapred.task.timeout** (10 min) - How long between progress before declaring failure.□
  - Task must give output, update counter, or change status within this amount of time
- **mapred.job.reduce.input.buffer.percen**t (0)
  - How much reducer memory is used to buffer input
  - Increase if reduce jobs are light on memory
- **mapred.reduce.copy.backoff** (300 s) - How long to wait on a mapper's input

Changes to Default Options

dfs.name.dir, dfs.data.dir

- Stores where HDFS metadata and blocks are stored
- Defaults to /tmp
  - Why is this a bad idea?
- Suggested change:
  - hadoop home directory (e.g. /home/hadoop/name)

mapred.system.dir

- Stores Hadoop system files
- Defaults to /tmp
- Change to /home/hadoop/system

Changes to Default Options

mapred.tasktracker.{map,reduce}.tasks.maximum

- Number of taks that can run on a single TaskTracker
- Defaults to 4
- Suggested change:
  - If tasks are IO bound, have twice the number of cores available

dfs.datanode.du.reserved

- Minimum amount of free space on DataNode
- Default is 0
- Stopes block writing when threshold is crossed
- Change to 1GB to improve stability

Changes to Default Options

mapred.reduce.tasks

- Number of default reduce tasks per job (of course, configurable per-job)
- Suggested change:
  - 0.8 * maximum number available
  - 1.5 * maximum number available
- Why might these be better ideas?

Cluster's Running ... Now What?

- Addressing common problems
- Improving scheduling
- Monitoring performance
- Adding new nodes

Changes in Response to Problems

- *Big data transferring slowly*:
  - mapred.reduce.parallel.copies - number of threads used to copy from mapper (default 5)
  - mapred.compress.map.output - are spills compressed (default false)
    - Increases CPU overhead per mapper but leads to faster transfer.
- *Long object initialization:* mapred.job.reuse.jvm.num.tasks - reuse the JVM more than once (default 1)
- *Sorts are taking too long:* increase io.sort.factor to a larger number (default 10) so that more spills can be merged at once

Scheduling Jobs

- FIFO
  - Default behavior
  - Early users can monopolize cluster
- FairScheduler
  - Users placed into pools
  - Each pool should get an equal share of resources
  - If resources are unequal for too long, preempt offending jobs
- CapacityScheduler
  - Slices cluster in the queues
  - Jobs are submitted to queues, which maintain FIFO scheduling

fsck and rebalance

- Like the Linux command, checks health of file system
    - Unlike the Linux command, doesn't fix them
- Reports replications
- Can also list where blocks are located for a file
- What to do when unbalanced?
    - Wait and let things sort themselves out
    - Run bin/start-balancer.sh
    - Restart HDFS

Adding New Nodes

- Simple version: Just point nodes at correct JobTracker and NameNode, start daemon
  - Security issue
- Better idea: explicitly specify hosts in dfs.hosts and mapred.hosts located on NameNode and JobTracker
- Is your cluster now good to go?

Removing Nodes

- Could just unplug ...
- Add the node to to *dfs.hosts.excludes* and *mapred.hosts.excludes*
- Jobs will not run
- Blocks will not count toward replication
- Run

```
bin/hadoop dfsadmin -refreshNodes
```

- Will begin to move data off nodes

Ongoing Activities

- Monitor health of cluster (e.g. Ganglia)
- Set up alerts to warn of impending issues
- If there are "bread and butter" applications, regularly benchmark them
- Adjust parameters as average use cases emerge
- Create infrastructure for changing and deploying new configurations

Recap

- Options for running your code on a scalable platform
  - Not rolling your own is often the better option
- Details of a real installation
  - Data storage
  - Network connectivity
  - Scheduling
  - Adding and removing nodes
- Messy details, but this is the glue that holds the web together