Hadoop Cluster Management

http://hadoop.apache.org/

Marco Nicosia
marco@escape.org

Marco Nicosia

First computer job in 1989 working on SimCity

Over 15 years of systems administration experience

Started in systems administration in 1991

- UC Berkeley's Open Computing Facility
- Stretched facility from 3,000 to 10,000 drop-in users
- Deployed the first public-access on-campus web server, hosting over 170 user group websites

First startup in 1996

- GNN, one of the first web magazines to make money selling advertising paired with editorial-generated content

Production Engineering: Inktomi, Walmart.com

Most recently, built the Grid Ops team for Yahoo!

Building a Hadoop Service

What we really do is "Utility" computing

- Nobody knows what "utility computing" is, but everyone has heard of “grid computing”
  - Grid computing implies sharing across resources owned by multiple, independent organizations
  - Utility computing implies sharing one owner’s resources by multiple independent customers

Ultimate goal is to provide shared compute and storage resources

- Instead of going to hardware committee to provision balkanized resources, a project allocates a part of its budget for use on the cluster
- Pay as you go
  - Users only use 100 computers for 15 minutes of compute time, instead of buying 100 computers that are captive 24x7

What is a Grid Service?

Thousands of commodity computers using basic network hardware

- It's hard to program for many machines

Clustering and sharing software

- Hadoop, (Hadoop On Demand, Torque/Maui)
  - Hadoop abstracts both storage and program execution

Terabytes of data

- It's a challenge to load data from many sources

An attached development environment

- A clean, well lighted place to interact with your cluster

User support / Solutions

- Now we're cooking with gas!

Compute Model: Map-Reduce

Programming model fits "Big Data" problems well

- Functional programming means no side effects
- Large data sets can be split up into arbitrary units of work
- Units of work can be repeated (or even run in competition)
- Inspired by Google and the 1970s LISP Map-Reduce programming paradigm
- Easy access to the programmer
- A framework to run code over large data

Hadoop Overview

Hadoop Cluster Management

Hadoop is an Apache Project

Apache Software Foundation Open Source project
- Originally a sub-project of the Lucene search engine.

Now a full Apache project
- With sub-projects if its own: Core, HBase, Hive, Pig, ZooKeeper
- Yahoo! is the major source code contributor to Core, Pig and ZooKeeper
- Powerset (now Microsoft) has contributed HBase, a columnar-storage database which uses HDFS
  - Powerset has adapted Hadoop to run on Amazon's EC2
- Facebook has contributed Hive, an SQL-like interface on Hadoop
- There is significant interest from other companies: Amazon (Elastic Hadoop!), IBM (Eclipse), many start-ups, even Google

Hadoop Distributed File System

Cannot be mounted as a “file system”
- Access via command line or Java API
- Not a Unix-like, POSIX file system

Prefers large files (multiple terabytes) to many small files

Files are write once, read many
- Append operation available in Hadoop v0.18+

Systems Administration friendly features
- Users, Groups and Permissions (Hadoop v0.16)
- Name and Space Quotas (Hadoop v0.18 and v0.19)
- Per-user client-side trash can (Hadoop v0.12)
- Snapshot (on restart)
- Rebalance tool (Hadoop v0.16)
- Audit logs (Hadoop v0.18)

Interacting with the HDFS

Simple commands: hadoop dfs -ls, -du, -rm, -rmr

Uploading files

- $ hadoop dfs -put foo mydata/foo
- $ cat ReallyBigFile | hadoop dfs -put - mydata/ReallyBigFile

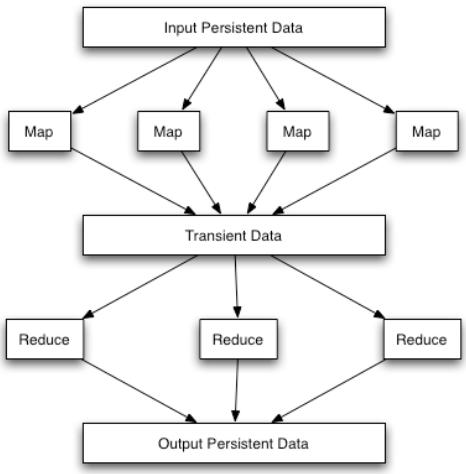
Downloading files

- $ hadoop dfs -get mydata/foo foo
- $ hadoop dfs -get - mydata/ReallyBigFile | grep “the answer is”
- $ hadoop dfs -cat mydata/foo
- $ hadoop dfs –tail [–f] mydata/foo

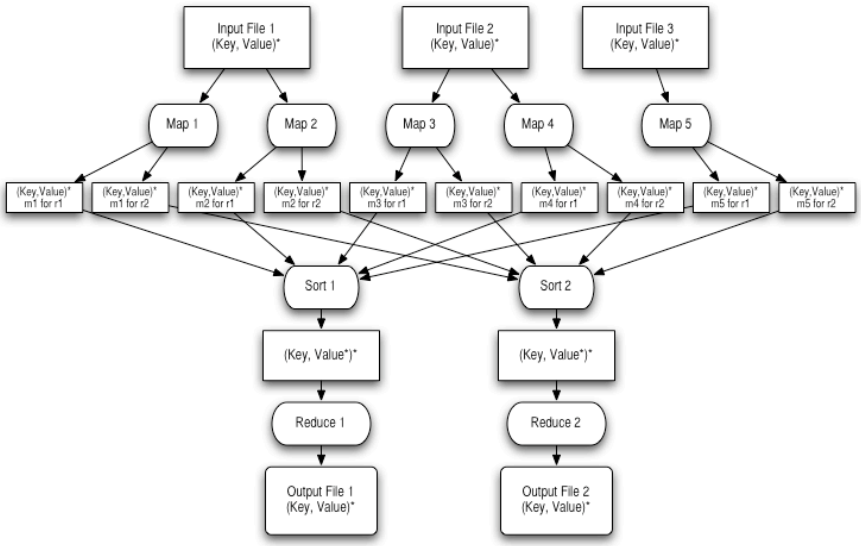
Configurable Options

- Blocksize and Replication factor are configurable per file

Map/Reduce Dataflow



Map/Reduce Dataflow Details



Hadoop Installation Demo

Hadoop Cluster Management

“Small Scale” Hardware

The demo cluster is built out of dual-socket, dual core Intel machines with 4 250GB drives, and 3GB of RAM

- Each system is worth less than $2K
- You can now buy better systems for under $2K

For some larger clusters, specialized hardware is required

- Big RAM NameNodes
- NFS device for metadata backup
- Flat network with big core switches

Basic Installation

Three URLs:

- http://hadoop.apache.org/core/docs/current/cluster_setup.html
- http://hadoop.apache.org/core/releases.html
- http://mirrors.kahuki.com/apache/hadoop/core/stable/hadoop-0.16.4.tar.gz

19 hosts: 1 master (NameNode and JobTracker) and 18 slaves

Simple SSH (with a key and key-agent) to distribute code and conf

Installation Commands

These commands assume an NFS mounted directory

- There are many ways to do this, scp, rsync, curl, etc.

Copy out Java and make a symlink to it

$ for i in `cat ~/slaves`; do ssh $i '(cd /grid/0/marco; tar xzf ~/jre1.6.0_06.tgz)'; done

$ for i in `cat ~/slaves`; do ssh $i '(cd /grid/0/marco; ln -s jre1.6.0_06 jre)'; done

Copy out Hadoop (v0.16.4 in this example) and make a symlink to that

$ for i in `cat ~/slaves`; do ssh $i '(cd /grid/0/marco; tar xzf ~/hadoop-0.16.4.tar.gz)'; done

$ for i in `cat ~/slaves`; do ssh $i '(cd /grid/0/marco; ln -s hadoop-0.16.4 hadoop)'; done

Create stub directories for HDFS and MapReduce

$ for i in `cat ~/slaves`; do ssh $i 'mkdir /grid/0/marco/data; mkdir /grid/1/marco/data; mkdir /grid/2/marco/data; mkdir /grid/3/marco/data'; done

$ for i in `cat ~/slaves`; do ssh $i 'mkdir /grid/0/marco/mrtmp; mkdir /grid/1/marco/mrtmp; mkdir /grid/2/marco/mrtmp; mkdir /grid/3/marco/mrtmp'; done

Configuration

```
$ vi hadoop-0.16.4/conf/hadoop-env.sh
< # export JAVA_HOME=/usr/lib/j2sdk1.5-sun
---
> # export JAVA_HOME=/grid/0/marco/jre
>
> # HADOOP_HEAPSIZE - The maximum amount of heapsize to use, in MB e.g. 2000MB.
> export HADOOP_HEAPSIZE=512

$ vi hadoop/conf/hadoop-site.xml
> <property>
>   <name>fs.default.name</name>
>   <value>kry4001:9000</value>
> </property>
> <property>
>   <name>mapred.job.tracker</name>
>   <value>kry4001:9001</value>
> </property>
> <property>
>   <name>dfs.replication</name>
>   <value>3</value>
> </property>
> <property>
>   <name>dfs.name.dir</name>
>   <value>/grid/0/marco/name</value>
> </property>
> <property>
>   <name>dfs.data.dir</name>
>
<value>/grid/0/marco/data,/grid/1/marco/data,/grid/2/marco/data,/grid/3/marco/data</value>
> </property>
> <property>
>
<name>/grid/0/marco/mrtmp,/grid/1/marco/mrtmp,/grid/2/marco/mrtmp,/grid/3/marco/mrtmp</name>
>   <value></value>
> </property>
> <property>
>   <name></name>
>   <value></value>
> </property>
```

Starting HDFS

Set Environment variables

- $ export JAVA_HOME=/grid/0/marco/jre
- $ export HADOOP_HOME=/grid/0/marco/hadoop

Disable Secondary NameNode

- cp /dev/null hadoop/conf/masters

Copy out hadoop configs

- $ for i in `cat ~/slaves`; do echo -n "${i}: "; ssh $i 'cp hadoop-site.xml hadoop-env.sh /grid/0/marco/hadoop/conf'; echo "${i} done"; done

Start HDFS

- $ bin/start-dfs.sh

start-dfs.sh output

```
$ bin/start-hdfs.sh
starting namenode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-namenode-kry4001.out
kry4002: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4002.out
kry4004: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4004.out
kry4009: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4009.out
kry4014: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4014.out
kry4013: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4013.out
kry4005: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4005.out
kry4008: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4008.out
kry4011: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4011.out
kry4017: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4017.out
kry4019: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4019.out
kry4015: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4015.out
kry4007: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4007.out
kry4018: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4018.out
kry4003: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4003.out
kry4020: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4020.out
kry4012: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4012.out
kry4006: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4006.out
kry4016: starting datanode, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-datanode-kry4016.out
```

Review NameNode WebUI

NameNode 'kry4001.inktomisearch.com:9000'

Started: Thu Jun 19 22:45:23 UTC 2008
Version: 0.16.4, r652614
Compiled: Fri May 2 00:18:12 UTC 2008 by hadoopqa
Upgrades: There are no upgrades in progress.

Browse the filesystem

Cluster Summary

1 files and directories, 0 blocks = 1 total. Heap Size is 17.44 MB / 493.06 MB (3%)

Capacity	:	15.98 TB
DFS Remaining	:	14.86 TB
DFS Used	:	1.41 MB
DFS Used%	:	0 %
Live Nodes	:	18
Dead Nodes	:	0

HDFS Nodes

Live Datanodes : 18

Node	Last Contact	Admin State	Size (GB)	Used (%)	Used (%)	Remaining (GB)	Blocks
kry4002	1	In Service	909.95	0		846.58	0
kry4003	2	In Service	909.08	0		845.77	0
kry4004	1	In Service	909.08	0		845.77	0
kry4005	0	In Service	909.08	0		845.77	0
kry4006	2	In Service	909.08	0		845.77	0
kry4007	2	In Service	909.95	0		846.58	0
kry4008	0	In Service	909.08	0		845.77	0
kry4009	1	In Service	908.21	0		844.96	0
kry4011	0	In Service	909.95	0		846.58	0
kry4012	0	In Service	909.08	0		845.77	0
kry4013	0	In Service	909.08	0		845.77	0
kry4014	0	In Service	908.21	0		844.96	0
kry4015	0	In Service	908.21	0		844.96	0
kry4016	2	In Service	908.21	0		844.96	0
kry4017	0	In Service	908.21	0		844.96	0
kry4018	0	In Service	908.21	0		844.96	0
kry4019	0	In Service	908.21	0		844.96	0
kry4020	2	In Service	909.08	0		845.77	0

Dead Datanodes : 0

Upload a File

Upload a 156M file (1,000 copies of Alice in Wonderland)

$ bin/hadoop dfs -ls

Found 0 items

$ bin/hadoop dfs -put ~/1K-alice30.txt .

$ bin/hadoop dfs -ls

Found 1 items

/user/marco/1K-alice30.txt <r 3> 163218000 2008-06-19 23:19 rw-r--r-- marco supergroup

Starting Map-Reduce

$ bin/start-mapred.sh
starting jobtracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-jobtracker-kry4001.out
kry4003: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4003.out
kry4009: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4009.out
kry4007: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4007.out
kry4002: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4002.out
kry4004: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4004.out
kry4005: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4005.out
kry4017: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4017.out
kry4014: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4014.out
kry4008: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4008.out
kry4013: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4013.out
kry4011: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4011.out
kry4015: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4015.out
kry4006: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4006.out
kry4012: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4012.out
kry4018: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4018.out
kry4020: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4020.out
kry4016: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4016.out
kry4019: starting tasktracker, logging to /grid/0/marco/hadoop/bin/../logs/hadoop-marco-tasktracker-kry4019.out

Review JobTracker WebUI

kry4001 Hadoop Map/Reduce Administration

State: RUNNING
Started: Thu Jun 19 22:49:34 UTC 2008
Version: 0.16.4, r652614
Compiled: Fri May 2 00:18:12 UTC 2008 by hadoopqa
Identifier: 200806192249

Cluster Summary

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node
0	0	0	18	36	36	4.00

Running Jobs

Running Jobs
none

Run a Job

$ bin/hadoop jar hadoop-0.16.4-examples.jar wordcount -r 4 1K-alice30.txt alice-wc.out

```
08/06/19 23:21:23 INFO mapred.FileInputFormat: Total input paths to process : 1
08/06/19 23:21:23 INFO mapred.JobClient: Running job: job_200806192249_0003
08/06/19 23:21:24 INFO mapred.JobClient:  map 0% reduce 0%
08/06/19 23:21:34 INFO mapred.JobClient:  map 27% reduce 0%
08/06/19 23:21:39 INFO mapred.JobClient:  map 35% reduce 0%
08/06/19 23:21:44 INFO mapred.JobClient:  map 42% reduce 0%
08/06/19 23:21:49 INFO mapred.JobClient:  map 62% reduce 0%
08/06/19 23:21:54 INFO mapred.JobClient:  map 63% reduce 0%
08/06/19 23:21:59 INFO mapred.JobClient:  map 72% reduce 0%
08/06/19 23:22:04 INFO mapred.JobClient:  map 80% reduce 11%
08/06/19 23:22:09 INFO mapred.JobClient:  map 82% reduce 11%
08/06/19 23:22:14 INFO mapred.JobClient:  map 90% reduce 11%
08/06/19 23:22:19 INFO mapred.JobClient:  map 92% reduce 11%
08/06/19 23:22:24 INFO mapred.JobClient:  map 100% reduce 11%
08/06/19 23:22:38 INFO mapred.JobClient:  map 100% reduce 33%
08/06/19 23:22:39 INFO mapred.JobClient:  map 100% reduce 100%
08/06/19 23:22:40 INFO mapred.JobClient: Job complete: job_200806192249_0003
08/06/19 23:22:40 INFO mapred.JobClient: Counters: 12
08/06/19 23:22:40 INFO mapred.JobClient:   Job Counters
08/06/19 23:22:40 INFO mapred.JobClient:     Launched map tasks=3
08/06/19 23:22:40 INFO mapred.JobClient:     Launched reduce tasks=7
08/06/19 23:22:40 INFO mapred.JobClient:     Data-local map tasks=1
08/06/19 23:22:40 INFO mapred.JobClient:   Map-Reduce Framework
08/06/19 23:22:40 INFO mapred.JobClient:     Map input records=3853000
08/06/19 23:22:40 INFO mapred.JobClient:     Map output records=28200000
08/06/19 23:22:40 INFO mapred.JobClient:     Map input bytes=163218000
08/06/19 23:22:40 INFO mapred.JobClient:     Map output bytes=265799000
08/06/19 23:22:40 INFO mapred.JobClient:     Combine input records=28200000
08/06/19 23:22:40 INFO mapred.JobClient:     Combine output records=100436
08/06/19 23:22:40 INFO mapred.JobClient:     Reduce input groups=5908
08/06/19 23:22:40 INFO mapred.JobClient:     Reduce input records=100436
08/06/19 23:22:40 INFO mapred.JobClient:     Reduce output records=5908
```

Review Job

Hadoop job_200806192249_0003 on kry4001

User: marco
Job Name: wordcount
Job File: /tmp/hadoop-marco/mapred/system/job_200806192249_0003/job.xml
Status: Running
Started at: Thu Jun 19 23:21:23 UTC 2008
Running for: 10sec

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	0.00%	3	0	3	0	0	0 / 0
reduce	0.00%	4	0	4	0	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Launched map tasks	0	0	3
	Launched reduce tasks	0	0	4
	Data-local map tasks	0	0	1

Job Progress

Hadoop job_200806192249_0003 on kry4001

User: marco
Job Name: wordcount
Job File: /tmp/hadoop-marco/mapred/system/job_200806192249_0003/job.xml
Status: Running
Started at: Thu Jun 19 23:21:23 UTC 2008
Running for: 41sec

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	80.17%	3	0	2	1	0	0 / 0
reduce	11.11%	4	0	4	0	0	0 / 0

Validate Output

```
$ hadoop dfs -get alice-wc.out/\*
$ ls -1sh
total 80K
20K part-00000
20K part-00001
20K part-00002
20K part-00003
$ egrep '^Alice[:space:]+' *
part-00003:Alice:       7000
```


Hadoop for Data Intensive Super Computing (DISC)

Hadoop Cluster Management

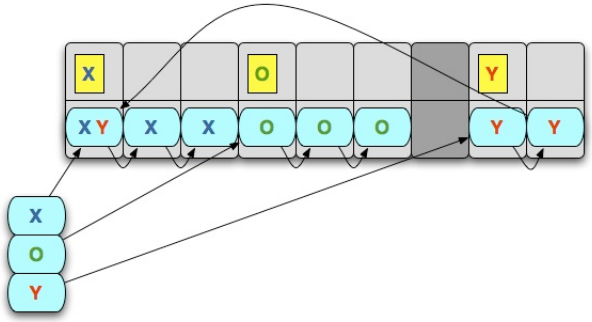
Hadoop: Two Services in One

Input file split into (64MB) blocks during loading

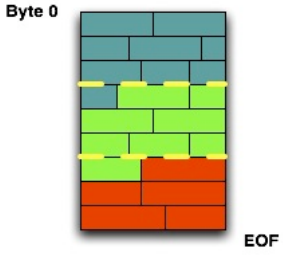
- Data flows straight from client to DataNodes (no bottleneck)
- Bad nodes are automatically skipped

Replicas automatically streamed to other DataNodes

Map-Reduce is location aware, optimizes placement of tasks



Job Input: InputSplit



Some network traffic is likely

- M-R handles this transparently behind the scenes, by asking for "just a bit" of the next block
- Unless 1st, ignore all before 1st separator
- Read-ahead to next block to complete last record

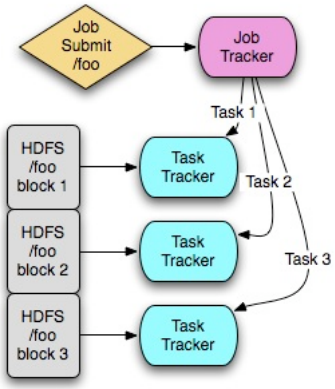
InputFormat

- SequenceFileInputFormat, TextInputFormat

Map-Reduce Process Level

Job Submission

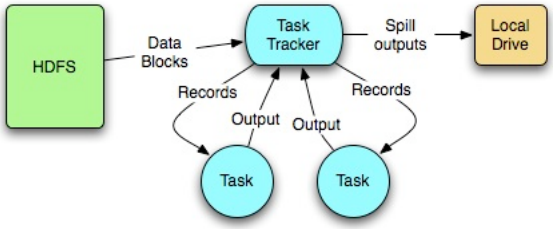
- Map Function + Reduce Function + List of inputs
- JobTracker receives list of inputs, breaks inputs into discrete map tasks
- Map tasks are marshaled to long-running TaskTrackers
- Map outputs are collected by Reducer tasks (not shown) sorted, and fed to the reduce function
- Reduce outputs (not shown) are stored back to the HDFS



Map-Reduce Process Level (2)

TaskTracker is responsible for handling inputs/outputs

- Map inputs are read (typically from HDFS) and fed, record by record, to each task
- Map outputs are collected in memory, and if necessary, spilled to local disk
- Partitioned outputs are typically retrieved by TaskTrackers hosting reduce tasks, and those outputs are typically stored back in HDFS



Hadoop Programming APIs

Hadoop Cluster Management

Java: MapperClass

```
public static class MapClass extends MapReduceBase
    implements Mapper {

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(WritableComparable key, Writable value,
                  OutputCollector output,
                  Reporter reporter) throws IOException {
    String line = ((Text)value).toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      output.collect(word, one);
      reporter.incrCounter(Counter.WORDS, 1);
    }
  }
}
```

Java: CombinerClass / ReducerClass

```
public static class Reduce extends MapReduceBase implements
    Reducer {

  public void reduce(WritableComparable key, Iterator values,
                     OutputCollector output,
                     Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += ((IntWritable) values.next()).get();
      reporter.incrCounter(Counter.VALUES, 1);
    }
    output.collect(key, new IntWritable(sum));
  }
}
```


Java: Launcher

```
public static void main(String[] args) throws IOException {
  JobConf conf = new JobConf(WordCount.class);
  conf.setJobName("wordcount");

  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(IntWritable.class);

  conf.setMapperClass(WCMap.class);
  conf.setCombinerClass(WCReduce.class);
  conf.setReducerClass(WCReduce.class);

  conf.setInputPath(new Path(args[0]));
  conf.setOutputPath(new Path(args[1]));

  JobClient.runJob(conf);
}
```

Streaming Map-Reduce Demo

$ run jar hadoop-streaming.jar
-numReduceTasks 4
-input /data/news/20070218
-output wc-streaming.out
-mapper "**perl -ane 'print join(\"\n\", @F), \"\n\"'**"
-reducer "**uniq -c**"

Currently, combiner only available as a JavaClass, cannot provide shell command as a combiner (HADOOP-4842, Hadoop v0.21)

File: /user/marco/wc-streaming.out/part-00000

Goto : /user/marco/wc-streamin go

Go back to dir listing
Advanced view/download options

View Next chunk View Prev chunk

```
     88 Moraga
     24 Moraga,
      1 Moraga/Fotos
      9 Moragahakanda
      3 Moragahakanda,
      4 Moragne
      1 Moragne,
      4 Moragues.
      4 Moraine-Pewaukee
      3 Moraine.
    147 Morais
      2 Morais's
     80 Morais,
      5 Moraka
     15 Moraka,
      1 Morakul,
      1 Moral'
      1 Moral,"
      3 Moral-Luc??a
      2 Moral-Spritze
     18 Moral.
      7 Moral:
      1 Moral?]]><\/body>
      2 Morala]]><\/title>
      1 Moralas
```

Download this file
TAIL this file
Chunk Size to view (in bytes, upto file's DFS blocksize): 2097152 Refresh

Total number of blocks: 2
org.apache.hadoop.dfs.LocatedBlock@f23851: 74.6.130.87:50010 72.30.62.167:50010 72.30.62.152:50010
org.apache.hadoop.dfs.LocatedBlock@4f88cb: 72.30.62.152:50010 72.30.62.213:50010 72.30.62.149:50010

Pig

A high-level declarative language (similarity to SQL)

- Pig Latin is a simple query algebra that lets you express data transformations such as merging data sets, filtering them, and applying functions to records or groups of records. Users can create their own functions to do special-purpose processing.

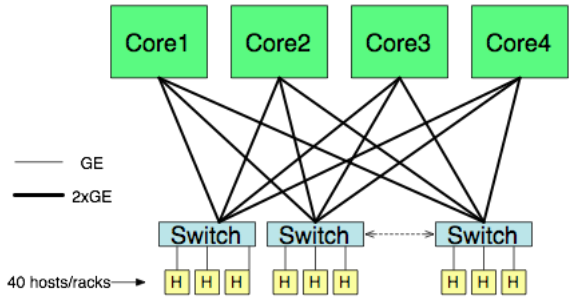
A SW layer above MR, implementing a grouping syntax

```
grunt> A = load '1K-alice30.txt' using
  TextLoader()
grunt> W = foreach A generate
  flatten(TOKENIZE(*));
grunt> G = GROUP W BY $0;
grunt> C = FOREACH G GENERATE group, COUNT(W);
grunt> store C into 'alice-wc.pig';
```

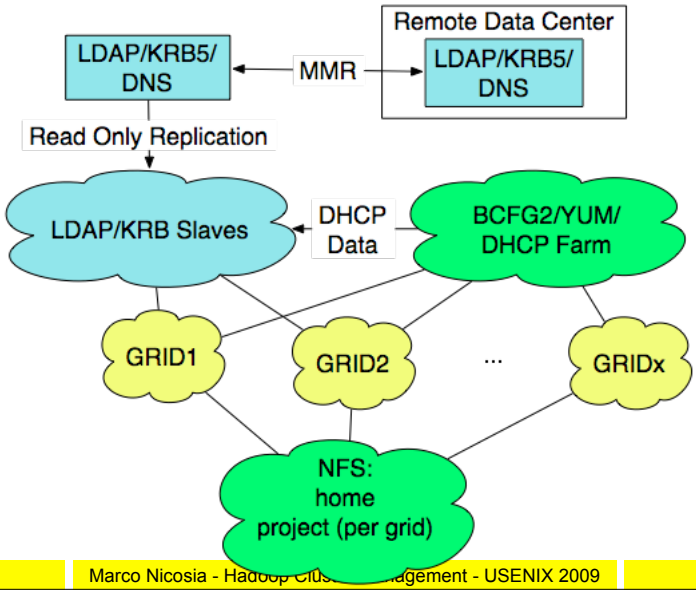

Systems Concerns

Hadoop Cluster Management

Flat Networking



System Management



System Configuration

Memory (typically) dictates how many task slots per node

- Nodes need not be homogenously configured

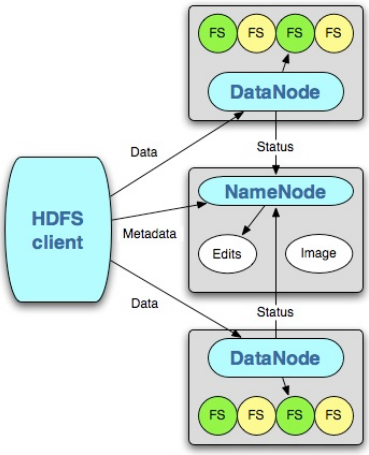
Drives can be configured either as RAID0 or JBOD

- RAID0 theoretically faster
- Losing a single drive in RAID0 means losing an entire node's worth of replicas (which may be tolerable)
- Creating a RAID0 stripe allows for larger single-task outputs than if restricted to a single drive

HDFS Advanced Topics

Hadoop Cluster Management

HDFS Process Level



- NameNode and DataNodes run continuously
- DataNodes report status to NameNode via heartbeats
- NameNode inspects state of the file system, instructs DataNodes to perform operations on blocks (replicate, delete) via heartbeat responses
- Metadata (file creation, new block, etc) is logged by the NameNode
- HDFS clients run on **or** off the HDFS nodes
- Block replicas are pipelined (not shown) from the first DataNode to subsequent nodes

NameNode Overview

Namespace

- HDFS supports a traditional hierarchical file organization
- A user or an application can create directories and store files inside these directories

Block Replication

- NameNode is responsible for keeping track of which datanodes all block replicas are stored on, and maintaining replication policy
- Datanodes only track what blocks they have on local disk, and report to NameNode. They don't know anything about replication policy or file metadata

HDFS fsck

Usage: hadoop fsck [GENERIC_OPTIONS] <path> [-move | -delete | -openforwrite] [-files [-blocks [-locations | -racks]]]

<path>
- Start checking from this path.

-move
- Move corrupted files to /lost+found

-delete
- Delete corrupted files.

-openforwrite
- Print out files opened for write.

-files
- Print out files being checked.

-blocks
- Print out block report.

-locations
- Print out locations for every block.

-racks
- Print out network topology for data-node locations.

HDFS Safemode

hadoop dfs [-safemode enter | leave | get | wait]

During start up Namenode loads the filesystem state from fsimage and edits log file. It then waits for datanodes to report their blocks so that it does not prematurely start replicating the blocks though enough replicas already exist in the cluster. During this time Namenode stays in safemode. A Safemode for Namenode is essentially a read-only mode for the HDFS cluster, where it does not allow any modifications to filesystem or blocks. Normally Namenode gets out of safemode automatically at the beginning. If required, HDFS could be placed in safemode explicitly using 'bin/hadoop dfsadmin -safemode' command. Namenode front page shows whether safemode is on or off. A more detailed description and configuration is maintained as JavaDoc for setSafeMode().

Note: Recent versions have a confusing bug:

"The ratio of reported blocks 1.0000 has not reached the threshold 1.0000" (HADOOP-5650)

Hadoop DFS Upgrades

Upgrade Process

- Enable safemode
- Run fsck, lsr, and dfsadmin –report, save outputs
- stop-dfs.sh
- Restart NameNode only to digest edits
- Install new Hadoop version
- start-dfs.sh –upgrade
- Run fsck, lsr, and dfsadmin –report, save outputs

Later

$ bin/hadoop dfsadmin –finalizeUpgrade

- Must be run while the HDFS is up!

OR

$ bin/start-dfs.h -rollback

DataNode Lists

slaves

- Used by start-*.sh/stop-*.sh

dfs.include and dfs.exclude

- IPs or FQDNs of hosts allowed/ignored in the HDFS

Active datanode list: include list - exclude list

- Dead list in NameNode Status

Optional HDFS Configuration

dfs.datanode.du.reserved
- Reserved space in bytes per volume. Always leave this much space free for non dfs use.

dfs.datanode.du.pct
- When calculating remaining space, only use this percentage of the real available space

dfs.hosts
- Names a file that contains a list of hosts that are permitted to connect to the namenode. The full pathname of the file must be specified. If the value is empty, all hosts are permitted.

dfs.hosts.exclude
- Names a file that contains a list of hosts that are not permitted to connect to the namenode. The full pathname of the file must be specified. If the value is empty, no hosts are excluded.

Primary NameNode state

During normal operation, all state is kept in memory
- A subset of state is stored on disk in a file called 'fsimage'
- This on-disk copy is used to persist state data across NameNode restarts
- All other state is dynamically regenerated after restart

All metadata transactions (adds, deletes, renames, etc) are also logged to disk
- Edits are committed to a file called, 'edits'

The Primary NameNode cannot digest the edits log into the image file without restart
- Allowing the edits log to get large isn't dangerous, but possibly risky
- A large edits file means very long restart times as primary digests large edits file
- Admin command saveNamespace (HADOOP-4826, v0.20)

NameNode SPOF

The chances of a single node in 1,000 failing are very high

The chances of a specific node in 1,000 failing are pretty rare

- NameNodes don't fail very often

We run our NameNode with no RAID

- NameNode has a feature to write image/edits to multiple locations
  - dfs.name.dir
    - Determines where on the local filesystem the DFS name node should store the name table. If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.
- Upon startup the NameNode automatically consults all configured locations of its state and reads the most up to date image and journal. If all of the Namenodes copies of data are unavailable state can be (mostly) recovered from the secondary Namenode using the '-importCheckpoint' switch. (HADOOP-2585)

NameNode SPOF (2)

- Second location is an NFS device which has RAID
  - Moving to a different NameNode is easy
    - Copy image and edits from NFS device
    - Move VIP
    - Restart HDFS (Optional)

Secondary NameNode

Offline digestion of edits file for the Primary NN

- Process requests both edits and image file from Primary.
- Primary closes current edits file, starts writing to edits.new.
- Nice side-benefit: Keeps at least one copy of the image and edits file

NOT a suitable choice for Primary NameNode failover

- If you need to fail over primary to a second server, you still want the secondary to function normally
- Generally a bad idea to run both on the same machine

Secondary NameNode (2)

Two configuration options:

- fs.checkpoint.period, set to 1 hour by default, specifies the maximum delay between two consecutive checkpoints
- fs.checkpoint.size, set to 64MB (we use 2GB) by default, defines the size of the edits log file (in bytes) that forces an urgent checkpoint even if the maximum checkpoint delay is not reached.

Standby NameNodes

Streaming edits to a standby NameNode (HADOOP-4539, v0.21.0)

- Introduces Backup node
- Replaces Secondary NameNode with Checkpoint node

HDFS Permissions

Hadoop client draws user and group membership from client OS

- Implementation is naive, can be easily fooled

dfs.permissions - Optional feature

- Permissions are tracked throughout the system regardless
- Only permissions enforcement is optional

dfs.permissions.supergroup

- The name of the group of super-users.

dfs.umask

- The umask used when creating files and directories. **Decimal**, not octal! (Yahoo! uses 63.)

dfs.web.ugi

- The user account used by the web interface. (We use gopher,gopher, effectively “anonymous.”)

Hadoop Authentication

Hadoop has no HDFS or job client authentication
- Breathe. Relax.

Hadoop is still not 1.0
- Intention is to go to GSSAPI (Kerberos, HADOOP-1741)
- Code is not written yet

Currently handled by a 'Systems Engineering' solution
- Users are only given Unix login to a set of 'gateway' machines
- Only admins are given access to compute nodes
- Compute nodes run IP Tables rules to only accept HDFS and job submit from gateways
- Crude, but has allowed Dev team to focus on other pre-1.0 features for more than a year

HDFS Audit Logs

Undocumented! (HADOOP-3336, v0.18.0)

This is one of your best tools to identify a (D)DOS “attack” against the NameNode
- Usually a well meaning but oblivious user

Add/Uncomment these lines to your NameNode’s log4j.properties file:

```
log4j.logger.org.apache.hadoop.fs.FSNamesystem.audit=INFO,DRFAAUDIT
log4j.additivity.org.apache.hadoop.fs.FSNamesystem.audit=false
log4j.appender.DRFAAUDIT=org.apache.log4j.DailyRollingFileAppender
log4j.appender.DRFAAUDIT.File=/var/log/hadoop-audit.log
log4j.appender.DRFAAUDIT.DatePattern=.yyyy-MM-dd
log4j.appender.DRFAAUDIT.layout=org.apache.log4j.PatternLayout
log4j.appender.DRFAAUDIT.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
```

Hadoop Rack Awareness

The HDFS and the Map-Reduce components are rack-aware (HADOOP-692, v0.11.0)

- The API resolves the slave's DNS name (also IP address) to a rack id.
- Configure topology.script.file.name.
  - If topology.script.file.name is not set, the rack id /default-rack is returned for any passed IP address.
- Additional configuration in the Map-Reduce part is mapred.cache.task.levels
  - Determines the number of levels (in the network topology) of caches.
  - So, for example, if it is the default value of 2, two levels of caches will be constructed - one for hosts (host -> task mapping) and another for racks (rack -> task mapping)

3 replicas

- Local, different node on same rack, node on different rack

Decommissioning Nodes

hadoop dfsadmin -refreshNodes

- Replicates blocks from any live nodes in the exclude list
- Do not decommission too many nodes (200+) at once! It's easy to saturate the NameNode.
- Decommissioned nodes will show up in "dead" list on dfs WebUI after they've completed and shut down

"Each entry defined in dfs.hosts and also in dfs.host.exclude is stopped from decommissioning if it has aleady been marked for decommission. Entires not present in both the lists are decommissioned."

Space Management

Older versions of Hadoop produce very confusing space reports. Two bugs address tehse issues: HADOOP-2816 WebUI, HADOOP-4281 dfsadmin -report, v0.19

Datanode heartbeat reported capacity information is: sum of all the diskspace of data directories minus the reserved space configured using dfs.datanode.du.reserved config param

Cluster Summary

Configured Capacity: Sum of the file system capacity of all the data directories - reserved space

Present Capacity: Represents the present capacity available for DFS use. This is sum of DFS Remaining and DFS Used

DFS Used%: Calculated based on Present Capacity

Node data columns

- Last Contact, Admin State, Capacity (TB), Present Capacity (TB) Used (%), Used (%) Remaining (TB), Blocks
- Capacity: Sum of file system capacity of all the data directories - reserved space

HDFS Quotas

- HDFS will become unusable (and possibly unstable) when approaching the maximum capacity limits
  - If jobs cannot write output, cluster is effectively down
- To prevent rogue users or jobs from affecting the whole cluster, HDFS has name space (v0.18) and disk space (v0.19) quotas
  - Quotas are applied to directory trees, not users!
    - Allows multiple users to share a single, quota'd project directory
  - Quotas must always be manually set, there is no way to specify a default
  - Setting quotas will fail if the directory tree already exceeds the specified quota
  - There is also no way to audit quotas, only to report on quotas currently configured on a directory (HADOOP-5290)
    - $ hadoop dfs -count -q <directory>...<directory>

HDFS Name Space Quotas

- To keep metadata transactions fast, the entire namespace is held within NameNode RAM.
  - If NameNode has insufficient RAM, or users are creating needlessly small files, NameNode RAM capacity may artificially limit HDFS capacity
  - $ hadoop dfsadmin -setQuota <N> <directory>...<directory>
    - Largest quota of <N> is Long.Max_Value
  - $ hadoop dfsadmin -clrQuota <directory>...<directory>

HDFS Disk Space Quotas

A limit on the number of bytes used by a directory tree

- Each replica of a block counts against the quota
  - Changing the replication factor for a file will credit or debit quotas
- A quota of zero will allow files and directories to be created (no disk space used) but no blocks can be committed

$ hadoop dfsadmin -setSpaceQuota <N> <directory>...<directory>

- N can also be specified with a binary prefix for convenience, for e.g. 50g for 50 gigabytes and 2t for 2 terabytes etc.

$ hadoop dfsadmin -clrSpaceQuota <directory>...<directory>

HDFS Archive Tool

Quotas are complemented by ‘Hadoop archives’, which are a tool for users to manage their namespace consumption. A large number of files can be converted into a Hadoop archive using a Map/Reduce utility. A Hadoop archive is basically an HDFS directory with a small number of data files that consist of files from the original set concatenated together. An index stores the location of each file from the original set. Individual files in an archive can be accessed using a special URI with the ‘har’ schema. (HADOOP-3188, v0.18.0)

$ hadoop archive -archiveName name <src>* <dest>

HDFS Trash

fs.trash.interval

- Number of minutes between trash checkpoints. If zero, the trash feature is disabled

Implementation is not intuitive for users and admins

- Solely a means to allow users to recover from their own mistakes
- Only enabled when files are removed via the Hadoop dfs command line
  - Files removed via the Java API, externally or within a job are immediately lost
- Deleted files are actually moved to Trash/Current within user's homedir on HFDS

HDFS Trash (2)

- fs.trash.interval affects the frequency by which the NameNode moves the contents of Current to a timestamped directory within Trash
- Timestamped directories over 6 hours are removed by the NameNode.
  - Not configurable
  - This means that when users delete files, no space is freed for 6 hours unless manually deleted from Trash directories

HDFS Rebalancer

HADOOP-1652, v0.16.0

Important when adding nodes to an existing HDFS

- Spreads data evenly across all nodes
- Improves performance by spreading data across racks
- Gives more breathing room for Map-Reduce jobs

Use with caution – we're still discovering bugs

- Improvements have been implemented in Hadoop v0.18

Do not run directly on NameNode, but a host nearby

$ bin/start-balancer.sh [-threshold threshold]

- Threshold is max percentage of over/under-utilization per data node (0 = perfect balance, balancer will likely never finish)
- Each data node has a limited bandwidth for rebalancing. The default value for the bandwidth is 5MB/s (perhaps 1MB/s, the documentation conflicts)

Copy Between Clusters: DistCp

Distributed tool for large inter-cluster copies

$ hadoop distcp hdfs://nn1:8020/<srcpath> hdfs://nn2:8020/<dstpath>

- Use hftp://<dfs.http.address>/<path> to copy from a different version of HDFS (HFTP is read-only)

Many useful options (see manual on website)

-p {rbugp}: Replication, Block size, User, Group, Permissions

-i: Ignore failures (continue with remaining copies)

-log <logdir>: Write logs

-m <num_maps>: More maps isn't always faster

-overwrite: Overwrites pre-existing files in dstpath

-update: Source replaces destination file if they differ

-delete: Delete files existing in dstpath, but not srcpath

Map-Reduce Topics

Hadoop Cluster Management

JobTracker

As you've seen earlier, JobTracker runs 24/7 and is responsible for supplying TaskTrackers with tasks

Without an advanced scheduler, running and queued jobs are not durable across restart

Simple queue, available via web page

- There is no corollary to dfsadmin -report
- Job history logs are available

Job Submission

User code is typically responsible for job submission, including setting up job configuration

- Relevant files (jar, config) are uploaded to HDFS and automatically set to replication 10
- Use DistributedCache to automatically upload additional required files

All files and job tasks are contained by a temporary directory

- ${mapred.local.dir}/taskTracker/jobcache/$jobid/$taskid
- Temporary directories are cleaned up after task exits and map outputs have been collected
- It is possible, but ill-advised for user code to write outside of that directory

Memory Management

Users/admins can also specify the maximum virtual memory of the launched child-task, and any sub-process it launches recursively, using mapred.child.ulimit. Note that the value set here is a per process limit. The value for mapred.child.ulimit should be specified in kilo bytes (KB). And also the value must be greater than or equal to the -Xmx passed to JavaVM, else the VM might not start.

Scheduling

Three different types of scheduling

- Simple JobTracker queues
  - The core of Hadoop job execution
  - Simple queue to supply TaskTrackers with map and reduce tasks
- Plug-in Schedulers (Fair Share, Capacity)
  - Additional layer on top of JobTracker queue, to allow advanced configuration of queue management
- Hadoop-On-Demand
  - Dynamically allocates a private set of machines to a user
  - Deprecated, requires Torque (and Maui)
  - Allocates dynamic Map-Reduce clusters on top of static (or dynamice) HDFS
  - Good for handling explosive growth, bad for overall utilization

JobTracker Queue

The JobTracker maintains a simple queue of jobs that have been submitted, and assigns map and reduce tasks to TaskTrackers when slots become available

All of the map tasks from the first job in the queue will be scheduled before any map tasks from subsequent jobs

Frequently, map (and even reduce tasks) can run in open slots while reduce tasks from previous jobs continue to run

- Entire jobs can sneak through while a job's mega-reducers grind away on a (relatively) few nodes

Plug-in Schedulers

Schedulers are an attempt to automate fine(r)-grain control over the JobTracker's queue

- Very recent technology, Hadoop v0.20 and v0.21
- I don't have enough concrete experience with these to document them well, see the guides on the Hadoop website

Available as jar files

- Must make sure HADOOP_CLASSPATH includes path to jar
- Also configure hadoop-site.xml to include:

```
<property>
  <name>mapred.jobtracker.taskScheduler</name>
  <value>org.apache.hadoop.mapred.SchedulerName</value>
</property>
```

- Choose one of FairScheduler or CapacityTaskScheduler

Fair Share Scheduler

- When there is a single job running, that job uses the entire cluster. When other jobs are submitted, tasks slots that free up are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. Fair sharing can also work with job priorities - the priorities are used as weights to determine the fraction of total compute time that each job should get.
- Fair Scheduler allows assigning guaranteed minimum shares to pools, which is useful for ensuring that certain users, groups or production applications always get sufficient resources.

Capacity Scheduler

- Supports multiple queues which are guaranteed a fraction of the capacity of the cluster. Free resources can be allocated to any queue beyond its guaranteed capacity, and can be returned to a queue within N minutes of needing them.
- Queues optionally support job priorities. Within a queue, jobs with higher priority will have access to the queue's resources before jobs with lower priority. Queues also support a percentage limit of resources allocated to any one user at a given time.

Hadoop on Demand (HOD)

Deprecated

Before advanced schedulers were written, Yahoo! Implemented, "Hadoop on Demand."

- The cluster ran only HDFS processes
- Using Torque/Maui, users allocated sets of nodes from the cluster, and deployed temporary Map-Reduce clusters onto them
- Users then had a private JobTracker to submit their jobs to, thus allowing many users to share many nodes
- High cost: Mapper nodes idle while relatively few reduces continue to run before releasing all nodes when job finishes

Still possibly useful for Test/QA environments?

Map-Reduce Configuration

mapred.child.java.opts=-Xmx640m
mapred.reduce.parallel.copies=30
mapred.tasktracker.map.tasks.maximum=2
mapred.tasktracker.reduce.tasks.maximum=2
io.sort.factor=100
io.sort.mb=256
io.file.buffer.size=131072
mapred.compress.map.output=false
fs.inmemory.size.mb=256
mapred.map.tasks.speculative.execution=false
mapred.reduce.tasks.speculative.execution=false
ipc.client.idlethreshold=8000
ipc.client.connection.maxidletime=30000
ipc.client.connect.max.retries=5

Best Practices

Hadoop Cluster Management

Best Practices - HDFS

Run fsck and dfsadmin -report regularly, save the output

- Alarm any time HDFS is not HEALTHY
  - Any file missing all three replicas of one or more blocks will be considered CORRUPT
- Be especially aware of chronically under-replicated blocks - Hadoop v0.18 and later have complex append code that cause occasional unrecoverable replicas
  - These broken blocks must be handled manually; usually the file must be deleted or recreated
  - Some versions of Hadoop will consider 0-byte blocks (ie, blocks opened for append) as CORRUPT

Best Practices - Monitoring

Use SLAs

- Minimum number of gateways responding
- Minimum number of Datanodes participating in HDFS
- NameNode available for HDFS transactions
- JobTracker available for job submission

Use two levels of monitoring

- A low-level monitoring system to track the health of the worker nodes - no alarms
- A high-level monitoring system to track the SLAs mentioned above - alarm based on availability

New Features

HDFS proxy (HADOOP-4575)

- An HTTP server for allowing files to be read by non-HDFS clients

Distributed ch{mod,own} (HADOOP-4661)

- Changing lots of file permissions using -R option can take days

Synthetic Load Generator for NameNode testing (HADOOP-3992, HADOOP-4142)

http://hadoop.apache.org/

Marco Nicosia
marco@escape.org