Hadoop 24/7

Allen Wittenauer

March 25, 2009

Dear SysAdmin,

Please set up Hadoop using these machines. Let us know when they are ready for use.

Thanks,
The Users

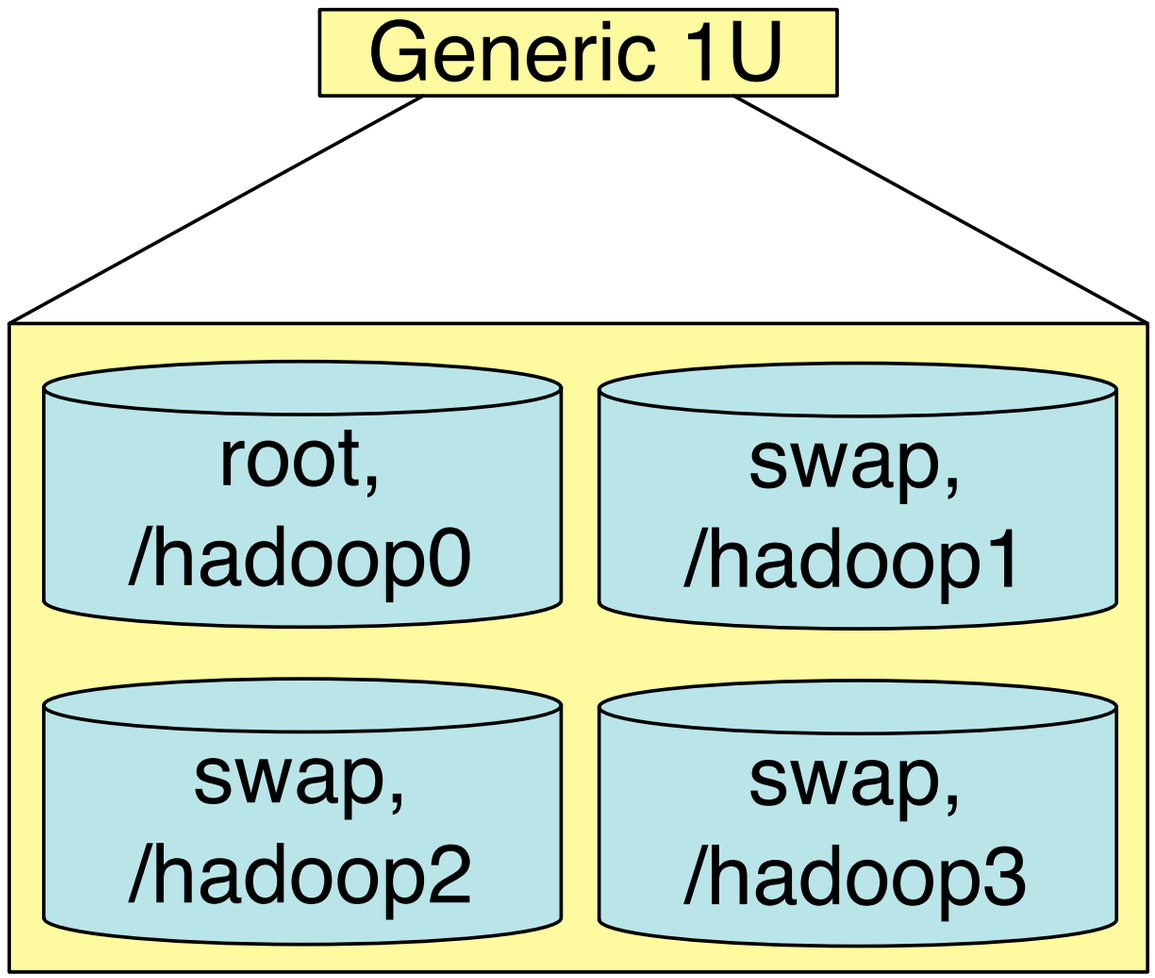
Install some nodes with Hadoop...

Individual Node Configuration

- MapReduce slots tied to # of cores vs. memory
- DataNode reads/writes spread (statistically) even across drives
- hadoop-site.xml dfs.data.dir:

```
<property>
   <name>dfs.data.dir</name>
   <value>/hadoop0,/hadoop1,
      /hadoop2,/hadoop3</value>
</property>
```

- RAID
  - If any, mirror NameNode only
  - Slows DataNode in most configurations



NameNode's Lists of Nodes

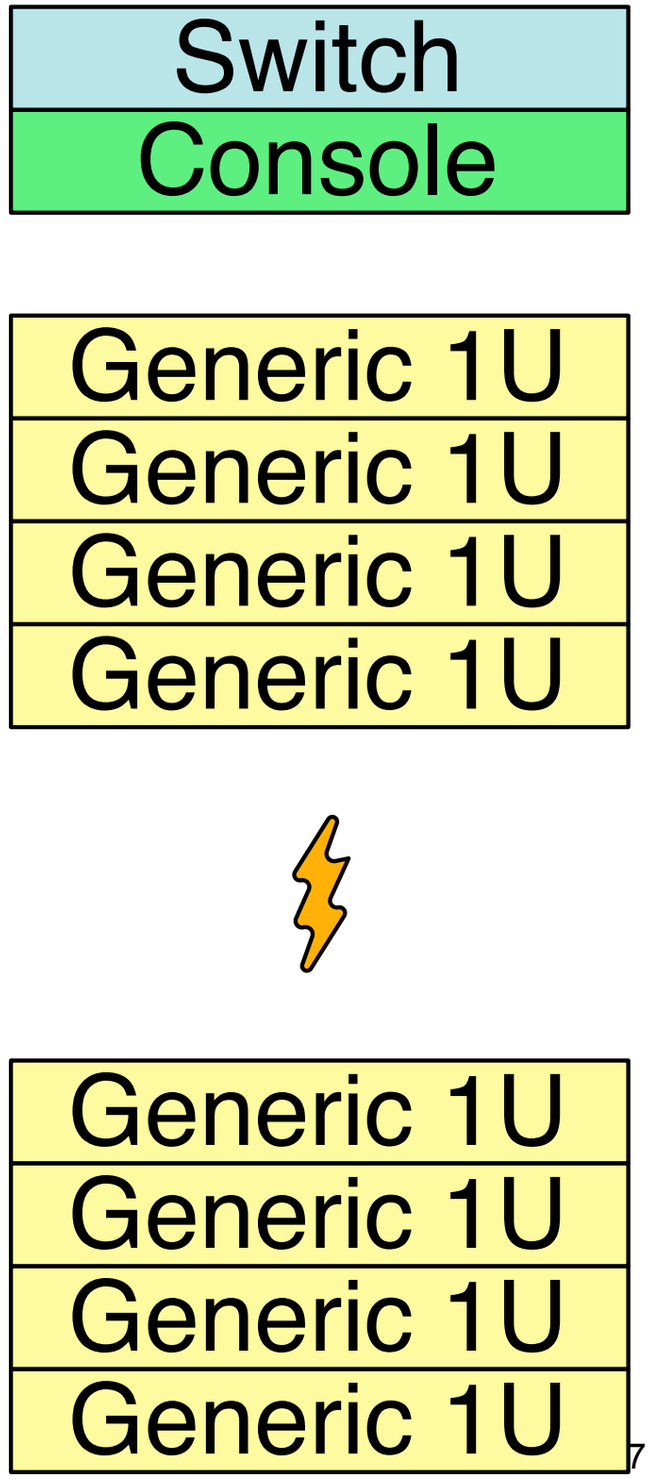
- slaves
  - used by start-*.sh/stop-*.sh
- dfs.include
  - IPs or FQDNs of hosts allowed in the HDFS
- dfs.exclude
  - IPs or FQDNs of hosts to ignore
- active datanode list=include list-exclude list
  - Dead list in NameNode Status

Adding/Removing DataNodes Dynamically

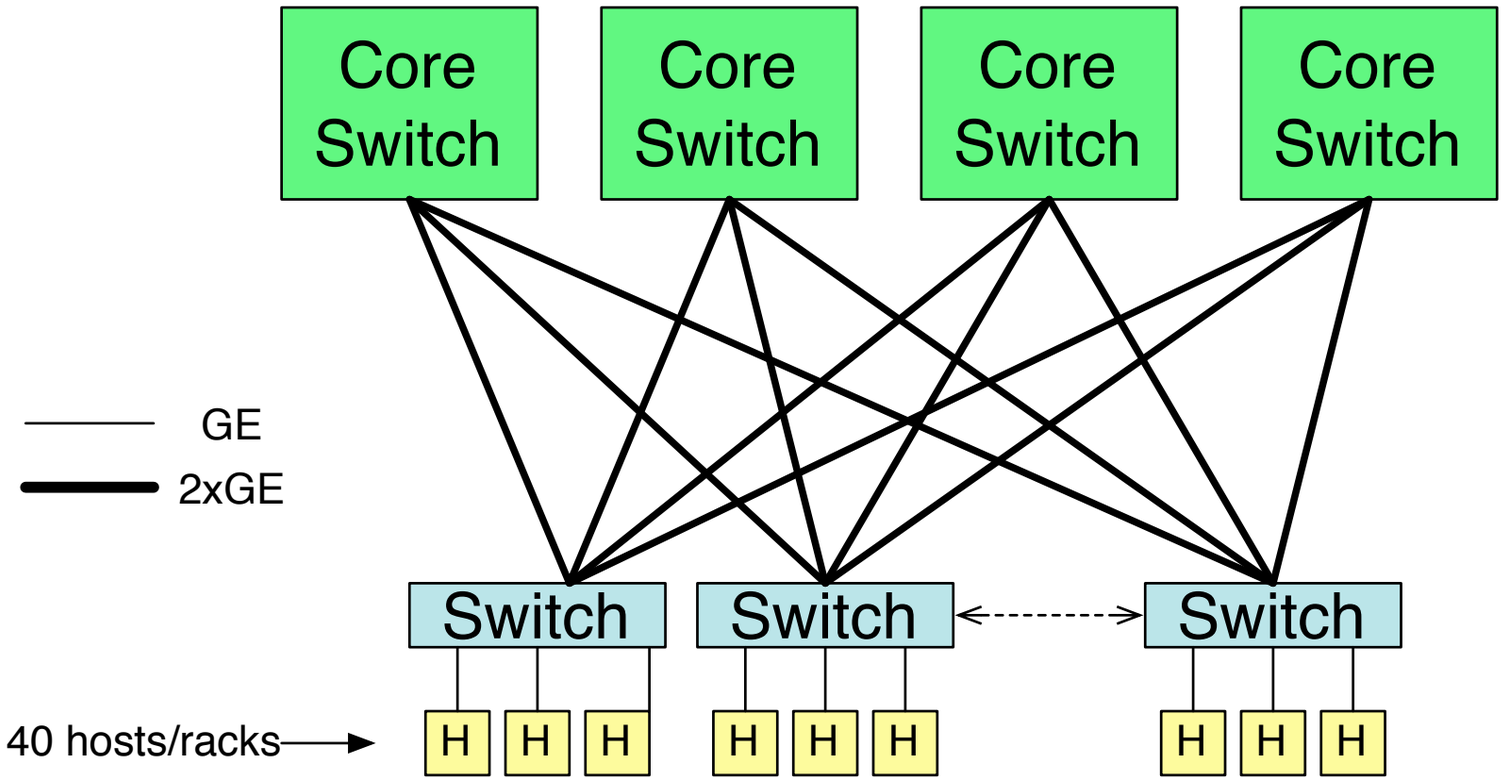
- Add nodes
  - Add new nodes to dfs.include
- (Temporarily) Remove Nodes
  - Add nodes to dfs.exclude
- Update Node Lists and Decommission
  - hadoop dfsadmin -refreshNodes
    - Replicates blocks from any live nodes in the exclude list
  - Hint: Do not decommission too many nodes (200+) at once! Very easy to saturate namenode!

Racks Of Nodes

- Each node
  - 1 connection to network switch
  - 1 connection to console server
- Dedicated
  - Name Nodes
  - Job Tackers
  - Data Loaders
  - ...
- More and More Racks...



Networks of Racks, the Yahoo! Way



- Each switch connected to a bigger switch
- Physically, one big network
- Loss of one core covered by redundant connections
- Logically, lots of small networks (netmask /26)

Rack Awareness (HADOOP-692)

- Hadoop needs node layout (really network) information
  - Speed:
    - read/write prioriti[sz]ation (*)
      - local node
      - local rack
      - rest of system
  - Data integrity:
    - 3 replicas: write local -> write off-rack -> write on-the-other-rack -> return

- Default: flat network == all nodes of cluster are in one rack

- Topology program (provided by **you**) gives network information
  - hadoop-site.xml parameter: topology.script.file.name
  - Input: IP address   Output: /rack information

* or perhaps gettext(“prioritization”) ?

Rack Awareness Example

- Four racks of /26 networks:
  - 192.168.1.1-63, 192.168.1.65-127,
  - 192.168.1.129-191, 192.168.1.193-254

- Four hosts on those racks:
  - sleepy 192.168.1.20 mars 192.168.1.73
  - frodo 192.168.1.145 athena 192.168.1.243

Host to lookup	Topology Input	Topology Output
sleepy	192.168.1.20	/192.168.1.0
frodo	192.168.1.145	/192.168.1.128
mars	192.168.1.73	/192.168.1.64
athena	192.168.1.243	/192.168.1.192

Rebalancing Your HDFS (HADOOP-1652)

- Time passes
  - Blocks Added/Deleted
  - New Racks/Nodes

- Rebalancing places blocks uniformly across nodes
  - throttled so not to saturate network or name node
  - live operation; does not block normal work

- hadoop balancer [-t <threshold>]
  - (see also bin/start-balancer.sh)
  - threshold is % of over/under average utilization
    - 0 = perfect balance = balancer will likely not ever finish
  - Bandwidth Limited: 5 MB/s default, dfs.balance.bandwidthPerSec
    - per datanode setting, need to bounce datanode proc after changing!

- When to rebalance?

HDFS Reporting

- “What nodes are in what racks?”
- “How balanced is the data across the nodes?”
- “How much space is really used?”

- The big question is really:

“What is the health of my HDFS?”

- Primary tools
  - hadoop dfsadmin -fsck
  - hadoop dfsadmin -report
  - namenode status web page

hadoop fsck /

- Checks status of blocks, files and directories on the file system
  - Hint: Partial checks ok; provide path other than /
  - Hint: Run this nightly to watch for corruption

- Common Output:
  - A bunch of dots
    - Good blocks
  - Under replicated blk_XXXX. Target Replication is X but found Y replica(s)
    - Block is under replicated and will be re-replicated by namenode automatically
  - Replica placement policy is violated for blk_XXXX.
    - Block violates topology; need to fix this manually
  - MISSING X blocks of total size Y B
    - Block from the file is completely missing

“Good” fsck Summary

```
 Total size:    506115379265905 B (Total open files size: 4165942598 B)
 Total dirs:    358015
 Total files:   10488573 (Files currently being written: 246)
 Total blocks (validated):          12823618 (avg. block size 39467440 B) (Total
open file blocks (not validated): 51)
 Minimally replicated blocks:       12823618 (100.00001 %)
 Over-replicated blocks:  25197 (0.196489 %)
 Under-replicated blocks: 9 (7.0183E-5 %)
 Mis-replicated blocks:             1260 (0.00982562 %)
 Default replication factor:        3
 Average block replication:         3.005072
 Corrupt blocks:            0
 Missing replicas:                  10 (2.5949832E-5 %)
 Number of data-nodes:              1507
 Number of racks:                   42


The filesystem under path '/' is HEALTHY
```

Bad fsck Summary

Status: CORRUPT

```
 Total size:    505307372371599 B (Total open files size: 2415919104 B)
 Total dirs:    356465
 Total files:   10416773 (Files currently being written: 478)
 Total blocks (validated):      12763719 (avg. block size 39589352 B) (Total open file blocks (not
validated): 288)
  ********************************
```

CORRUPT FILES: **1**

MISSING BLOCKS: **1**

MISSING SIZE: **91227974 B**

CORRUPT BLOCKS: **1**

```
 ********************************
Minimally replicated blocks:   12763718 (99.99999 %)
Over-replicated blocks:        970560 (7.6040535 %)
Under-replicated blocks:       4 (3.133883E-5 %)
Mis-replicated blocks:         1299 (0.0101772845 %)
Default replication factor:    3
Average block replication:     3.0837624
Corrupt blocks:                1
Missing replicas:              5 (1.2703163E-5 %)
Number of data-nodes:          1509
Number of racks:               42
```

The filesystem under path '/' is CORRUPT

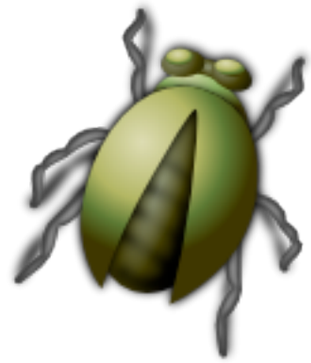
dfsadmin -report Example

- hadoop dfsadmin -report

```
Total raw bytes: 2338785117544448 (2.08 PB)
Remaining raw bytes: 2377132300316700 (216.2 TB)
Used raw bytes: 1538976032374394 (1.37 PB)
% used: 65.8%

Total effective bytes: 0 (0 KB)
Effective replication multiplier: Infinity
-------------------------------------------------
Datanodes available: 1618

Name: 192.168.1.153:50010
Rack: /192.168.1.128
State          : In Service
Total raw bytes: 1959385432064 (1.78 TB)
Remaining raw bytes: 234818330641(218.69 GB)
Used raw bytes: 1313761392777 (1.19 TB)
% used: 67.05%
Last contact: Thu Feb 19 21:57:01 UTC 2009
```



NameNode Status

NameNode 'mynamenode.example.com:8020'

Started:	Wed Feb 04 16:18:50 UTC 2009
Version:	0.18.3-2486615, r
Compiled:	Thu Jan 29 16:43:04 UTC 2009 by hadoopqa
Upgrades:	There are no upgrades in progress.

Browse the filesystem

Cluster Summary

10877074 files and directories, 12860721 blocks = 23737795 total. Heap Size is 13.57 GB / 13.57 GB (100%)

Capacity	:	2.08 PB
DFS Remaining	:	216.08 TB
DFS Used	:	1.37 PB
DFS Used%	:	65.81 %
Live Nodes	:	1403
Dead Nodes	:	215

Live Datanodes : 1403

Node	Last Contact	Admin State	Size (TB)	Used (%)	Used (%)	Remaining (TB)	Blocks
sleepy	2	In Service	1.42	67.19		0.11	25705
dopey	0	In Service	1.43	66.83		0.12	26399
grumpy	0	In Service	1.42	67.46		0.11	25789

The Not So Secret Life of the NameNode

- Manages HDFS Metadata
  - in-memory (Java heap determines size of HDFS!)
  - on-disk

- Image file
  - static version that gets re-read on startup

- Edits file
  - log of changes to the static version since startup
  - Restarting namenode applies edits to the image file

- hadoop-site.xml:

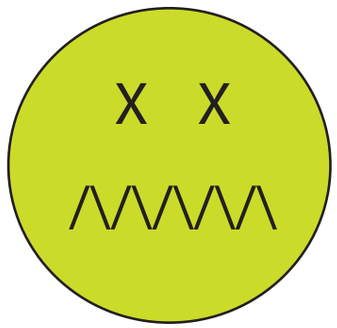
```
<property>
  <name>dfs.name.dir</name>
  <value>/hadoop/var/hdfs/name</value>
</property>
```

NameNode: Your Single Point of Failure

- When NameNode dies, so does HDFS
- In practice, does not happen very often
- Multiple directories can be used for the on-disk image
  - `<value>/hadoop0/var/hdfs/name,/hadoop1/var/hdfs/name</value>`
  - sequentially written
  - 2nd directory on NFS means always having a copy
- Hint: Watch the disk space!
  - Namenode logs
  - image and edits file
  - audit logs (more on that later)

Why NameNodes Fail

- Usually not a crash; brownout
  - Hint: Monitoring
    - Checking for dead process is a fail
    - Must check for service!
- Bugs
  - No, really.
- Hardware
  - Chances are low



- Misconfiguration
  - Not enough Java heap
  - Not enough physical RAM
    - swap=death
- As HDFS approaches full DataNodes cannot write add'l blocks
  - inability to replicate can send NameNode into death spiral
- Users doing bad things

HDFS NameNode Recovery

- When NN dies, bring up namenode on another machine
  - mount image file from NFS
  - create local directory path
  - change config to point to new name node
  - restart HDFS
  - NameNode process will populate local dir path with copy of NFS version

- Hint: Use an A/CNAME with small TTL for namenode in hadoop-site.xml
  - Move the A/CNAME to the new namenode
    - No config changes required on individual nodes
  - For CNAMEs, restart the DataNodes to pick up changes
    - See HADOOP-3988 for details

- But what about the secondary?

Secondary NameNode: Enabling Fast Restarts

- **NOT** High Availability

- Merge the edits file and image file without namenode restart
  - Service is **down** until merge is finished when run on the primary
  - Secondary does this live with no downtime

- Optional, but for sizable grids, this should run
  - 40g edits file will take ~6hrs to process
    - Weeks worth of changes from 800 users on a 5PB HDFS

- Requires the same hardware config as namenode
  - due to some issues with forking, may require more memory
    - swap is fine here..
    - HADOOP-4998 and HADOOP-5059 have some discussion of the issues

Herding Cats... err.. Users

- Major user-created problems
  - Too many metadata operations
  - Too many files
  - Too many blocks

- Namespace quotas (0.18 HADOOP-3187)
  - Limit the number of files per directory
  - hadoop dfsadmin -setQuota # dir1 [dir2 ... dirn]
  - hadoop dfsadmin -clrQuota dir1 [dir2 ... dirn]

- Size quotas (0.19 HADOOP-3938, 0.18 HADOOP-5158)
  - Limit the total space used in a directory
  - hadoop dfsadmin -setspaceQuota # dir1 [dir2 ... dirn]
    - defaults to bytes, but can use abbreviations (e.g., 200g)
  - hadoop dfsadmin -clrspaceQuota dir1 [dir2 ... dirn]

More on Quotas

- Reminder: Directory-based, not User-based
  - /some/directory/path has a limit
  - user allen does not

- No defaults
  - User creation scripts need to set “default” quota

- No config file
  - Store as part of the metadata
  - HDFS must be up; no offline quota management

- Quota Reporting
  - hadoop fs -count -q dir1 [dir2 ...]
  - There is no “give me all quotas on the system” command
    - HADOOP-5290

Trash, The Silent Killer That Users Love

- Recovery of multi-TB files is hard
- hadoop fs -rm / client-side only feature
  - MR, Java API will not use .Trash
- Deleted files sent to HOMEDIR/.Trash/Current
  - “poor man’s snapshot”
  - hadoop-site.xml: fs.trash.interval
    - Number of minutes between cleanings
- Typical scenario:
  - Running out of space
  - Users delete massive amounts of files
  - Still out of space
  - Need to remove files out of trash to reclaim

Hadoop Permission System

- "Inspired" by POSIX and AFS
  - users, groups, world
  - read/write/execute
  - Group inheritance

- User and Group
  - Retrieved from client
  - Output of whoami, id, groups

- hadoop-site.xml: dfs.umask
  - umask used when creating files/dirs
  - **Decimal**, not octal
    - 63 not 077

HADOOP IS NOT SECURE! RUN FOR YOUR LIVES!

- Server never checks client info
- Permission checking is easily circumvented
  - App asks namenode for block #'s that make up file (regardless of read perms)
  - App asks datanode for those blocks

- Strategy 1: Who cares?
- Strategy 2: User/Data Separation
  - Firewall around Hadoop
  - Provision users only on grids with data they can use
  - Trust your users not to break the rules

Audit Logs (HADOOP-3336)

- When, Who, Where, How, What

```
2009-02-27 00:00:00,299 INFO org.apache.hadoop.fs.FSNamesystem.audit:
ugi=allenw,users ip=/192.168.1.100  cmd=create          src=/project/cool/data/
file1  dst=null  perm=allenw:users:rw-------
```

- log4j.properties

```
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
```

Multiple Grids

- Needed for
  - Security
  - Data redundancy

- How separate should they be?
  - Separate user for namenode, datanode, etc, processes?
  - Separate ssh keys?
  - Separate home directories for users?

- Data redundancy
  - Dedicated loading machines
  - Copying data between grids

distcp - distributed copy

- hadoop distcp [flags] URL [URL ...] URL
  - submits a map/reduce job to copy directories/files
- hadoop distcp hdfs://nn1:8020/full/path hdfs://nn2:8020/another/path
  - copies block by block using Hadoop RPC
  - very fast
- Important flags
  - p = preserve various attributes, except modification time
  - i = ignore failures
  - log = write to a log file
  - m = number of copies (maps)
    - very easy to flood network if too many maps are used!
  - filelimit / sizelimit = limits the quantity of data to be copied
    - Another safety check against eating all bandwidth

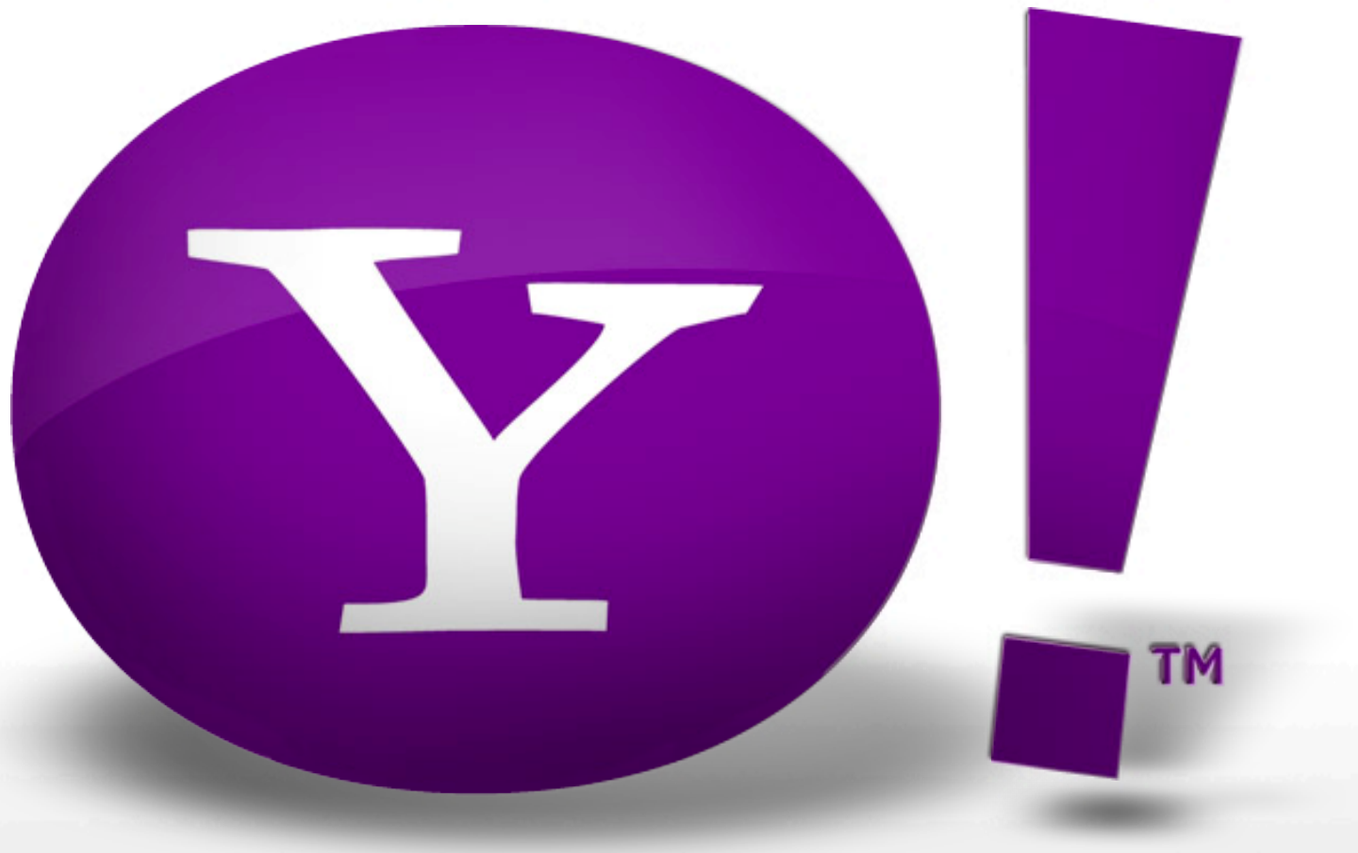
Copying Data Between Hadoop Versions

- hdfs method uses Hadoop RPC
  - versions of Hadoop must match!

- hadoop distcp hftp://nn1:50070/path/to/a/file hdfs://nn2:8020/another/path
  - file-level copy
  - slow
  - fairly version independent
  - must run on **destination** cluster
  - cannot write via hftp

- Uses a single port for copying

Q & A



YAHOO!