DiskSetup

Setting up Disks for Hadoop

Here are some recommendations for setting up disks in a Hadoop cluster. What we have here is anecdotal -hard evidence is very welcome, and everyone should expect a bit of trial and error work.

Key Points

Goals for a Hadoop cluster are normally massive amounts of data with high I/O bandwidth. Your MapReduce jobs may be IO bound or CPU/Memory bound -if you know which one is more important (effectively how many CPU cycles/RAM MB used per Map or Reduce), you can make better decisions.

Hardware

You don't need RAID disk controllers for Hadoop Data Node, as it copies data across multiple machines instead. This increase the likelihood that there is a free task slot near that data, and if the servers are on different PSUs and switches, eliminates some more points of failure in the data center.

While the Hadoop Name Node and Secondary Name Node can write to a list of drive locations, they will stop functioning if it can not write to ALL the locations. In this case a mirrored RAID is a good idea for higher availability.

Having lots of disks per server gives you more raw IO bandwidth than having one or two big disks. If you have enough that different tasks can be using different disks for input and output, disk seeking is minimized, which is one of the big disk performance killers. That said: more disks have a higher power budget; if you are power limited, you may want fewer but larger disks.

Configuring Hadoop

Pass a list of disks to the `dfs.data.dir` parameter, Hadoop will use all of the disks that are available. When one goes offline it is taken out of consideration. Hadoop does not check for the disk coming back -it assumes it is "gone".

How to limit Data node's disk usage?

Use dfs.datanode.du.reserved configuration value in $HADOOP_HOME/conf/hdfs-site.xml for limiting disk usage.

```
<property>
  <name>dfs.datanode.du.reserved</name>
  <!-- cluster variant -->
  <value>182400</value>
  <description>Reserved space in bytes per volume. Always leave this much space free for non dfs use.
  </description>
</property>
```

Logging

- The environment variable, `HADOOP_LOG_DIR` sets the directory Hadoop logs to.
- the log4j.properties file in your hadoop configuration dir controls logging in more detail
- Don't log to the root directory, as having a machine that does not boot because the logs are overflowing can be inconvenient.
- Have a plan to clean up log output, otherwise jobs that log too much to the console will fill up log directories.
- Get your developers to use the commons-logging APIs in their MapReduce code, so that you can turn logging up or down without recompiling the code. They can run in debug mode on their test machines, you can run at WARN level in production.
- Some JVMs (JRockit) seem to log more. Tune your Log4j settings for your JVM, and only capture the stuff you really want.

Do not keep stuff under /tmp

1. Hadoop defaults to keeping things under `/tmp` so that you can play with Hadoop without filling up your disk. This is dangerous in a production cluster, as any automated cleanup cron job will eventually delete stuff in `/tmp`, at which point your Hadoop cluster is in trouble.
2. You will need cron job to clean stuff in `/tmp` up eventually. Plan for it.
3. Configure Hadoop to store stuff in stable locations, preferably off that root disk.
   1. Java stores the info for `jps` under `/tmp/hsperfdata_${user}` -after the cleanup jps won't work. Have your script leave those directories alone, or get used to using `ps -ef | grep java` to find Java processes instead.

Underlying File System Options

If mount the disks as `noatime`, then the file access times aren't written back; this speeds up reads. There is also `relatime`, which stores some access time information, but is not as slow as the classic atime attribute. Remember that any access time information kept by Hadoop is independent of the atime attribute of individual blocks, so Hadoop does not care what your settings are here. If you are mounting disks purely for Hadoop, use `noatime`.

Formatting and tuning options are important. Using `tunefs` to set the reserve to zero percent can save you over 25 GigaBytes on a 1 TeraByte disk. Also the underlying file system is going to have many large files, you can get more space by lowering the number of inodes at format time.

Ext3

Yahoo! has publicly stated they use ext3. Regardless of the merits of the filesystem, that means that HDFS-on-ext3 has been publicly tested at a bigger scale than any other underlying filesystem that we know of.

XFS

From Bryan on the core-user list on 19 May 2009:

We use XFS for our data drives, and we've had somewhat mixed results. One of the biggest pros is that XFS has more free space than ext3, even with the reserved space settings turned all the way to 0. Another is that you can format a 1TB drive as XFS in about 0 seconds, versus minutes for ext3. This makes it really fast to kickstart our worker nodes.

We have seen some weird stuff happen though when machines run out of memory, apparently because the XFS driver does something odd with kernel memory. When this happens, we end up having to do some fscking before we can get that node back online.

As far as outright performance, I actually *did* do some tests of xfs vs ext3 performance on our cluster. If you just look at a single machine's local disk speed, you can write and read noticeably faster when using XFS instead of ext3. However, the reality is that this extra disk performance won't have much of an effect on your overall job completion performance, since you will find yourself network bottlenecked well in advance of even ext3's performance.

The long and short of it is that we use XFS to speed up our new machine deployment, and that's it.

Ext4

The Ext4 Linux filesystem has delayed allocation of data which makes it handle unplanned server shutdowns/power outages less well than classic ext3. Consider turning off the delalloc option in /etc/fstab unless you trust your UPS.