HiveAws

= Hive and Amazon Web Services =

Background

This document explores the different ways of leveraging Hive on Amazon Web Services - namely S3, EC2 and Elastic Map-Reduce.

Hadoop already has a long tradition of being run on EC2 and S3. These are well documented in the links below which are a must read:

- Hadoop and S3
- Amazon and EC2

The second document also has pointers on how to get started using EC2 and S3. For people who are new to S3 - there's a few helpful notes in S3 for n00bs section below. The rest of the documentation below assumes that the reader can launch a hadoop cluster in EC2, copy files into and out of S3 and run some simple Hadoop jobs.

Introduction to Hive and AWS

There are three separate questions to consider when running Hive on AWS:

1. Where to run the Hive CLI from and store the metastore db (that contains table and schema definitions).
2. How to define Hive tables over existing datasets (potentially those that are already in S3)
3. How to dispatch Hive queries (which are all executed using one or more map-reduce programs) to a Hadoop cluster running in EC2.

We walk you through the choices involved here and show some practical case studies that contain detailed setup and configuration instructions.

Running the Hive CLI

The CLI takes in Hive queries, compiles them into a plan (commonly, but not always, consisting of map-reduce jobs) and then submits them to a Hadoop Cluster. While it depends on Hadoop libraries for this purpose - it is otherwise relatively independent of the Hadoop cluster itself. For this reason the CLI can be run from any node that has a Hive distribution, a Hadoop distribution, a Java Runtime Engine. It can submit jobs to any compatible hadoop cluster (whose version matches that of the Hadoop libraries that Hive is using) that it can connect to. The Hive CLI also needs to access table metadata. By default this is persisted by Hive via an embedded Derby database into a folder named metastore_db on the local file system (however state can be persisted in any database - including remote mysql instances).

There are two choices on where to run the Hive CLI from:

1. Run Hive CLI from within EC2 - the Hadoop master node being the obvious choice. There are several problems with this approach:
    - Lack of comprehensive AMIs that bundle different versions of Hive and Hadoop distributions (and the difficulty in doing so considering the large number of such combinations). Cloudera provides some AMIs that bundle Hive with Hadoop - although the choice in terms of Hive and Hadoop versions may be restricted.
    - Any required map-reduce scripts may also need to be copied to the master/Hive node.
    - If the default Derby database is used - then one has to think about persisting state beyond the lifetime of one hadoop cluster. S3 is an obvious choice - but the user must restore and backup Hive metadata at the launch and termination of the Hadoop cluster.

2. Run Hive CLI remotely from outside EC2. In this case, the user installs a Hive distribution on a personal workstation, - the main trick with this option is connecting to the Hadoop cluster - both for submitting jobs and for reading and writing files to HDFS. The section on Running jobs from a remote machine details how this can be done. Case Study 1 goes into the setup for this in more detail. This option solves the problems mentioned above:

- Stock Hadoop AMIs can be used. The user can run any version of Hive on their workstation, launch a Hadoop cluster with the desired Hadoop version etc. on EC2 and start running queries.
- Map-reduce scripts are automatically pushed by Hive into Hadoop's distributed cache at job submission time and do not need to be copied to the Hadoop machines.
- Hive Metadata can be stored on local disk painlessly.

However - the one downside of Option 2 is that jar files are copied over to the Hadoop cluster for each map-reduce job. This can cause high latency in job submission as well as incur some AWS network transmission costs. Option 1 seems suitable for advanced users who have figured out a stable Hadoop and Hive (and potentially external libraries) configuration that works for them and can create a new AMI with the same.

Loading Data into Hive Tables

It is useful to go over the main storage choices for Hadoop/EC2 environment:

- S3 is an excellent place to store data for the long term. There are a couple of choices on how S3 can be used:
    - Data can be either stored as files within S3 using tools like aws and s3curl as detailed in S3 for n00bs section. This suffers from the restriction of 5G limit on file size in S3. But the nice thing is that there are probably scores of tools that can help in copying/replicating data to S3 in this manner. Hadoop is able to read/write such files using the S3N filesystem.
    - Alternatively Hadoop provides a block based file system using S3 as a backing store. This does not suffer from the 5G max file size restriction. However - Hadoop utilities and libraries must be used for reading/writing such files.

- HDFS instance on the local drives of the machines in the Hadoop cluster. The lifetime of this is restricted to that of the Hadoop instance - hence this is not suitable for long lived data. However it should provide data that can be accessed much faster and hence is a good choice for intermediate/tmp data.

Considering these factors, the following makes sense in terms of Hive tables:

1. For long-lived tables, use S3 based storage mechanisms
2. For intermediate data and tmp tables, use HDFS

Case Study 1 shows you how to achieve such an arrangement using the S3N filesystem.

If the user is running Hive CLI from their personal workstation - they can also use Hive's 'load data local' commands as a convenient alternative (to dfs commands) to copy data from their local filesystems (accessible from their workstation) into tables defined over either HDFS or S3.

Submitting jobs to a Hadoop cluster

This applies particularly when Hive CLI is run remotely. A single Hive CLI session can switch across different hadoop clusters (especially as clusters are bought up and terminated). Only two configuration variables:

- `fs.default.name`
- `mapred.job.tracker`
  need to be changed to point the CLI from one Hadoop cluster to another. Beware though that tables stored in previous HDFS instance will not be accessible as the CLI switches from one cluster to another. Again - more details can be found in Case Study 1.

Case Studies

1. Querying files in S3 using EC2, Hive and Hadoop

Appendix

<<Anchor(S3n00b)>>

S3 for n00bs

One of the things useful to understand is how S3 is used as a file system normally. Each S3 bucket can be considered as a root of a File System. Different files within this filesystem become objects stored in S3 - where the path name of the file (path components joined with '/') become the S3 key within the bucket and file contents become the value. Different tools like [S3Fox|https:--addons.mozilla.org-en-US-firefox-addon-3247] and native S3 !FileSystem in Hadoop (s3n) show a directory structure that's implied by the common prefixes found in the keys. Not all tools are able to create an empty directory. In particular - S3Fox does (by creating a empty key representing the directory). Other popular tools like aws, s3cmd and s3curl provide convenient ways of accessing S3 from the command line - but don't have the capability of creating empty directories.