Compacted Aggregated Logs

12/9/14 -- Robert Kanter

Problem

Turning on log aggregation allows users to easily store container logs in HDFS and subsequently view them in the YARN web UIs from a central place. Currently, there is a separate log file for each Node Manager. This can be a problem for HDFS if you have a cluster with many nodes as you'll slowly start accumulating many (possibly small) files per YARN application. The current "solution" for this problem is to configure YARN (actually the JHS) to automatically delete these files after some amount of time.

Proposal

We can provide a better solution by compacting the aggregated log files for each node into a single log file per application.

The per-node aggregated log files currently reside all in the same directory, per application and are in a binary format (`AggregatedLogFormat`, which is essentially a `TFile`). Each file consists of some header metadata (ACLs, owner, and version), followed by key-value pairs. Each of the keys is a Container ID and the values are that Container's logs (stdout, stderr, and syslog are all here) and their lengths.

We can write a program that that takes the aggregated logs for an application as input and outputs them to a single compacted file. Finding a specific container ID in an `AggregatedLogFormat` file is somewhat expensive (you have to search through the file), so we should have an index to handle the potentially large size of the compacted log file. This index would be a list of the container IDs, their offset in the file, and the length of their logs. Then, given the index information, we can quickly find the right location in the compacted file.

The format of the single file can essentially just be a "dump" of the log information from each per-node aggregated log file, all put together. The index file can include the few pieces of metadata that the per-node aggregated log files had (i.e. owner, acls), plus the index itself.

This program is pretty independent and can be implemented as a Service so we can put pretty much anywhere. The service would be responsible for running this program when all of an application's logs have been uploaded to HDFS. Ideally, we don't want this program to run on the same node all the time because that local datanode will have one replica for every compacted log file. We also need a way ensure that the program only runs once all the aggregated logs are finished uploading. To handle both of these situations, we can have the Service that runs the program live as an auxiliary service in each NodeManager. The RM, can have a "companion" Service that checks if the aggregated logs are done uploading by counting the number of log files in HDFS vs the number of nodes assigned to that application; once it determines they are done, the companion Service can tell an arbitrary NodeManager

(randomly, round-robin, etc) to run the compaction. This will ensure that the replicas for the compacted log files are more evenly distributed around the cluster.

The JHS/ATS will also need to be updated to be able to understand the new format of the single compacted aggregated log file for serving in the Web UI; though this should be fairly minimal work because we can reuse parsing code from the `AggregatedLogFormat`. They would try to load the compacted log file first; if they don't exist yet, they can load the aggregated log files.

We can include some configuration options, such as:

- enable/disable compaction
    - (would automatically be disabled if log aggregation is disabled)
- location for compacted log files
- how long to retain compacted log files