Concatable Aggregated Logs

YARN-2942

4/2/15 -- 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 concatenating 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 the logs (stdout, stderr, and syslog are all here) of that container and their lengths.

Unfortunately, the `AggregatedLogFormat` is not append-friendly, meaning that we can't concatenate multiple aggregated log files together. As such, we need to use a new format that is append-friendly for the concatenation to result in a valid file. The new format, `ConcatableAggregatedLogFormat`, stores all of the logs in one file similar to the `AggregatedLogFormat`, but also has a separate index file. This allows the log files to be concatenated together and the index file appended to (the index file is small enough that we can use append to write the additional index entries).

The JHS currently finds the logs for a specific Container ID in an aggregated log file by searching through it -- this is somewhat inefficient and would be even worse for the larger concatenated file. Luckily, now that we have a proper index in the new format, the JHS can simply look in the index file. Here's what the format of the index file would roughly look like:

```
|-------------------------------------|
|Version                              |
|ACLs                                 |
|Owner                                |
|ContainerID,logtype,offset,length    |
|...                                  |
|-------------------------------------|
```

There isn't currently a practical way to know when all nodes used by an Application are done uploading their aggregated logs; this is especially true in the case of a long running application (which will be addressed in YARN-2548). As a result, we need to have a distributed approach to concatenating the logs, with each NodeManager concatenating its own log at the right time. ZooKeeper/Curator will be used to store one lock per Application to ensure that only one NodeManager can concatenate at a time. A new `LogConcatenationService` will be added to the NodeManager that will look for completed aggregated log files that it "owns" and concatenate them. This service would essentially do the following per application in a loop:

1. Try Acquire lock
    a. Yes → Am I the first?
        i. Yes → simply rename the file from <hostname> to <app-id>
        ii. No → concat my log file and append my index
    b. No → Skip during this run
2. Release lock

The concatable aggregated log files would still get initially written using their hostname filenames (like the aggregated log files do today), but the final single concated log file will be named with the Application Id; hence why the first NM simply has to rename its file.

Because of this bottleneck at combining, the JHS (and presumably eventually ATS) code for retrieving logs will be updated to check the Application Id concated aggregated log file's index for a Container ID and fallback to looking at the node's concated aggregated log file if it can't be found. This will ensure that the user can get all of the logs as soon as they are uploaded to HDFS, without having to wait for the concatenation process to finish.

The work can be split up into 4 steps:

1. Create the `ConcatableAggregatedLogFormat` Reader and Writer
2. Modify the NM to be able to write using either the `ConcatableAggregatedLogFormat` or the `AggregatedLogFormat`
3. Create the `LogConcatenationService`
4. Modify the JHS and CLI to be able to read using either the `ConcatableAggregatedLogFormat` or the `AggregatedLogFormat`

We will include some configuration options, such as:

- Enable/Disable the new format and concatenation
- ZooKeeper/Curator configs
- Frequency of concatenation service

Deleting old logs will be handled automatically by the existing `AggregatedLogDeletionService` because the new log format and concatenation reuses the same directory setup as the original aggregated logs.

Backwards Compatibility

The Yarn and MapReduce reading components (e.g. JHS, NM, etc) can be made to support both the `AggregatedLogFormat` and the `ConcatableAggregatedLogFormat`. However, anyone using the `AggregatedLogFormat` directly won't be able to interact with logs if we switch the format to the `ConcatableAggregatedLogFormat`. As a result, we'll have to add a new configuration property to enable the new format with the concat behavior, with the default being the old format with no concatenation.

Follow-up Work

- Logs from long running services are not considered here at this time, though the new format is designed to be accommodating pending future work in YARN-2548. For now, they should be left alone by the concatenation code (step 3).