Combined Aggregated Logs

YARN-2942

5/5/15 -- 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 per YARN application. 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 and blocks. To work around this, users are forced to delete the log files (through JHS configs) frequently even though disk space itself is not a concern.

Proposal

Combining the per-node-aggregated log files into a single log file per application would allow users to keep more logs without stressing out the NameNode.

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.

The `AggregatedLogFormat` is not "append-friendly". That is, we can't simply append new data to the end of an aggregated log file for creating the single log file per application. So we introduce a new format, `CombinedAggregatedLogFormat`. This format can be similar to that of the aggregated log files; we can essentially just "dump" the information from the aggregated log file into a "combined aggregated log file", which gives us the advantage of reusing most of the log parsing code that already exists when reading and writing the file. 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 combined file. The solution to this is to have a second file which stores the index, which can simply be a mapping of the container IDs with their offsets and lengths in the combined log file; we can also put the header metadata in here as well. Here's what the format of the index file would roughly look like:

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

YARN-1376 and related JIRAs recently added the log aggregation status of an application to the NM heartbeat; this allows the RM to know the log aggregation status for every application. Armed with this information, the RM is now the perfect place to add a new Service to handle combining the logs. Unlike the previous versions of this proposal, we no longer need to use ZooKeeper to coordinate NMs. The new `RMAggregatedLogsCombinationService` can create a thread pool which looks for applications which have successfully finished log aggregation, and combine their logs.

For tracking, we need to modify the `LogAggregationStatus` enum. This enum is used by each NM to tell the RM the aggregation status. The RM then reuses the same enum to rollup the aggregation status on a per-app basis. For the new combining code, we're interested in the per-app rolledup status.

Current Values	**New Values**
`DISABLED`	`DISABLED`
`NOT_START`	`NOT_START`
`RUNNING`	`AGGREGATION_RUNNING`
`SUCCEEDED`	`AGGREGATION_SUCCEEEDED`
`FAILED`	`AGGREGATION_FAILED`
`TIME_OUT`	`TIME_OUT`
N/A	`COMBINING_RUNNING`
N/A	`COMBINING_SUCCEEDED`
N/A	`COMBINING_FAILED`

* The `COMBINING_` values would only be used for the per-app status.

The changes should be fairly straightforward. As an example, here's what the full workflow would look like:
`NOT_START → AGGREGATION_RUNNING → AGGREGATION_SUCCEEDED → COMBINING_RUNNING → COMBINING_SUCCEEDED`
As there has yet been no Hadoop release with `LogAggregationStatus` (fix version is 2.8.0), changing these enums shouldn't be a problem.

Once the `AGGREGATION_SUCEEDED` status is reached, all logs will be available to the user (as before). The combining procedure will be such that the original aggregated log files are only deleted once they are appended to the combined aggregated log file; the JHS and yarn

CLI can be updated to transparently handle looking up logs in either file (i.e. if the Container ID cannot be found in the combined aggregated log, look in the aggregated log).

Here's an overview of the procedure that the `RMAggregatedLogsCombinationService` will follow:

1. Get the `LogAggregationStatus` for each Application
2. If the status for Application_X is `AGGREGATION_SUCCEEDED`
    a. Set status to `COMBINING_RUNNING` and start combining the log
        i. Delete the original aggregated log file once each node's aggregated log file is consumed
    b. When the logs are all combined, set status to `COMBINING_SUCCEEDED`
3. Repeat

When combining a log for a container, the `CombinedAggregatedLogFormat` Writer first appends the log data, and then writes the Container ID, offset, and length to the index; it then repeats this for each container in an aggregated log file. This way, if the RM dies, it will be as if that partially appended log does not exist (because it's not in the index). When the RM comes back, it can resume combining the logs, and will re-append that container's logs. This leaves essentially "dead space" in the combined aggregated log file because in HDFS, we can't remove it. The Reader looks at the index file, so it will ignore this space because no index entries point to it. The `RMAggregatedLogsCombinationService` will know that it needs to resume combining the logs for an application by checking for any applications with `COMBINING_RUNNING` on startup. And it will know which container to do next by reading the index file.

The combined aggregated log files would be located in the same place as the aggregated log files. Some advantages of this include: these directories are already created by the `LogAggregationService` with the correct permissions, no need to duplicate the directory structure, and the `AggregatedLogDeletionService` will handle cleaning up combined aggregated log files with no additional code changes.

The work can be split up into 3 tasks:

1. Create the `CombinedAggregatedLogFormat` Reader and Writer
2. Create the `RMAggregatedLogsCombinationService` and modify `LogAggregationStatus` and related code
3. Modify the JHS and CLI to be able to read using either the `CombinedAggregatedLogFormat` or the `AggregatedLogFormat` with fallback

We will include some configuration options, such as:

- Enable/Disable combining
- Number of threads in the `RMAggregatedLogsCombinationService`'s thread pool

Follow-up Work

- Logs from long running services (which are uploaded on a per container basis instead of per node) are not considered here at this time (YARN-2548 can update it to handle them). For now, they should be left alone by the `RMAggregatedLogsCombinationService`.