[Proposal] New Log Aggregation Format

Problem

The underlying log format for the aggregated logs in YARN is T-File. Recently, we have seen some limitations:

- Performance issues
    - To get log meta, we have to search/scan the whole log file from the beginning.
    - To get a specific log, we have to search/scan the whole log file from the beginning.
- Is not appendable
    - For long running service, we have to create a new aggregated log file when partial log aggregation happens.
    - Force that yarn.nodemanager.log-aggregation.roll-monitoring-interval-seconds should be at least 3600
    - Have an extra configuration: yarn.nodemanager.log-aggregation.num-log-files-per-app to limit the number of the aggregated log files.

Requirement

The new log format should satisfy the following requirements:

- Fast access/fetch the log meta
    - filter by application id
    - filter by container id
    - filter by filename
- Fast fetch the log
    - filter by application id
    - filter by container id
    - filter by filename pattern and/or container id
- Support old format
- Support multiple format (customized file format)
- Download the compressed files
- (Lower priority), Fast search logs in a large cluster. (thousands of nodes).

Solution

The new aggregated log format will contain:

- Separated compressed log content
- Aggregated MetaInfo

    - Metainfo contains
        - Version
        - User
        - Acls
        - NodeId
        - Compress-Method Name
        - A list of logMeta
            - Remote log file name
            - Log upload Time Stamp
            - A map of FileLogMeta
                - ContainerId
                - File Name
                - File Size
                - File Compressed-Size
                - Start index
                - Last modification time

The overall layout of the aggregated log will be:

Compressed log content1 Compressed log content2...MetaInfo Length-of-meta

For long running service and partial log aggregation, we will only have one log file. All logs would be appended together. For the MetaInfo, We would aggregate all previous meta with the latest meta.

Failure Recovery

After the log aggregation, we would only have one big aggregated log file even for the long running applications. For the partial log aggregation scenario, we would append the partial log to previous aggregated logs with the aggregated log meta. But this requires that the append operation is atomic otherwise we would get corrupted file. This is the approach that we will use in our newly designed Log aggregation file format.

- For write operation:
    - Alway create an simple index file. The index file only contains one line record which is the end index of the latest successful aggregated log file.
    - Remove the index file if the log aggregation in cycle is successful.
- For read operation:
    - Check whether the index file exists.
        - If not, it is not the corrupted file. Then read the log meta from the bottom of the log file

    - If it exists, the aggregated log file has been corrupted. We would read the index from the index file. For the corrupted log file, We only need to parse the valid content from 0 to the index.

Configurations:

yarn.log-aggregation.file-formats	Define how many file formats we will support. To support backward compatibility, T-file format will be always added automatically.
yarn.log-aggregation.file-format.%s.class	Define the class for each file formats
yarn.log-aggregation.%s.remote-app-log-dir	Different file format could have different remote log dir
yarn.log-aggregation.%s.remote-app-log-dir-suffix	Different file format could have different suffix

To support backward compatibility, we have

- All previous log aggregation-related configurations are still valid.
- T-File format will always be added to the format list automatically.

Additionally,

- When do log write, the writer will use the first configured log format to write.
    - If we set yarn.log-aggregation.file-formats as “Index-file-format, azure-file-format”, we will support three file format: index-file-format, azure-file-format and T-file-format(add automatically)
    - The writer will use index-file-format to write log
    - We could read logs which is index-file-format, azure-file-format or T-file-format.
- Each file format should have different combination of remote-app-log-dir and suffix.
- Different file format could define their own aggregated log directory.
- When do log read, we would use this combination to figure out the correct file format.