Exploration of HDFS performance after solving basics bugs

Louis-Claude Canon

April 5, 2012

1 Changes and bugs solving

The first issue was related to the \texttt{hdfsRead} function, which only read the first block (problem when the block size is lower than the data to transmit). Using \texttt{hdfsPread} instead by specifying a zero offset solves the problem.

The second issue is related to the \texttt{NotReplicatedYetException} exception. The HOD user guide indicates that it is related to the fact that the namenode takes too much time to contact the datanode. The datanode has therefore been put on a separate nodes to limit interference (hence, there is two servers). Additionally, the file is removed after each test. However, the problem subsists (some warning from time to time, with a sleep of half a second).

The following parameters were set by default:

\begin{itemize}
  \item \texttt{io.bytes.per.checksum} 1024 B.
  \item \texttt{fs.checkpoint.size} 128 MiB.
  \item \texttt{dfs.block.size} 32 MiB.
  \item \texttt{dfs.replication} 2 (to check that it has an impact).
\end{itemize}

Sequence file seems to consist of the concatenation of small files using a key/value scheme (where the value may be a binary).

2 Experiment failures

Of the 292 experiments, 12 have failed. It seems to be related to the inability of HDFS to deal with a lot of blocks (most of the block size are closed phenomenas: some of the data are sometime not written to each replica (here one), whereas it may the number of written data which is not the same as requested.

The list of settings which resulted in failures is given by

\begin{verbatim}
cat exploit2bis/\* | grep "^\*1" | awk '{ print $4 }' | xargs cat > exploit2bis_failure
\end{verbatim}

\begin{verbatim}
> F <- read.table("exploit2bis_failure")
> dim(F)

[1] 12 5
\end{verbatim}

\begin{verbatim}
> names(F) <- c("servers", "clients", "strip.size", "file.size",
+    "consecutive")
> plot(F[", file.size"], F[", strip.size"], log = "xy")
> summary(F[", file.size"]/F[", strip.size"])
\end{verbatim}
Most failed cases involve block sizes below 10 kiB and a number of blocks greater than 10000.

```r
> D <- read.table("explo1bis_summary")
> dim(D)

[1] 280 13

> names(D) <- c("servers", "clients", "strip_size", "file_size",
+                  "consecutive", "MPI_rank", "file_size", "consecutive", "write_start",
+                  "write_end", "write_close", "read_start", "read_end")
> length(which(D[, "file_size"]/D[, "strip_size"] < 10000))/dim(D)[1]

[1] 0.9964286
```

In most working cases, there is less than 10000 blocks. Then, it must be related to the limit of HDFS in terms of blocks management.
3 Dissection of the protocol

Based on the log information, we can assume the following facts:

• the client contacts the namenode for each block before sending it to the datanode
• no log are kept on the namenode about the read requests, maybe it is not involved
• the delete operation results in the namenode sending delete requests to the datanode per group of 100. The datanode schedule them and delete 100 blocks each 3 seconds.

Therefore, in order to ensure that each datanode cleans its blocks before each operation, the block size should probably be bigger than \((100 \text{ MB/s})/(100 \text{ blocks/3 seconds}) = 3 \text{ MB}\).

4 Preliminary analysis

```r
> duration <- cbind(D[, c("strip_size", "file_size")], write = D[, "write_end"] - D[, "write_start"], closing = D[, "write_close"] - D[, "write_end"], complete = D[, "write_close"] - D[, "write_start"], read = D[, "read_end"] - D[, "read_start"])
> pairs(~log10(strip_size) + log10(file_size) + log10(write) + log10(closing) + log10(complete) + log10(read), data = duration, cex = 0.1)
> min(duration)
[1] 8.703e-05
```
The observations are similar: several thresholds when writing data without syncing, many noise, strip size seems to have little effect.

5 Write operation

```r
> plot(duration[, "file_size"], duration[, "write"], log = "xy",
+     cex = 0.1)
> segments(x0 = 1, x1 = 1e+09, y0 = 3e-04)
> max(duration[duration[, "write"] < 3e-04, "file_size"])
```

[1] 1006

```r
> min(duration[duration[, "write"] > 3e-04, "file_size"])
```

[1] 1038

```r
> segments(x0 = 1, x1 = 1e+09, y0 = 0.003)
> max(duration[duration[, "write"] < 0.003, "file_size"])
```

[1] 61275
The first threshold is indeed related to the `io.bytes.per.checksum` value. However, the second threshold does not seem to be impacted by either `fs.checkpoint.size` or `dfs.block.size`.

```r
> plot(duration[, "file_size"], duration[, "complete"], log = "xy", + cex = 0.1)
> (latency <- median(duration[duration[, "file_size"] < 1e+05, + "complete"]))
[1] 0.03360749
> curve(latency + x/1e+08, add = TRUE)
```
There is only two cases where the complete bandwidth is used. It may be related to the sleeps that occur when the previous warning is issued. Need to restart the experiments with larger block sizes (4 MiB) to allow the system to clean the file when they are deleted.

6 Read operation

```r
> plot(duration[, "file_size"], duration[, "read"], log = "xy",
+       cex = 0.1)
> (latency <- median(duration[duration[, "file_size"] < 1e+05,
+       "read")])
[1] 0.002974985
> curve(latency + x/1e+08, add = TRUE)
```
Performance seems to be better, but still with many noise.

7 Conclusion

Redo experiments with default values and a block size greater than or equal to 4 MiB.