Exploration of HDFS performance
Louis-Claude Canon
April 1, 2012

1 Setup

Hadoop 1.0.1 has been installed on top of a Grid5000 squeeze-x64-nfs system. The previous benchmarking application has been adapted to connect to and exchange data with the HDFS servers (MPICH 1.4.1p1 was used). In addition to the size of the data to transmit, the program can also select the buffer size, the strip size and the replication. The JNI C interface was used to access HDFS functions.

Measurements were done on the paradent cluster (latency around 0.180 ms, bandwidth of 943 Mb/s).

2 List of parameters of interest and default values

local.cache.size 10 GB (for out of band data).

io.bytes.per.checksum Number of bytes per checksum (512 B).

io.file.buffer.size Size of the buffer for read/write operations on sequence files (4096 B).

dfs.block.size Strip size, must be a multiple of io.bytes.per.checksum (64 MiB).

dfs.replication Replication of each block (3).

Sequence file is still not clear. Does a FSDataOutputStream results in such a file?

In the experiments, one server hosts the namenode, a datanode and the secondernamenode. The client sends and receives data with a replication of 1 and the default parameter values. The logarithm of the block and file sizes follow a uniform distribution between 1 B and 1 GB (the block size being a multiple of 512).

3 Experiment failures

Of the 88 experiments, 36 have failed. There seems to be two distinct 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

```
wc explo1/result_* | grep "^[ \]* 1" | awk '{ print $4 }' | xargs cat > explo1_failure
```

```
F <- read.table("explo1_failure")
> dim(F)
[1] 36 5
> names(F) <- c("servers", "clients", "strip_size", "file_size",
+ "consecutive")
> plot(F[, "file_size"], F[, "strip_size"], log = "xy")
> curve(1 * x, add = TRUE)
> F[F[, "file_size"] < F[, "strip_size"], ]
```
All failures occur when the strip size is lower than the file size (except for one case). However, for the last measures, strange things were happening. Also, there is a higher concentration of failures for large files.

### 4 Preliminary analysis

```r
> D <- read.table("explo1_summary")
> dim(D)

[1] 52 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")
> 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"])
```
The minimum measured duration is about 100 µs, which accounts for the overhead of JNI (negligible in comparison to any data transfer).

4.1 Write

The relation between the file size and the write time suggests there is two modes depending on the data size for small messages (around 500 B, maybe related to \texttt{io.bytes.per.checksum}).

```r
> plot(duration[, "file_size"], duration[, "write"], log = "xy")
> max(duration[, "write"] < 3e-04, "file_size")

[1] 430
> min(duration[, "write"] > 3e-04, "file_size")
```
However, the complete time, which includes the flushing of the data, seems to be unaffected and is close to 30 ms. This time is mostly spend in the closing operation. Note that for large messages, this time is negligible as data must be automatically flushed when some threshold is reached (around 50 MB). If this is related to the default `dfs.block.size`, then it means that the given strip size is ignored. Changing it in the global configuration file will confirm it. But then, how to explain the first issue for large file that occurs only when the file size is greater than the strip size? Another parameter whose default value is 64 MiB is `fs.checkpoint.size`.

```r
> plot(duration[, "file_size"], duration[, "complete"], log = "xy")
> median(duration[duration[, "file_size"] < 1e+05, "complete"])
[1] 0.03347659
> max(duration[duration[, "write"] < 0.01, "file_size"])
[1] 44496
> min(duration[duration[, "write"] > 0.01, "file_size"])
[1] 77027
```
4.2 Read

The read operation seems to be similar to the write operation, except there is both less noise and extreme values.

```r
> plot(duration[, "file_size"], duration[, "read"], log = "xy")
> median(duration[duration[, "file_size"] < 1e+05, "read"])

[1] 0.00290706
```
5 Conclusion

After solving the first two described problems, two parameters could be changed by default to confirm the previous observations:

io.bytes.per.checksum 1024 B.

fs.checkpoint.size 128 MiB.

dfs.block.size 32 MiB (to see which one has an effect if any).