C API to HDFS: libhdfs

Content-Type text/html; utf-8

Table of contents

1 C API to HDFS: libhdfs....2
2 The APIs....2
3 A sample program....2
4 How to link with the library....2
5 Common problems....3
6 libhdfs is thread safe....3

1. C API to HDFS: libhdfs

libhdfs is a JNI based C api for Hadoop's DFS. It provides C apis to a subset of the HDFS APIs to manipulate DFS files and the filesystem. libhdfs is part of the hadoop distribution and comes pre-compiled in ${HADOOP_HOME}/libhdfs/libhdfs.so .

2. The APIs

The libhdfs APIs are a subset of: hadoop fs APIs.

The header file for libhdfs describes each API in detail and is available in ${HADOOP_HOME}/src/c++/libhdfs/hdfs.h

3. A sample program

```
#include "hdfs.h"

int main(int argc, char **argv) {

    hdfsFS fs = hdfsConnect("default", 0);
    const char* writePath = "/tmp/testfile.txt";
    hdfsFile writeFile = hdfsOpenFile(fs, writePath, O_WRONLY|O_CREAT, 0,
0, 0);
    if(!writeFile) {
          fprintf(stderr, "Failed to open %s for writing!\n", writePath);
          exit(-1);
    }
    char* buffer = "Hello, World!";
    tSize num_written_bytes = hdfsWrite(fs, writeFile, (void*)buffer,
strlen(buffer)+1);
    if (hdfsFlush(fs, writeFile)) {
           fprintf(stderr, "Failed to 'flush' %s\n", writePath);
          exit(-1);
    }
   hdfsCloseFile(fs, writeFile);
}
```

4. How to link with the library

See the Makefile for hdfs_test.c in the libhdfs source directory (${HADOOP_HOME}/src/c++/libhdfs/Makefile) or something like: gcc above_sample.c -I${HADOOP_HOME}/src/c++/libhdfs -L${HADOOP_HOME}/libhdfs -lhdfs -o above_sample

5. Common problems

The most common problem is the CLASSPATH is not set properly when calling a program that uses libhdfs. Make sure you set it to all the hadoop jars needed to run Hadoop itself. Currently, there is no way to programmatically generate the classpath, but a good bet is to include all the jar files in ${HADOOP_HOME} and ${HADOOP_HOME}/lib as well as the right configuration directory containing hadoop-site.xml

6. libhdfs is thread safe

Concurrency and Hadoop FS "handles" - the hadoop FS implementation includes a FS handle cache which caches based on the URI of the namenode along with the user connecting. So, all calls to hdfsConnect will return the same handle but calls to hdfsConnectAsUser with different users will return different handles. But, since HDFS client handles are completely thread safe, this has no bearing on concurrency.

Concurrency and libhdfs/JNI - the libhdfs calls to JNI should always be creating thread local storage, so (in theory), libhdfs should be as thread safe as the underlying calls to the Hadoop FS.