cloudera
Ask Bigger Questions

A Native Client for the Hadoop Distributed Filesystem

by Colin P. McCabe

About Me

- I work on HDFS and related storage technologies at Cloudera.
- Committer on the HDFS and Hadoop projects.
- Previously worked on the Ceph distributed filesystem

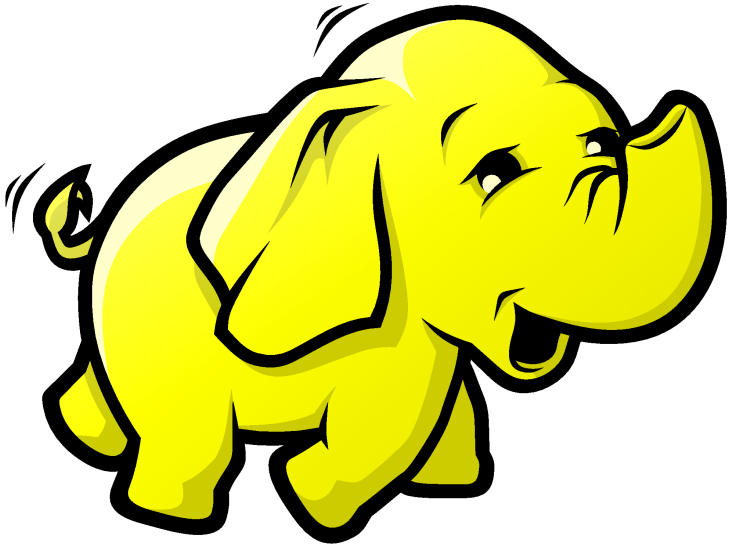
Roadmap

- Motivations
- Design goals
- Progress
- Challenges
- Future work

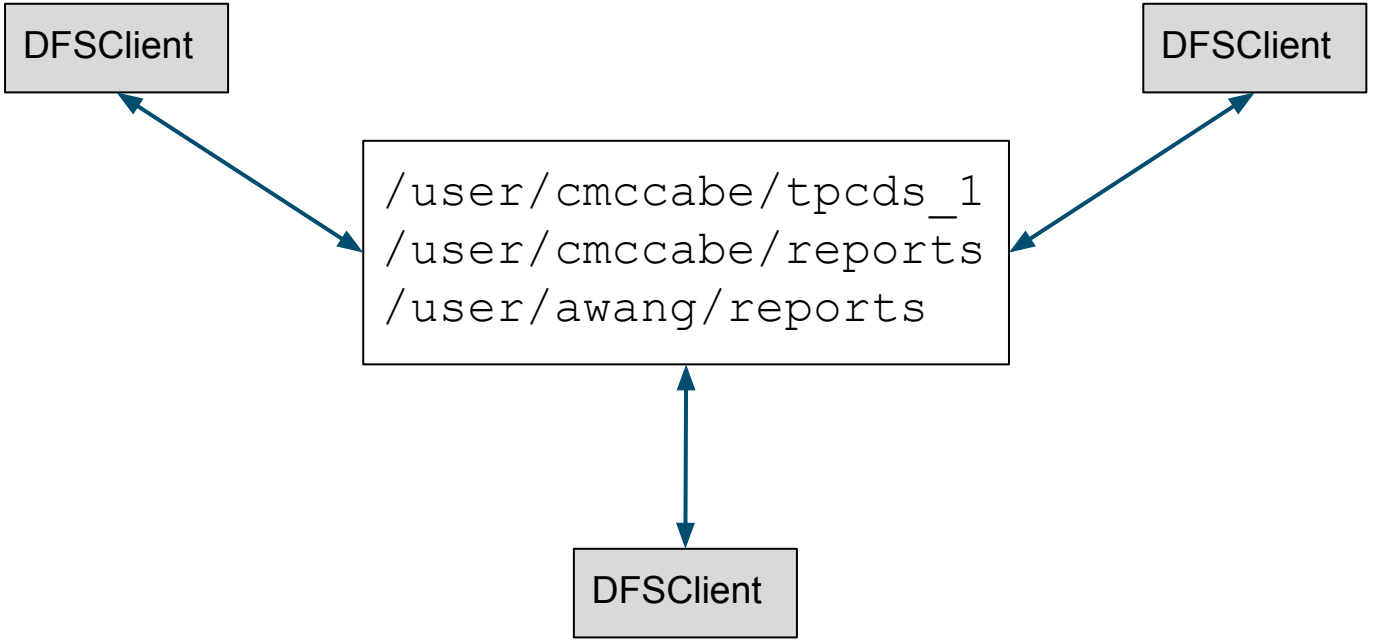
The Hadoop Distributed Filesystem

HDFS is the most popular storage system for Hadoop.

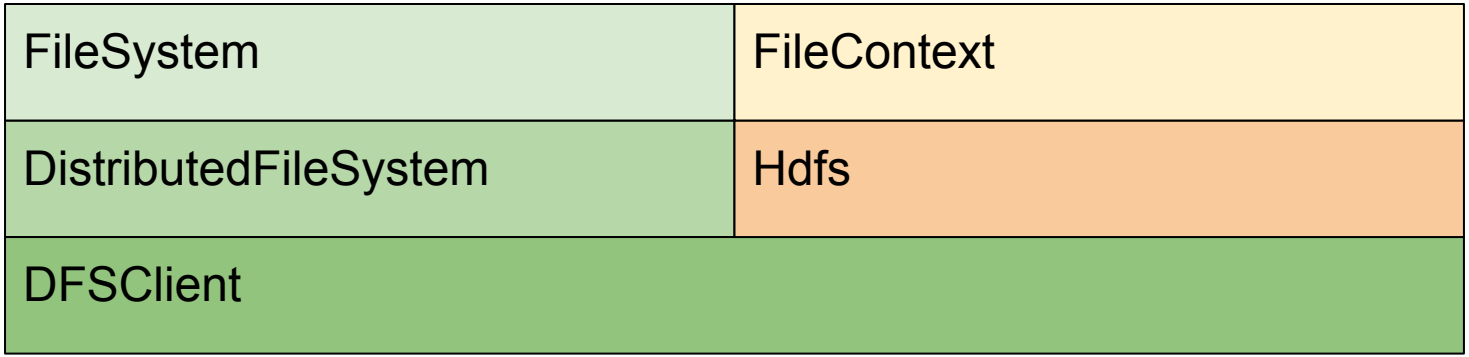
Based on the concepts of robustness, fault-tolerance, and bringing the computation to the data



HDFS Provides a Shared Namespace



HDFS Java Client

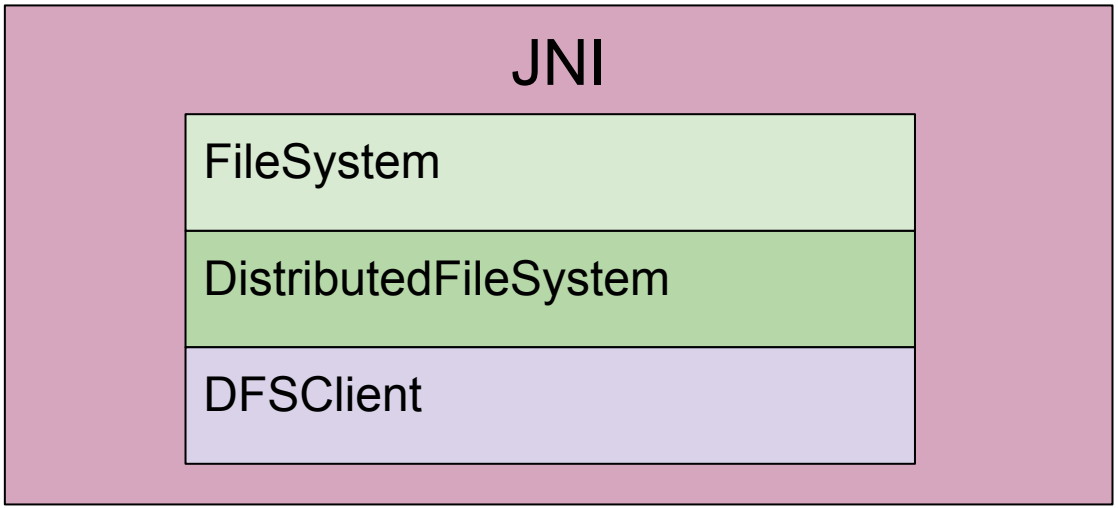


Accessing HDFS from a non-JVM Language

- The HDFS client is implemented in Java.
- How can C and C++ (or even Python) code access HDFS?
  - libhdfs
  - webhdfs
  - libwebhdfs

libhdfs

- A thin JNI wrapper around the code in Hadoop's FileSystem.java.



libhdfs

- Difficult to debug because it's JNI (no gdb, signals)
- Requires HDFS jars to be deployed on every client
- Long startup time (must start up a JVM.)
- Misnamed: can access other FSes besides HDFS
- But: still the most widely used way of interfacing native code with HDFS.

webhdfs

- Exposes a REST API for HDFS
- Creating HTTP requests is convenient in many languages (but not so much in C / C++...)
- Many optimizations not possible (short-circuit, mmap, etc.)

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
...
```

libwebhdfs

- A C library that makes webhdfs requests
- More convenient for C and C++ programs than creating HTTP requests manually
- Doesn’t require the Hadoop jars installed on every machine
- Not as mature as libhdfs
- Same performance disadvantages as raw webhdfs

hdfs.h

- The C and C++ API for libwebhdfs and libhdfs
- Backwards-compatible

```
/**
 * Create an HDFS builder.
 *
 * @return The HDFS builder, or NULL on error.
 */
struct hdfsBuilder *hdfsNewBuilder(void);

/**
 * Set the username to use when connecting to the HDFS cluster.
 *
 * @param bld The HDFS builder
 * @param userName The user name.  The string will be shallow-copied.
 */
void hdfsBuilderSetUserName(struct hdfsBuilder *bld, const char *userName);

...
```

Fully Native Library

- Quick startup time (no need to start a JVM)
- Doesn’t require Hadoop jars installed everywhere
- Can support short-circuit and zero-copy optimizations
- None of the debugging issues with JNI

Roadmap

- Motivations
- Design goals
- Progress
- Challenges
- Future work

Goals

- Usability
  - Create a client usable from C, C++, Objective C, and other non-JVM languages
  - Support the existing libhdfs.h API, which many programs have been written against
  - Cross-platform support: Linux, Windows, other UNIXes...

Goals

- Deployability
  - Don’t require the Hadoop jars to be installed to function
  - Read existing HDFS configuration XMLs
  - Provide a shared library that can be linked into existing applications

Goals

- Deployability
  - Ideally, libhdfs3 can linked against existing code which uses libhdfs or libwebhdfs and “just work”

Goals

- Maintainability
  - Create a client which the Hadoop community can maintain in the future
  - Good code quality

Non-Goals

- Server implementations
  - libhdfs3 is only a client library; it doesn't include implementations of the HDFS servers.
- Completely replacing other access methods
  - Other access methods have different strengths
  - libhdfs can access non-HDFS filesystems and make use of new Java client features more quickly than a native client.

Roadmap

- Motivations
- Design goals
- Progress
- Challenges
- Future work

Progress

- HADOOP-10388 branch
  - Started by myself, Abraham Elmarek, Binglin Chan, and Wenwu Peng.
  - A C client with implementations of all NameNode RPCs
  - Has an async Hadoop RPCv9 client

Progress

- HDFS-6994 branch
  - Based on code contributed by Zhanwei Wang and other folks at Pivotal
  - Reviews and additional code by myself, Haohui Mai, and others
  - C++ code with support for Kerberos, NameNode HA, and the client read and write paths.

Merger

- HDFS-6994 branch supersedes HADOOP-10388 branch
- Move over any useful code or features from HADOOP-10388, and merge our efforts.

Cleanup

- Remove pre-RPCv9 code
- Add appropriate Apache headers
- Use standard Hadoop file name conventions
- Fix some naming issues and calling conventions
- CMake cleanup

Integration

- Use the protobuf files from the Hadoop source rather than a bundled set of files
- Integrate into source tree
- (Still to do) integrate with the Maven build cycle
- (Still to do) Fix configuration parsing issues

Implementation

- Implement InputStream functionality
- Implement file system functionality
- (in progress) Implement hdfs.h interface
- (in progress) Implement OutputStream functionality
- (in progress) Implement testing

Dependency Fixes

- Remove category X library dependencies and extra dependencies that might cause problems
- Remove non-threadsafe library dependencies

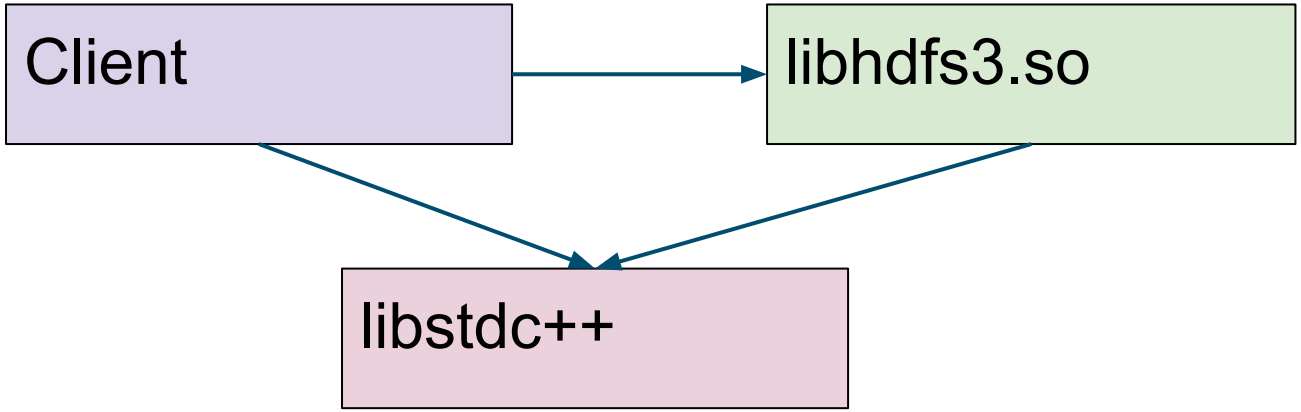
Roadmap

- Motivations
- Design goals
- Progress
- Challenges
- Future work

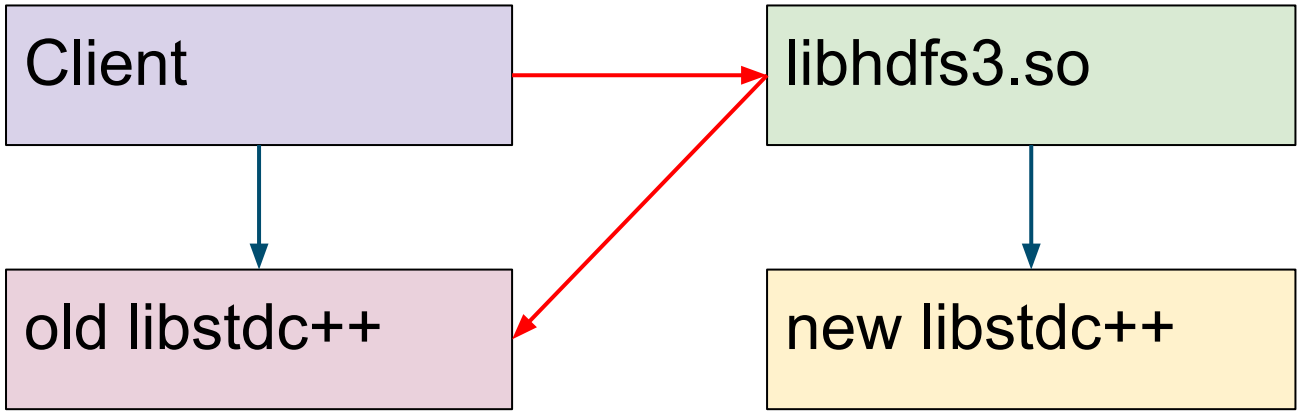
C++11

- Many nice features
- std::thread, std::chrono, std::shared_ptr
  - Widely used
  - Cross platform
- But… not supported everywhere yet.
- Compatibility challenges

C++11 Issues



C++11 Issues



Potential Solutions to C++11 Conundrums

- Avoid C++11 features
  - Use tr1, or our own code
- Link statically against libstdc++ or libc++ (very difficult)
- Use Boost

Boost

- … has implementations of std::thread, std::chrono, and many other C++11 libraries
- Can work with C++98
- Has its own compatibility issues
- A heavyweight dependency
- Use as “stepping stone” to C++11?

New Client Features

- The Java HDFS client is always getting new features
  - Encryption at rest support
  - “Hedged” reads
  - etc…
- The native client can reimplement these features, but it will probably always lag behind when doing so

Client Fallback

- Can the HDFS client fall back from libhdfs3 to libhdfs, if it requires a feature that libhdfs3 doesn't have?
- Some work on this in HDFS-7041: add a library that can forward hdfs.h calls to any other library implementation

Open Issues

- Library name
  - Does “libhdfs3” sound too much like version 3.0 of the unrelated libhdfs library?

Open Issues

- Code style
  - We need a uniform C++ code style across Hadoop… so far, we don't have that
  - Exceptions vs. return codes
  - Google Coding style?

Roadmap

- Motivations
- Design goals
- Progress
- Challenges
- Future work

Future Work

- Windows support
  - Need to wrap POSIX functions
  - UCS-2 issues on Windows?

Future Work

- C++ API
  - should be shared between libhdfs, libwebhdfs, and libhdfs3
  - Must avoid exposing library internals

Conclusion

- libhdfs3 will soon be a viable option for native clients to interface with HDFS
- Many advantages over existing methods
- Thanks to everyone who has helped with this

Thanks for Listening!

http://www.cloudera.com/careers