Welcome

This is the source code for CodFS described in our paper presented in USENIX FAST 2014. The system is tested on Ubuntu 12.04.2 64-bit with GCC 4.6.3.

Release 1.0.2 (11-11-2014) Code cleanup and documentation update

Installation

These are the required libraries that users need to download separately. Brackets denote the package names in Debian and Ubuntu platforms. Users can use apt-get to install the required libraries.

- build-essential (build-essential)
- scons (scons)
- boost libraries version 1.5x (libboost-dev, libboost-program-options-dev, libboost-thread-dev, libboost-filesystem-dev)
- FUSE (libfuse-dev)
- openssl (libssl-dev)
- pkg-config (pkg-config)

For Debian and Ubuntu users:

```
sudo apt-get install build-essential scons libboost-dev libboost-program-options-dev libboost-t
hread-dev libboost-filesystem-dev libfuse-dev libssl-dev pkg-config
```

The following libraries have to be compiled and installed manually:

Google Protocol Buffers

```
$ cd lib/protobuf-2.4.1
$ ./configure; make clean; make
$ sudo make install
$ sudo ln -s /usr/local/lib/libprotobuf.so.7 /usr/lib/libprotobuf.so.7
$ sudo ln -s /usr/local/lib/libprotoc.so.7 /usr/lib/libprotoc.so.7
```

MongoDB C++ Driver (only required for MDS)

```
$ cd lib/mongo
$ sudo scons install
```

Preparation

Compile

The program can be compiled using Linux make. Running make on the CodFS root will automatically compile all CodFS components.

```
$ make
```

For optimal performance, turn off debug mode by editing the following flags in Makefile:

```
OPTIMIZE := -O3
EXTRA CFLAGS := -std=c++0x -DDEBUG=0 -D FILE OFFSET BITS=64
```

Setting up MongoDB

We suggest hosting the MongoDB on the MDS for optimal performance.

- Set up MongoDB environment

On the MongoDB host, create the database parent directory

```
$ sudo mkdir -p /data/db
```

Change the MongoDB address "[mongodb*ip]:[mongodb*port]" in "mongo-codfs.js" if necessary

```
db = connect("localhost:27017/codfs");
```

Change the default MongoDB password in "mongo-codfs.js" for security reasons

```
"pwd": "cf8b7fa0b1bcd277da8217f04f498bd0"
```

Note

- "pwd" should be an MD5 hash, which can be generated by running `echo 'NEWPASSWORD' | md5sum` in Linux

- Change to the directory containing the MongoDB binary

  ```
  $ cd mongo/bin
  ```

- Start MongoDB daemon

  ```
  $ sudo ./mongod --fork --dbpath /data/db --logpath /var/log/mongodb.log --logappend
  ```

- Import MongoDB Setting

  ```
  $ ./mongo ../../mongo-codfs.js
  ```

Adjusting XML Settings

- Copy one of the two set of example XML files from the directory to the CodFS root (same level as the executables) For standalone setup, `cp example_xml/standalone/* .` For distributed setup, `cp example_xml/distributed/* .`

  - example_xml/standalone/
    - Example XML files for a standalone setup which runs all components on localhost (127.0.0.1)
  - example_xml/distributed/
    - Example XML files for a distributed setup
      - MONITOR on 192.168.0.11:53000
      - MDS on 192.168.0.12:50000
      - MongoDB daemon on 192.168.0.12:27017

- Adjust the following essential parameters in the XML files according to the cluster setup:

  - `mdsconfig.xml`
    - MDS listen port
    - MongoDB IP address, listen port, password
  - `osdconfig.xml`
    - Update scheme (FO=0, FL=1, PL=2, PLR=3) and reserved space size
  - `monitorconfig.xml`
    - MONITOR listen port
  - `clientconfig.xml`
    - Coding scheme (e.g., RS, RDP) and coding settings (e.g. n, k)
  - `common.xml`
    - MDS IP address and listen port
    - MONITOR IP address and listen port

- Refer to comments in the XML files for other individual settings (e.g., OSD capacity)

Running CodFS

Server-side

Copy the corresponding executable (MONITOR, MDS or OSD) and all the XML files to each server

Start the components in the following order

- MONITOR

  ```
  $ ./MONITOR
  ```

- MDS

  ```
  $ ./MDS
  ```

- Start OSD one-by-one

  ```
  $ ./OSD [component_id] [network_interface]
  ```

Note

- Assign a `component_id` that is unique from MDS, MONITOR, other OSDs and clients
- Use "ifconfig" to check the name of the network interface (e.g., eth0). If multiple interfaces exist, select the one that can be reached by other components

Client-side

On the client machine, create two directories

```
$ mkdir fusedir mountdir
```

Mount the FUSE volume with the following options

```
$ ./CLIENT FUSE -o big writes,large read,noatime -f mountdir [component id]
```

You can now access CodFS through `mountdir`. The following command copies a file into the cluster and outputs the time spent:

```
$ time cp testfile mountdir/
```

Note

- Assign a `component_id` that is unique from MDS, MONITOR, OSDs and other clients

Recovery

Automatic Recovery

CodFS does not automatically recover by default. To enable this features, uncomment the line `#define TRIGGER_RECOVERY` in `src/common/define.hh`

Note

- Adjust the sensitivity of recovery trigger by changing `SleepPeriod`, `DeadPeriod` and `UpdatePeriod` in `monitorconfig.xml`

Manual Recovery

Manual recovery is useful in benchmarks.

To simulate failure, kill the OSD process on one of the OSD hosts

```
$ pkill -9 OSD
```

To force failure detection and recovery, send a signal to the MONITOR process on the MONITOR host

```
$ pkill -USR1 MONITOR
```

Benchmarks

Test seq. read/write throughput using dd

After mounting the FUSE volume, test seq. write throughput by:

```
$ dd if=/dev/zero of=mountdir/ddtest count=256 bs=16M conv=fdatasync
```

Drop page cache:

```
$ sync && sudo echo 3 > /proc/sys/vm/drop caches
```

Test seq. read throughput by:

```
$ dd if=mountdir/ddtest of=/dev/null count=256 bs=16M
```

Test random write throughput using IOzone

Download latest IOzone tarball from http://www.iozone.org/.

Compile IOzone from source

```
$ tar xf iozone3_424.tar
$ cd iozone3_424/src/current
$ make linux-AMD64
```

Copy the IOzone executable to CodFS root, and execute the following:

```
$ ./iozone -Ra -+n -i0 -i2 -s 4g -r 128k -e -c -w -f mountdir/iozonetest
```

This will create a 4GB file and perform random read/write with 128KB record size.

Contact

Patrick P. C. Lee (http://www.cse.cuhk.edu.hk/~pclee)

Formatted using GitHub Flavored Markdown.