

EPLog

Prerequisite

EPLog is developed and runs on a Linux machine, e.g. Ubuntu Server 12.04LTS. Install the following libraries and GCC-4.6.3 or higher to compile EPLog:

- Build-essential (*apt*: build-essential; *yum*: make gcc gcc-c++)
- Boost development libraries for threads (*apt*: libboost-thread-dev; *yum*: boost-devel)
- OpenSSL development libraries (*apt*: libssl-dev; *yum*: openssl-devel)

EPLog also leverages some libraries from the open-source community:

- GF-complete
- Jerasure
- Threadpool
- libcuckoo

Test EPLog

Test cases

The example program performs the following series of tests:

- Create a new file
- Overwrite the whole new file
- Append another new file
- Modify the appended file
 - intra-segment partial modification
 - inter-segment partial modification

The example program compares and reports the test results and any difference between data written and read.

Run the example program

1. Compile EPLog
 1. \$ cd trunk/
 2. \$ make
2. Prepare the environment
 1. \$ cd bin/
 2. \$ for i in {1..8}; do dd if=/dev/zero of=./disk\${i} bs=10M count=1; done
 3. \$./example

Settings

- Array of disks
 1. trunk/src/server/unit_test/example.cc, line 56-65
 - Each disk is represented by the data structure `DiskInfo`
 - E.g. data disk with id 0, device path `disk1`, and 10MB capacity
 - E.g. log disk with id 6, device path `disk6`, and 10MB capacity

```
1 // List the disks available in the system
2 DiskInfo disk1 (0, "disk1", 1048576ULL * 10);
3 DiskInfo disk2 (1, "disk2", 1048576ULL * 10);
4 DiskInfo disk3 (2, "disk3", 1048576ULL * 10);
5 DiskInfo disk4 (3, "disk4", 1048576ULL * 10);
6 DiskInfo disk5 (4, "disk5", 1048576ULL * 10);
7 DiskInfo disk6 (5, "disk6", 1048576ULL * 10);
8 DiskInfo disk7 (6, "disk7", 1048576ULL * 10, true);
9 DiskInfo disk8 (7, "disk8", 1048576ULL * 10, true);
10 vector<DiskInfo> v_diskInfo {disk1,disk2,disk3,disk4,disk5,disk6
    ,disk7,disk8};
```

- Encoding scheme
 1. trunk/src/server/unit_test/example.cc, line 68-73
 - Each coding scheme is represented by the data structure `CodeSetting`
 - Two coding schemes set, one for data segments, one for log segments
 - E.g. Cauchy Reed-Solomon Codes $(n,k,w)=(6,4,8)$ for data segments, and Cauchy Reed-Solomon

Codes (n,k,w)=(8,6,8) for log segments

```
1 CodeSetting codeSetting (6,4,8,CAUCHY_CODING);  
2 CodeSetting codeLogSetting (8,6,8,CAUCHY_CODING,true);  
3 vector<CodeSetting> codeSettingList;  
4 codeSettingList.push_back (codeSetting);  
5 codeSettingList.push_back (codeLogSetting);
```

2. trunk/bin/config.ini, line 10-14

- Number of data chunks in data segments and log segments respectively
- E.g. 4 and 6 data chunks per data segment and log segment respectively

```
1 [coding]  
2 ; number of data blocks per data segment  
3 numBlockPerSegment = 4  
4 ; number of data blocks per log segment  
5 numBlockPerLogSegment = 6
```

3. trunk/bin/config.ini, line 1-5

- Chunk Size
- E.g. 4KB per page and 1 page per chunk.

```
1 [ssd]  
2 ; page size  
3 pageSize = 4096  
4 ; chunk size in unit of pages  
5 numPagePerBlock = 1
```

Documentation

- Functions of EPLog modules
 - Internals
 - SyncMod: Controlling the generation of parities committed to disk
 - SegmentMetaDataMod: Managing segment metadata
 - RaidMod: Encoding/Decoding segments, performing parity commit and recovery
 - FileMetaDataMod: Managing file metadata
 - LogMod: Managing stripe buffers (for new writes) and device buffers (for update requests)
 - CacheMod: Managing cached chunks read from disks
 - Interfaces
 - StorageMod: Exporting block interface for read/writes
 - KVMod: Exporting key-value interface for read/writes
- More details
 - Refer to inline comments in the source file; OR
 - Generate documentation using doxygen in root-level folder (with Doxyfile): `$ doxygen`

Contact

[Helen H.W. Chan](#))

Publications

- [Yongkun Li](#), [Helen H. W. Chan](#), [Patrick P. C. Lee](#), and [Yinlong Xu](#). "Elastic Parity Logging for SSD RAID Arrays." *DSN* 2016

Acknowledgments

EPLog uses open source libraries Jerasure (Rev. 2.0) and GF-Complete developed by [Prof. James S. Plank](#). EPLog also uses [libcuckoo](#) to realize the key-value interface.

The prototype is an extension of the storage system framework built by [Jeremy Chan](#) and [Qin Ding](#).

Project Page

This repository serves as an archive of EPLog prototype. Please Visit the [project page](#) for more details of this project.