This source code is a simple implementation to evaluate the *correlation-aware stripe organization* (CASO) in erasure-coded storage systems. Please refer to the conference paper: *Correlation-Aware Stripe Organization for Efficient Writes in Erasure-Coded Storage Systems* (in Proceedings of 36th IEEE International Symposium on Reliable Distributed Systems (SRDS 2017)) for more details.

In what follows, we will describe how to install and run CASO code on Ubuntu. Before running the code, please make the following preparations first.

Preparations

1: ensure that the necessary libraries and compile tools are installed, including gcc, make, and the AIO libraries.

```
$ sudo apt-get install gcc make libaio-dev libaio1
```

2: change the default configurations in "config.h", including the erasure code you wish to use.

```
// ======================== Fill the erasure coding parameters ====================
/* parameters of erasure codes */
#define erasure_k 4      // the number of data chunks of a stripe in erasure codes
#define erasure_m 2      // the number of parity chunks of a stripe in erasure codes
#define num_lg 2         // the number of local groups in a stripe, it should be divisible by erasure_m and erasure_k
#define lg_prty_num 1    // the number of local parity in a local group
#define erasure_w 8      // the value of w to configure the arthmetic on the finite field
#define block_size 4096 // assume that the block size is 4KB
// ============================================================================
```

3: If you want to run evaluations on your testbed, please do the following two things before running the codes.

- In the "config.h" : assign the number of the disks to the global variable "num_disks"

```
// ======================== Fill the disk info ====================
/* parameters of testbed */
#define num_disks 15     // number of disks used in our testbed experiment.
// ================================================================
```

- In the "general.c" : specify the directories of the files for evaluating the parallel I/O accesses to the global variable "disk_array"

```
/* ====== Fill the information of the files on different disks for testbed evaluations ======*/
char *disk_array[num_disks]={"/dev/sde","/dev/sdf","/dev/sdg","/dev/sdh","/dev/sdi",
                             "/dev/sdj","/dev/sdk","/dev/sdl","/dev/sdm","/dev/sdn",
                             "/dev/sdo","/dev/sdp","/dev/sdq","/dev/sdr", "/dev/sds"};
/* ============================================================================================*/
```

An example of running CASO code

- extract the files from caso-1.0.0.tar

```
$ tar zxvf caso-1.0.0.tar
$ cd caso-1.0.0/
$ export CASO_HOME=$(pwd)
```

- generate the object files of Jerasure and the executable files in CASO

```
$ cd ${CASO_HOME}
$ make
```

- **Numerical test**: run the CASO test for the trace wdev_3 to perform the numerical test for partial stripe writes and degraded reads.

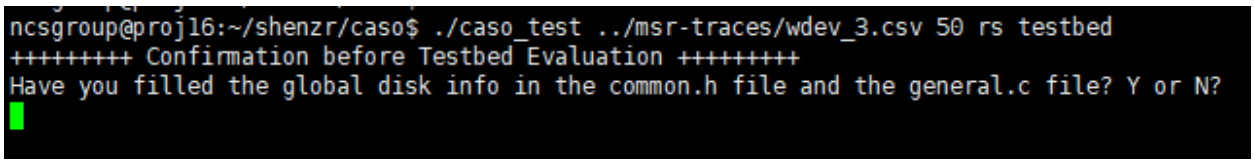
```
$ ./caso_test example_trace/wdev_3.csv 50 rs numeric
```

  The command indicates that we will evaluate CASO when deployed over RS code for the numerical test. In this test, the evaluated trace file is wdev_3 and the analysis ratio is 50%.

- **Testbed test**: run the CASO test for the trace wdev_3 to evaluate the time as well as the numerical test on partial stripe writes and degraded reads.

```
./caso_test example_trace/wdev_3.csv 50 rs testbed
```

  CASO will confirm that you have filled the information of disks for testbed evaluations in the common.h and the general.c before running the test. You should input Y before running the test.



If you have any question, please feel free to contact me (zhirong.shen2601@gmail.com).