UniIO Performance Scripts

install

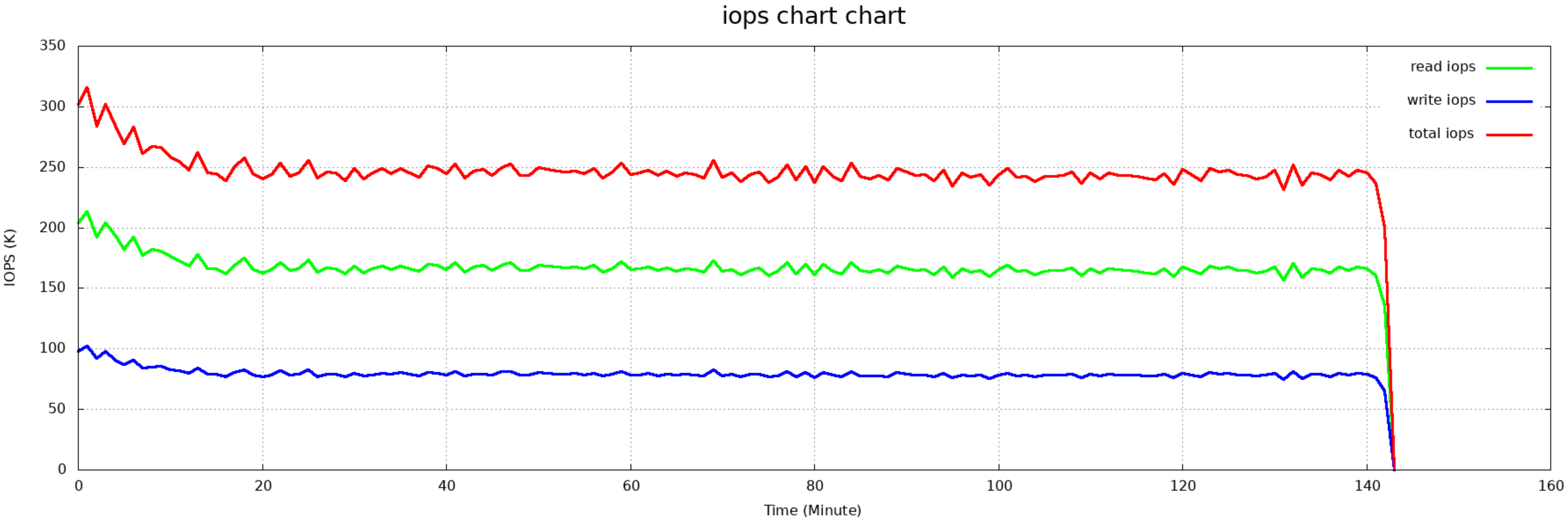
```
$ git clone https://github.com/fred-chen/uio_scripts.git
$ tree uio_scripts/
uio_scripts/
├── client
│   └── plotfio.sh
└── server
    ├── collect_cpu.sh
    ├── counterana.py
    ├── init_backend.sh
    ├── init_cluster.sh
    └── renice_iothreads.sh
```

client scripts

client/plotfio.sh

```
□□□□□□□ fio □□□□□□□□□□□□□□□□□fio□□□job□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□IOPS□SLAT, CLAT, LAT□
□□□
# client/plotfio.sh -h
usage: plotfio.sh <logname> [-t iops|clat|slat|lat] [--title chart_title] [-k|--keep]
options:
  -t: type of plots, can be one of: iops, clat, slat, lat.
  -k: keep temp files.
examples:
  plotfio.sh log/82rw*iops* -t iops  # plot iops chart for logs that the path match 'log/82rw*iops*'
```

IOPS □□□□□□



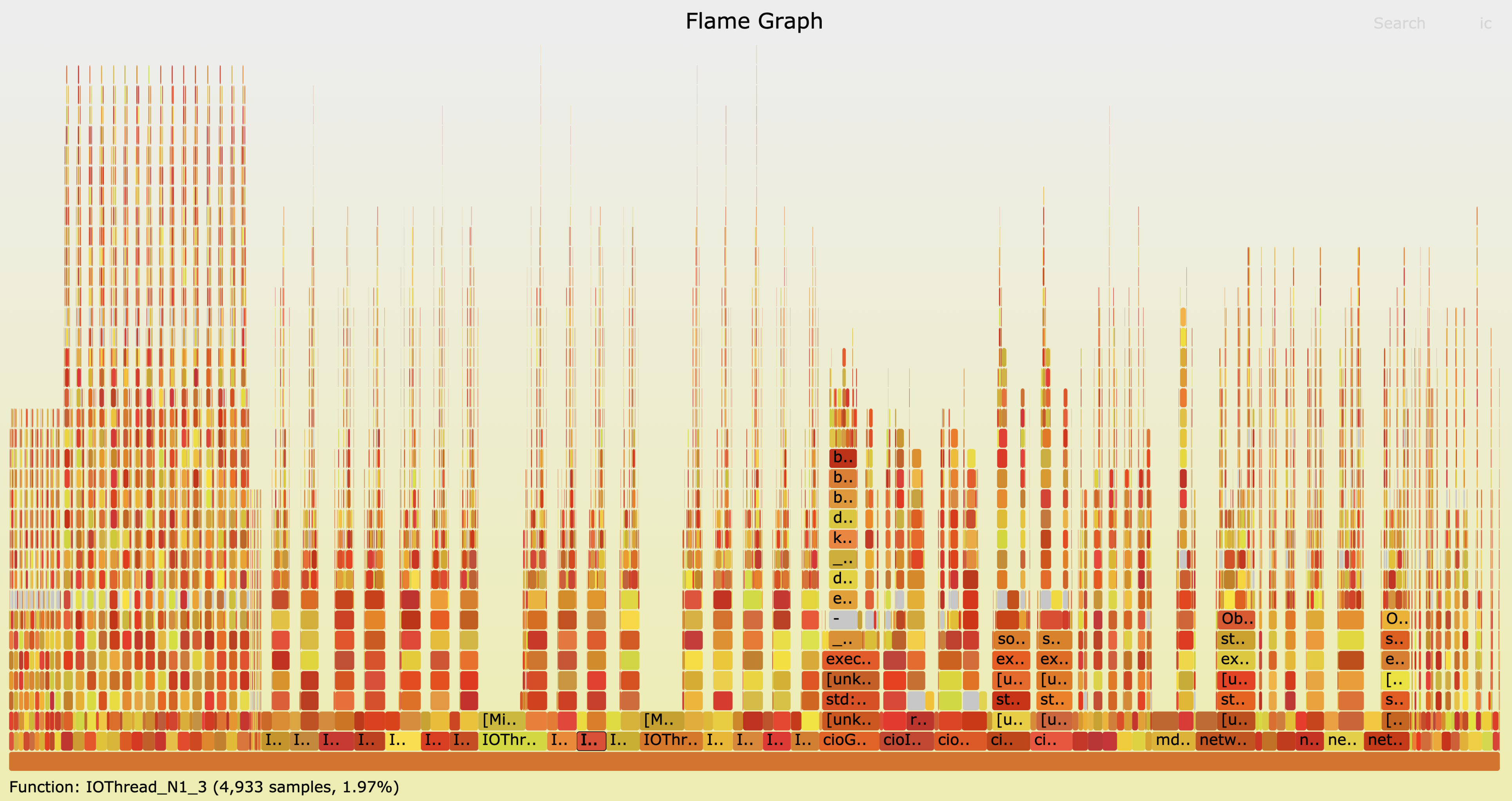
server scripts

server/collect_cpu.sh

```
□□□□□□ linux bcc/eBPF □□ oncpu□offcpu□wakeup□offwakup □□□□□□□□□□□□□□□□□□□□□ SVG □□□□
□□□□□□
```

```
  kernel version >4.8
  eBPF enabled with kernel
  bcc installed
  FlameGraph installed and located in ../FlameGraph
□□□
# server/collect_cpu.sh -h
usage: collect_cpu.sh [process_name] [-w prefix] [-t time] [-g oncpu|offcpu|wakeup|offwakeup] [-x exclude_stack] [-k]
options:
  -k:    keep temp files.
examples:
  collect_cpu.sh                 # gather all types of cpu data for 60 seconds and generate flame graphs. prefix 'this'
  collect_cpu.sh cio_array       # gather all types of cpu data of process 'cio_array' for 60 seconds
                                 # and generate flame graphs. prefix 'this'.
  collect_cpu.sh -w 82rw -t 30 -g oncpu     # gather oncpu data for 30 seconds
                                            # and generate flame graphs. prefix '82rw'
```

> oncpu SVG □□□□□□



server/counterana.py

```

□□□□□ 'arrayctl counters' □□□□□□ UniIO □□□□□□□□□□□□□□□□□
1. □□ counter □□□□:
   □□□□□□□□□□□□□□□□ counter □□□□□□□□□□□□□□□□
   * 'NOCHANGE'  □□□□□□□□□□□counter□□□counter□□□□□□□□□□□□□disable□□□□□□□□□□bug□
   * 'UP'       □□□□□counter□□□counter□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
   * 'DOWN'      □□□□□counter□□□counter□UP□□□counter□□□□□□□□□□□□□□□□□□□□□□□□□
   * 'SPIKES'    □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□cio_array□□□□□□□□□□□
   * 'FLAT'      □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□□counter□□□□□□□□□counter□□□□□ log2() □□□□
3. □□ counter □□□□□□□□□□min, max, mean, and standard deviation for counters
4. □□ counter □□□□□ counter □□□□□□□□□□□□□□□□□□□□□ counter □□□□ .png □□□
5. □□ '-r n| --ramplines n' □□□□□□□□□n□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
6. □□ '--startline s --endline e' □□□□□□□□□□□□□□□□□'counterana.py' □□□□ s □□□ e □□□□□□□□□□□□□□□□□□□□□□ '--startline 60 --endline 120' □
□□□□□2□□□□□□□
7. □□ '-g -c' □□□□□□□□counter□□□□□□□□□□□□□□□□counter□□□□□□□□'-c'□□□□□□counter□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ '-g'
□□□□□□□□□□counter□□□□□□
8. □□ '-g -d' □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□counter□□□□□□□□□□□□□□□□counter□□□□□□□□□
□□□
$ uio_scripts/server/counterana.py -h
usage:uio_scripts/server/counterana.py [logname] [-e counter_pattern] [-i] [-m|--histogram] [-r|--ramplines] [-k] [--startline n] [--endline
n]
                                        [-g|--graph] [-c|--combine] [-d|--diff]
Analyze UniIO counter log files.
options:
  -e pattern:       filter of counter names
  -i:               ignore case
  -g, --graph:      plot a scatter graph for counters
  -c, --combine:    use with '-g', plot all data onto a single chart
  -d, --diff:       use with '-g', plot changes between values of a counter
  -m, --histogram:  print histogram (log2 buckets)
  -r, --ramplines:  ramping lines. to skip first and last few lines of data
  --startline:      specify a start line, to only analyze lines after that line
  --endline:        specify an end line, to only analyze lines before that line
  -k:               keep temp files
if no 'logname' given in command line, counterana.py reads counter data from stdin
examples:
  counterana.py counter.log                # report all counters in 'counter.log' (massive lines will slow down the analysis)
  cat counter.log | counterana.py          # same as above
  counterana.py counter.log -e ss.obs      # only report counters that contain 'ss.obs'
  grep ss.obs counter.log | counterana.py  # same as above
  counterana.py counter.log -e ss.obs -g   # report counters that contains 'ss.obs' and plot a graph for each of the counters
  counterana.py counter.log -e ss.obs -gc  # report counters that contains 'ss.obs' and plot all counter data onto a single graph
  counterana.py counter.log -e ss.obs -m   # report counters that contains 'ss.obs' and print the histogram for each of the counters
  counterana.py counter.log --startline=60 --endline=120   # report all conter data betwen 60min ~ 120min (if sample interval is 60s)
output format:
  counter_name[sample_count][unit][trends]: min, max, mean, mean_squared_deviation, standard_deviation, pct_stddev:mean, slop
  * each line summarizes a unique counter *
how to intepret:
  sample_count:   how many samples(lines) have been aggregated for a counter
  unit:           the unit of a counter (counts, uSec, KiB)
  trends:         trends of the sample value from the first sample to the last in [UP|DOWN|FLAT|NOCHANGE|SPIKES]
  slop:           result of linear regression(the 'a' in y=ax+b). how fast the sample value increase|decreases
  self explained: min, max, mean, mean_squared_deviation, standard_deviation, pct_stddev:mean
```

□ □□ 'counterana.py' □□□□□□

1. □□□□□□□□□□□□□□□□□□□□□□□□ counter□□□□'UP','DOWN'□□□counter□□□□□□□□

```
# □□□□□□ obs □□□□ counters□
$ server/counterana.py -e ss.obs counter.log | grep -E 'UP|DOWN'
building aggregated array ... done.
===================================================================================
...
ss.obs.WriteSlab.outstanding[523][counts][DOWN]: min=1384126.0 max=4194304.0 mean=3042116.2 > > stddev=984976.2
stddev:mean=32.4% slop=-5432.041
ss.obs.cacheWriteEvictions[523][counts][UP]: min=742242.0 max=1237425534.0 mean=617743522.8 stddev=363576028.8
stddev:mean=58.9% slop=2407954.897
ss.obs.cacheMigrateFromWriteToRead[523][counts][UP]: min=310342.0 max=1260108523.0 mean=627497530.2 stddev=371270071.4
stddev:mean=59.2% slop=2458905.563
...
```

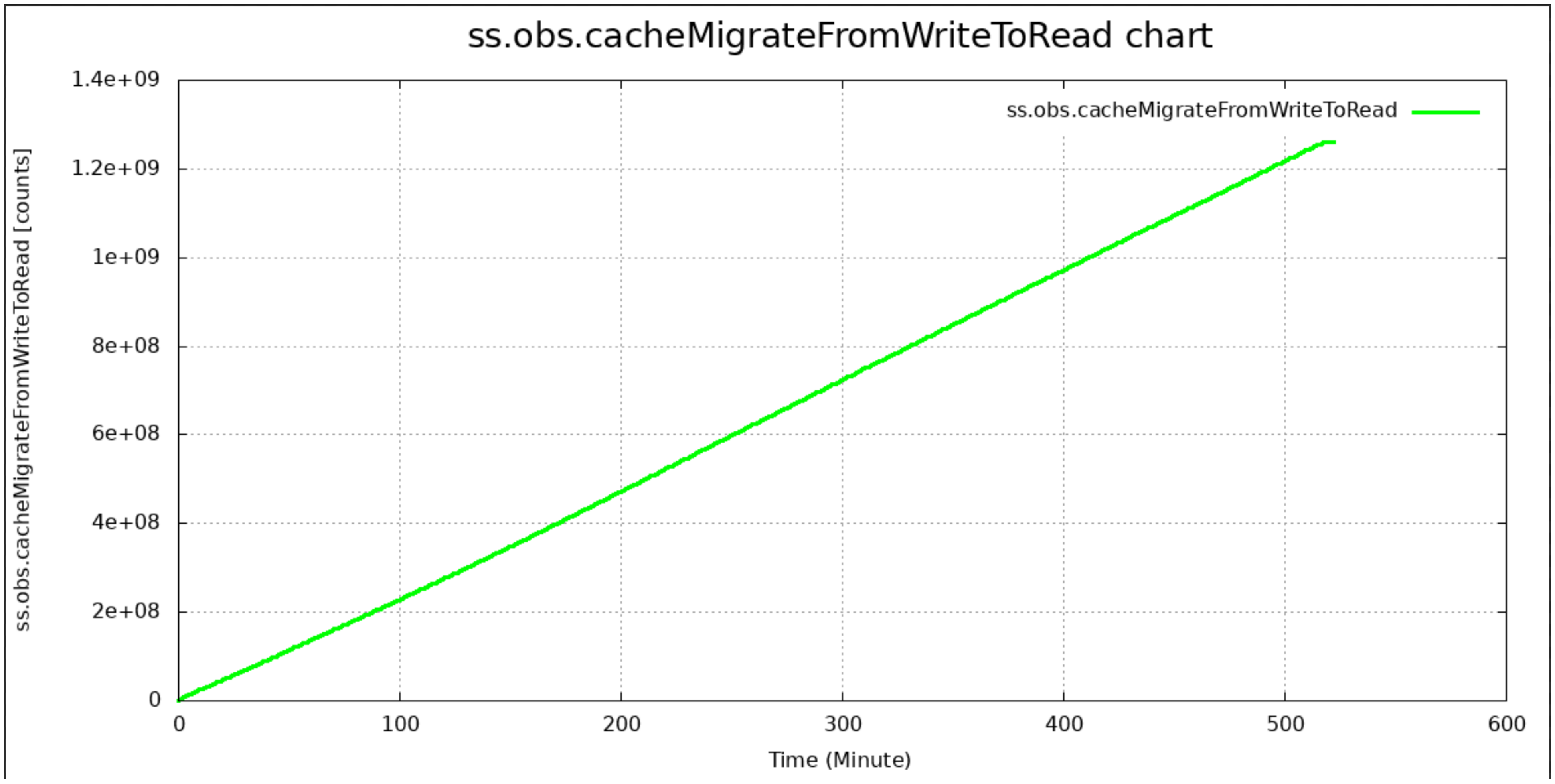
2. □□□□□□□□ ss.obs.cacheMigrateFromWriteToRead □□□□□□(stddev:mean=58.9%)□□□□□□□ UP □□□□□□□□(-m)□□□□ counter □□□□□□

```
$ server/counterana.py counter.log -e ss.obs.cacheMigrateFromWriteToRead -m
building aggregated array ... done.
===================================================================================
ss.obs.cacheMigrateFromWriteToRead[523][counts][UP]: min=310342.0 max=1260108523.0 mean=627497530.2 stddev=371270071.4
stddev:mean=59.2% slop=2458905.563
===================================================================================
Histogram for ss.obs.cacheMigrateFromWriteToRead (counts) ... 523 samples.
                 (0...1]    0
                 (1...2]    0
                 (2...4]    0
                 (4...8]    0
                (8...16]    0
               (16...32]    0
               (32...64]    0
              (64...128]    0
             (128...256]    0
             (256...512]    0
            (512...1024]    0
           (1024...2048]    0
           (2048...4096]    0
           (4096...8192]    0
          (8192...16384]    0
         (16384...32768]    0
         (32768...65536]    0
        (65536...131072]    0
       (131072...262144]    0
       (262144...524288]    1
      (524288...1048576]    0
     (1048576...2097152]    0
      (2097152...4194304]    1
    (4194304...8388608]    2
     (8388608...16777216]    3
    (16777216...33554432]    8
    (33554432...67108864]    15
   (67108864...134217728]    30
  (134217728...268435456]    58
  (268435456...536870912]    108
 (536870912...1073741824]    216
(1073741824...2147483648]    81
```

3. □□□□□counter□□□□□□□□□□□□□□□□□□□□counter□□□□□□□(-g)□□□□□□□□□

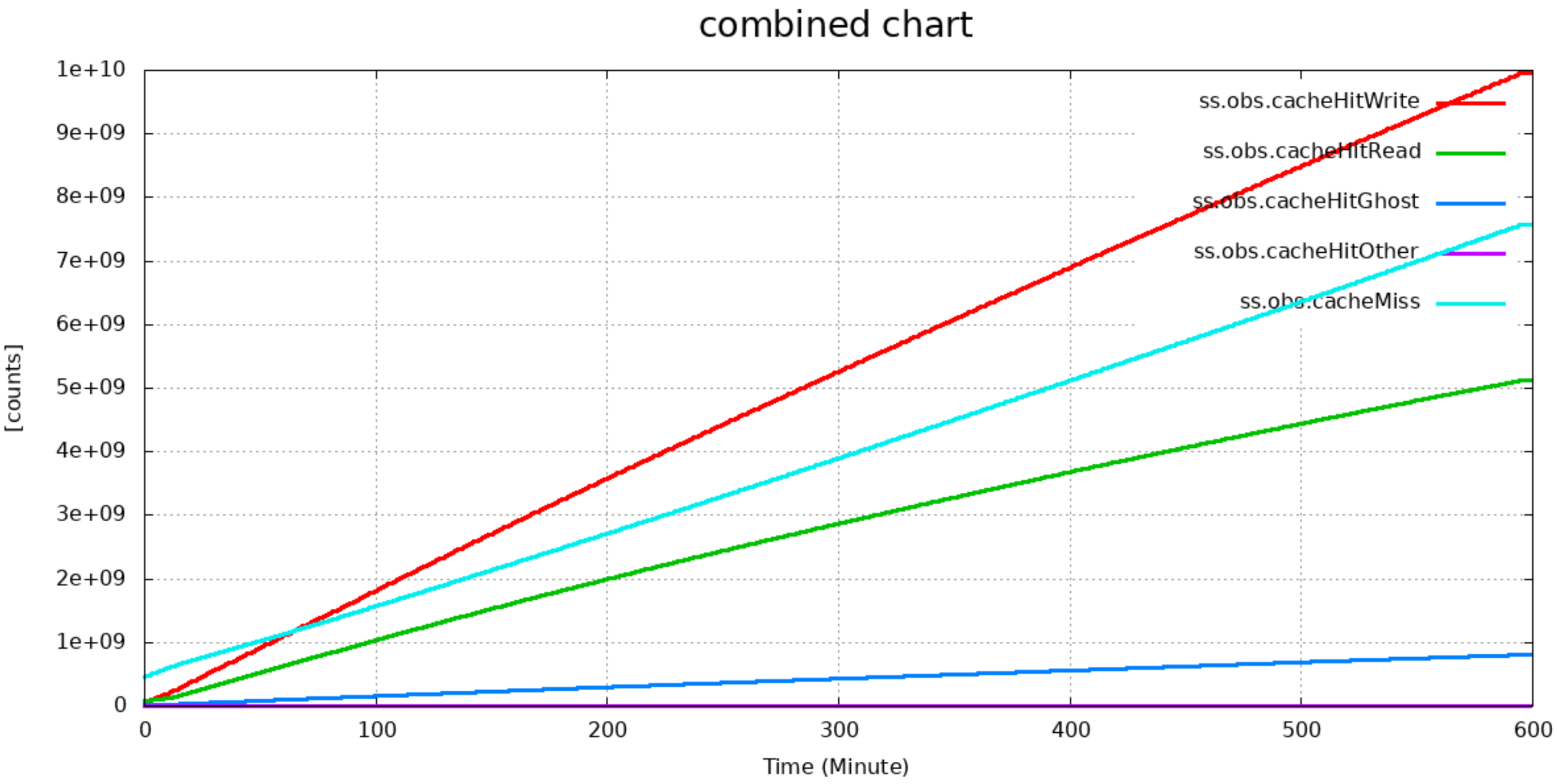
```
$ server/counterana.py counter.log -e ss.obs.cacheMigrateFromWriteToRead -g
building aggregated array ... done.
===============================================================================
ss.obs.cacheMigrateFromWriteToRead[523][counts][UP]: min=310342.0 max=1260108523.0 mean=627497530.2 stddev=371270071.4
stddev:mean=59.2% slop=2458905.563
===============================================================================
ss.obs.cacheMigrateFromWriteToRead.png
```

4. □□□□□□□□□ ss.obs.cacheMigrateFromWriteToRead.png □□□□□□□□□□(□□□ plotfio.sh □□□□□□iops□latency□□)□□□□□□□□□□□□□□□□□



5. □□□□□□□□□□□□ counter □□□□□□□□□□□□

```
$ uio_scripts/server/counterana.py counter.log -e 'ss.obs.cache(?=Miss|Hit)' -gc
 building aggregated array ... done.
===============================================================================
ss.obs.cacheHitWrite[600][counts][UP]: min=61190410.0 max=9962262664.0 mean=5170040745.2 stddev=2889397492.9
stddev:mean=55.9% slop=16676433.790
ss.obs.cacheHitRead[600][counts][UP]: min=79458824.0 max=5126165414.0 mean=2767213820.0 stddev=1479525548.5
stddev:mean=53.5% slop=8527487.472
ss.obs.cacheHitGhost[600][counts][UP]: min=13269294.0 max=806473330.0 mean=423335945.3 stddev=230776016.9
stddev:mean=54.5% slop=1331940.169
ss.obs.cacheHitOther[600][counts][NOCHANGE]: min=0.0 max=0.0 mean=0.0 stddev=0.0 stddev:mean=0.0% slop=0.000
ss.obs.cacheMiss[600][counts][UP]: min=454818307.0 max=7578260188.0 mean=3942654687.8 stddev=2070806872.7
stddev:mean=52.5% slop=11952484.988
===============================================================================
/root/fred/ss.obs.cacheHitWrite_more.plotdata.png
```



□□ UniIO counters □□□□□□□□□□□□□□□

```
$ cat counters.sh
#!/usr/bin/env bash
#usage: ./counters.sh [interval] [runtime]
runtime=36000  # how long, default 10 hours
interval=60    # how often, default every 60 seconds
[[ ! -z "$1" ]] && runtime=$1
[[ ! -z "$2" ]] && interval=$2
total=0
while true
do
  date
  arrayctl counters
  sleep $interval
  total=$((total+$interval))
  [[ $total -ge $runtime ]] && break
done
$ nohup ./counters.sh 36000 > counter.log 2>&1 &
```

server/init_backend.sh

```
□□□
  1. □□ UniIO □□□
  2. □ DP □□□□ 'config.ini' □□□□
  3. □□□□□□□□□□□□□□□□□ coredump □□□
□□□□□□□□□□□□□□□□□ root □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□
$ server/init_backend.sh -h
usage: init_backend.sh [ clear|init ] [ -G dumpdev_size ]
```

server/init_cluster.sh

```
□□□uniio □□□□□□□□□□□□□□□□□□□RPM□□□□□□□□□□□□□□LUN
!!! □□□□□□□'-d|--initbackend'□□□□□□□□□□□□'init_backend.sh'□□□□□□□□□□□□□□ root □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□
$ server/init_cluster.sh -h
usage: init_cluster.sh [-f] [-s|--stoponly]
                      [-b|--bootonly]
                      [-r|--replace rpm_dir]
                      [-d|--initbackend] [-G dump_size]
                      [-i|--initarray]
                      [-c|--createluns --management_ip ip --iscsi_ip ip --topology ip,ip...]
 -f: force (killing cio_array)
```

```
  -s: stop only
  -b: start objmgr and objmgr-fab
  -d: initialize backend
  -G: prereserve size for coredump device
  -i: initialize array
  -c: create new luns and mappings
  --management_ip: specify the management IP address for the federation
  --iscsi_ip:      specify the management IP address for the federation
  --topology:      specify the node IP addresses for the federation
```

auto perftest scripts

auto/perfauto.py

```
[illegible] UniIO Federation [illegible], fio [illegible], [illegible]
1. '-c' [illegible]
2. '-u' [illegible] UniIO Federation [illegible]perfauto.py [illegible]uniio, uniio-ui, sysmgmt, nasmgmt[illegible]RPM[illegible]
[illegible]RPM[illegible]
3. ‘-u --binonly’ [illegible]RPM[illegible]uniio[illegible]cio_array, cio_array.sym)[illegible]UniIO Federation [illegible]
UniIO Federation [illegible] '/opt/uniio/sbin/cio_array'[illegible]‘-u --binonly=./replacefile/cio_array’
4. ‘--binonly=xxx' [illegible]cio_array[illegible]cio_array.sym[illegible]'--binonly=xxx'[illegible]
[illegible]git[illegible]commit[illegible] '-c' [illegible]git[illegible]'--binonly=xxx'[illegible]git[illegible]
2[illegible]a. [illegible]uniio[illegible]RPM[illegible]2. [illegible]'--binonly'[illegible]'--binonly=xxx' [illegible]'--binonly=conf'[illegible]
[illegible]git[illegible]
5. '-i' [illegible]uniio[illegible]'perfauto.py'[illegible]'init_cluster.sh'[illegible]
6. '-p' [illegible]fio[illegible]lun[illegible]fio[illegible]counter[illegible]cpu[illegible]
[illegible]'runfio.sh'[illegible]'counters.sh'[illegible]
7. '-p --cpudata' [illegible]fio[illegible]uniio[illegible]’collect_cpu.sh‘[illegible]cpu[illegible]
8. '-p --fill=sec' [illegible]fio[illegible]LUN[illegible]sec[illegible]
9. '--fullmap' [illegible]'--createluns'[illegible]ISCSI[illegible]LUN[illegible]LUN[illegible]
[illegible]
$ uio_scripts/auto/perfauto.py -h
usage: perfauto.py [ -c|--config configfile.json ]
                   [ -f|--force ] [ -s|--shutdown ]
                   [ -b|--boot ]
                   [ -u|--update ] [ --binonly (binpath|conf|tag|branch|commit) ]
                   [ -i|--init ]
                   [ -p|--perftest ] [ --cpudata ] [ --fill sec ]
                   [ --createluns num ] [ --fullmap ] [ --deleteluns ]
Coordinate UniIO nodes, build server and fio clients for performance test.
options:
  -c, --config:      config file path (.json)
  -f, --force:       force stop uniio node (kill cio_array)
  -s, --shutdown:    gracefully stop uniio nodes
  -b, --boot:        start uniio nodes
  -u, --update:      update uniio build
      --binonly:     use along with '-u', only update cio_array binary.
  -i, --init:        reinit uniio federation
  -p, --perftest:    run perftest
      --cpudata:     use along with '-p', collect cpu data as svg files while performance test is running
      --fill:        use along with '-p', fill the luns with pure write workload for a given time in seconds
  --createluns:      create a given number of luns
     --fullmap:      use along with '--createluns', all clients see all luns ( clients see different luns if not specified )
  --deleteluns:      delete all existing luns
```

[illegible]

```
{
    "runtime_dir" : "/tmp/uio",
    "client_nodes" : [
        ["192.168.100.169", "root", "p@ssword"],
        ["192.168.100.155", "root", "password"],
        ["192.168.100.156", "root", "password"]
    ],
    "federation_nodes" : [
        ["192.168.100.206", "root", "password"],
        ["192.168.103.248", "root", "password"],
        ["192.168.101.169", "root", "password"]
    ],
    "build_server" : ["192.168.100.120", "root", ".id_rsa", "/root/fred/.id_rsa"],
    "build_server_git_proxy" : "socks5://192.168.100.120:8899",
    "uniio_checkout" : "default",
    "num_luns" : 18,
    "lunsize_G" : 1000,
    "topology" : "192.168.101.169,192.168.103.248,192.168.100.206",
    "management_ip" : "192.168.103.253",
    "iscsi_ip" : "192.168.60.253",
    "fio_runtime" : 10800,
    "fio_ramp_time" : 0,
    "fio_dedupe_percentage" : 80,
    "fio_buffer_compress_percentage" : 60,
    "fio_random_distribution" : "random",
    "### fio_random_distribution can be any fio supported distributions: [random, zipf:0.96, pareto:ratio, ..]" : "",
    "fio_rw" : "randrw",
    "### fio_rw can be 'sepjob[_fio-supported-rw]' or any fio supported rw types" : "",
    "### fio_rw 'sepjob_xxx' means use different jobs for read and write in mixed workload" : "",
    "### fio_rw example: 'sepjob', 'sepjob_randrw', 'sepjob_rw' " : "",
    "fio_rwmixread" : 80,
    "fio_rwmixwrite" : 20,
    "runfio_jobs" : "1",
    "runfio_qdepth" : "4",
    "runfio_xxx is arguments for 'runfio.sh', e.g. --jobs --qdepth" : ""
}
```