UniIO Performance Scripts

- install
- client scripts
  - client/plotfio.sh
- server scripts
  - server/collect_cpu.sh
  - server/counterana.py
  - server/init_backend.sh
  - server/init_cluster.sh

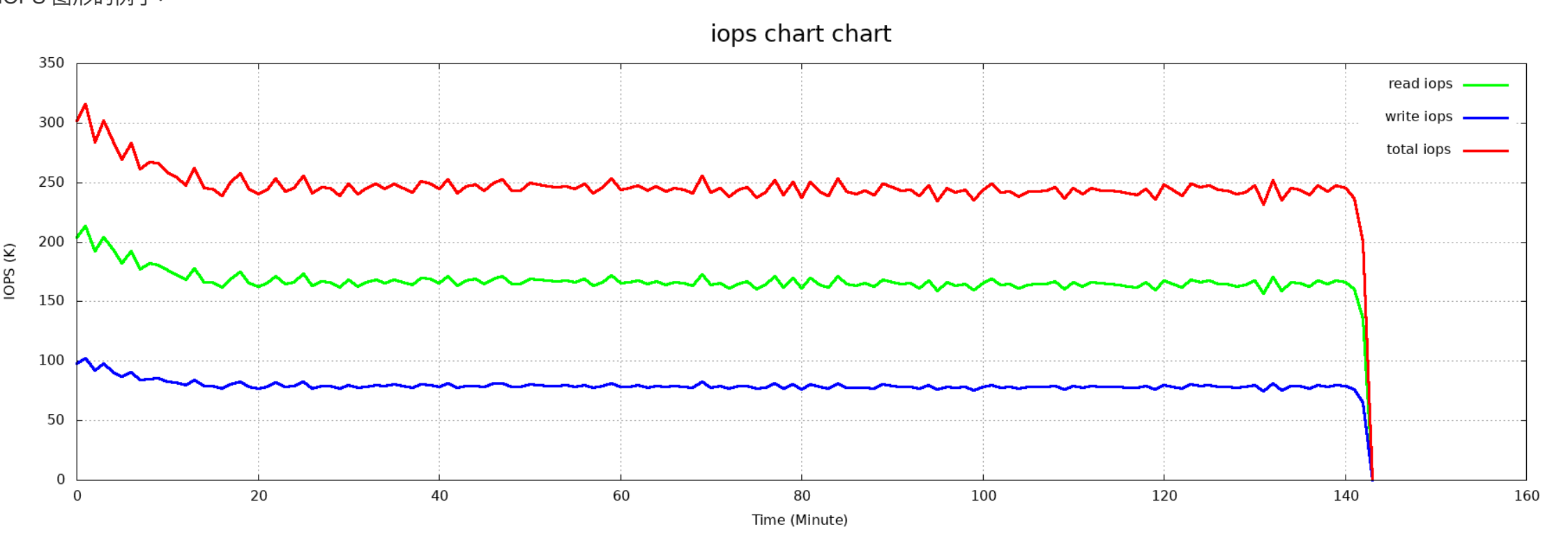
install

```
$ 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

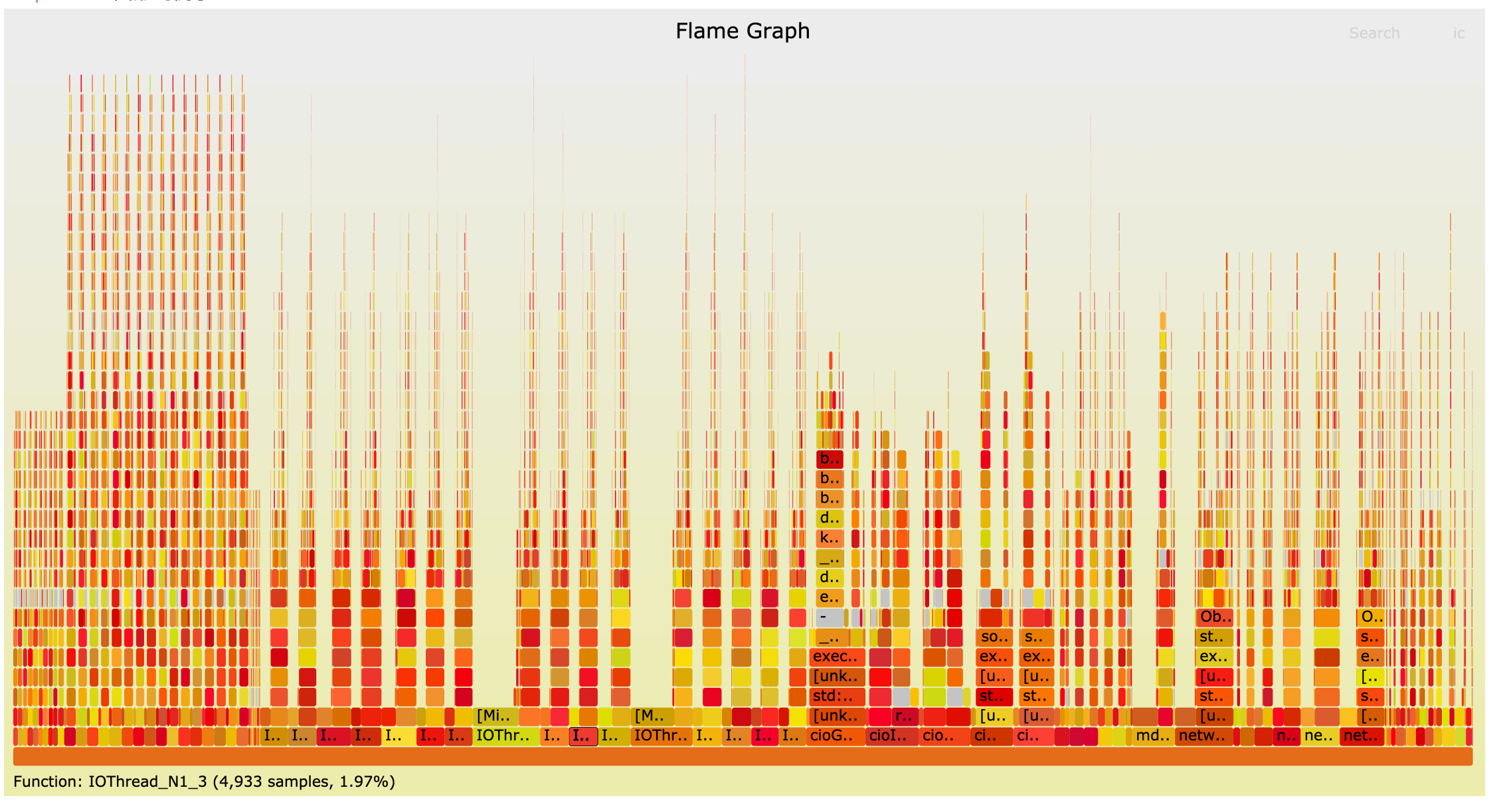
```
功能：对多个 fio 日志文件中的数据进行分类汇总（fio给每个job产生一个日志文件），按时间生成数据走势图。 支持的图形类型为：IOPS，SLAT, CLAT, LAT。

用法：
# 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 图形的例子：
>
> 
>

server scripts

server/collect_cpu.sh

```
功能：利用 linux bcc/eBPF 搜集 oncpu，offcpu，wakeup，offwakup 用户态和内核态软件栈，并生成交互式 SVG 图案。
运行条件：
  kernel version >4.8
  eBPF enabled with kernel
  bcc installed
  FlameGraph installed and located in ../FlameGraph

用法：
# 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 图形的例子：
>
> 
>

server/counterana.py

```
功能：分析 'arrayctl counters' 命令所记录的 UniIO 内部计数器日志。主要可用的功能有：
1. 分析 counter 值的走势:
  以线性回归的结果为基础，大致分析 counter 随时间变化的趋势。可能的结果有：
  * 'NOCHANGE'  从始至终都没有改变过的counter。这些counter往往是没用的。可能是因为被disable，或者未更新，或者有bug。
  * 'UP'        不停增加的counter。这些counter意味某种操作的时间或数量不断增加，或资源加速消耗，可能导致性能问题。
  * 'DOWN'      不停减少的counter。这些counter跟UP趋势的counter类似，可能意味着系统正走向某种瓶颈，导致性能问题。
  * 'SPIKES'    总体没有明显增减，但会有突然的剧烈波动。这种趋势可能意味着客户端压力的突变，或者cio_array内部资源分配出现抖动。
  * 'FLAT'      总体没有明显增减，也没有剧烈波动。这种趋势属较为平衡状态，关注优先级可以放低。
2. 打印直方图：可以分析一个或多个counter的值分布，直方图将counter数量分布在 log2() 个桶中。
3. 打印 counter 的基本分类汇总信息：min, max, mean, and standard deviation for counters
4. 绘制 counter 图形，展示 counter 的值随时间的变化。绘制完成后会为指定的每个 counter 生成一个 .png 文件。
5. 使用 '-r n| --ramplines n' 参数，可以跳过前后n次采样的数据。采样的开始和结束阶段系统往往还处于不太稳定的状态，跳过这些采样数据有助于提高分析的准确性。
6. 使用 '--startline s, --endline e' 参数，可以只分析某个时间段的数据。'counterana.py' 会抽取第 s 行和第 e 行之间的数据。如果数据是按每分钟采样的，那么 '--startline 60, --endline 120' 代表只分析第2个小时的数据。

用法：
$ uio_scripts/server/counterana.py -h
usage:uio_scripts/server/counterana.py [logname] [-e counter_pattern] [-i] [-g|--graph] [-m|--histogram] [-r|--ramplines] [-k] [--startline n] [--endline n]

Analyze UniIO counter log files.

options:
  -e pattern:      filter of counter names
  -i:              ignore case
  -g, --graph:     plot a scatter graph for counters
  -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 -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
```

> **使用 'counterana.py' 的建议流程：**
>
> 1. 第一步先分析整个日志文件，或某个子系统中的所有 counter，筛选出'UP','DOWN'趋势的counter，以便重点关注。

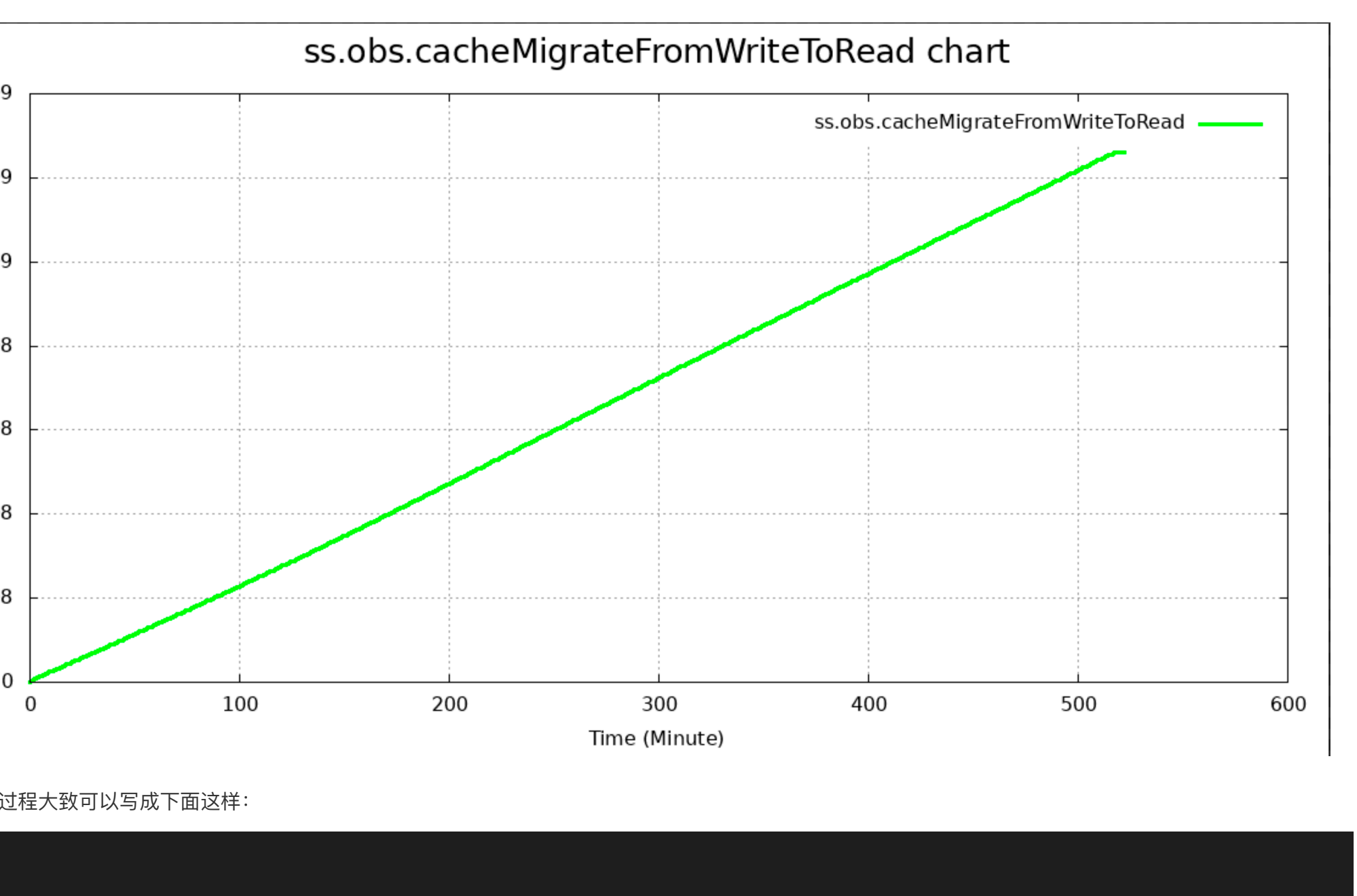
```
# 下面例子分析 obs 子系统的 counters：
$ server/counterana.py -e ss.obs counter.log | grep -E 'UP|DOWN'
building aggregated array ... done.
================================================================================
...
ss.obs.WriteSlab.outstanding[523][counts][DOWN]: min=1384126.0 max=4194304.0 mean=3042116.2 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. 观察输出，发现ss.obs.cacheMigrateFromWriteToRead 变化幅度较大(stddev:mean=58.9%)，且趋势是走高UP 。单独打印直方图(-m)查看可疑 counter 的分布情况。

```
$ 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. 初步发现该counter的值分布在高位居多，越高越多。最后将该counter的图形走势画出(-g)，进一步查看比对：

```
$ 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. 打开生成的图像文件 ss.obs.cacheMigrateFromWriteToRead.png ，将其趋势与其他数据(例如用plotfio.sh 生成的客户端iops或latency图形)交叉对比，分析其持续升高的原因。
>
> 
>

收集 UniIO counters 的过程大致可以写成下面这样：

```
$ cat counters.sh
while true
do
  date
  arrayctl counters
  sleep 60
done

$ nohup ./counters.sh > counter.log 2>&1 &
```

server/init_backend.sh

```
功能：
  1. 抹除 UniIO 数据盘
  2. 为 DP 后端生成 'config.ini' 配置文件
  3. 从每个后端磁盘中预留一部分空间作为 coredump 设备。
！ 注意：此脚本将重新初始化所有除了 root 设备之外的其他磁盘设备，具有相当危险性，只能用于实验环境。

用法：
$ server/init_backend.sh -h
usage: init_backend.sh [ clear|init ] [ -G dumpdev_size ]
```

server/init_cluster.sh

```
功能：uniio 单节点清空环境，后端初始化，服务启停，RPM包更换，集群拓扑初始化并创建LUN
!!! 注意，当指定了'-d|--initbackend'参数，需要当前目录下存在'init_backend.sh'，且脚本将重新初始化所有除了 root 设备之外的其他磁盘设备，具有相当危险性，只能用于实验环境。

用法：
$ 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
```