```
install

    client scripts

    client/plotfio.sh

    server scripts

    server/collect_cpu.sh

    server/counterana.py

      • server/init_backend.sh

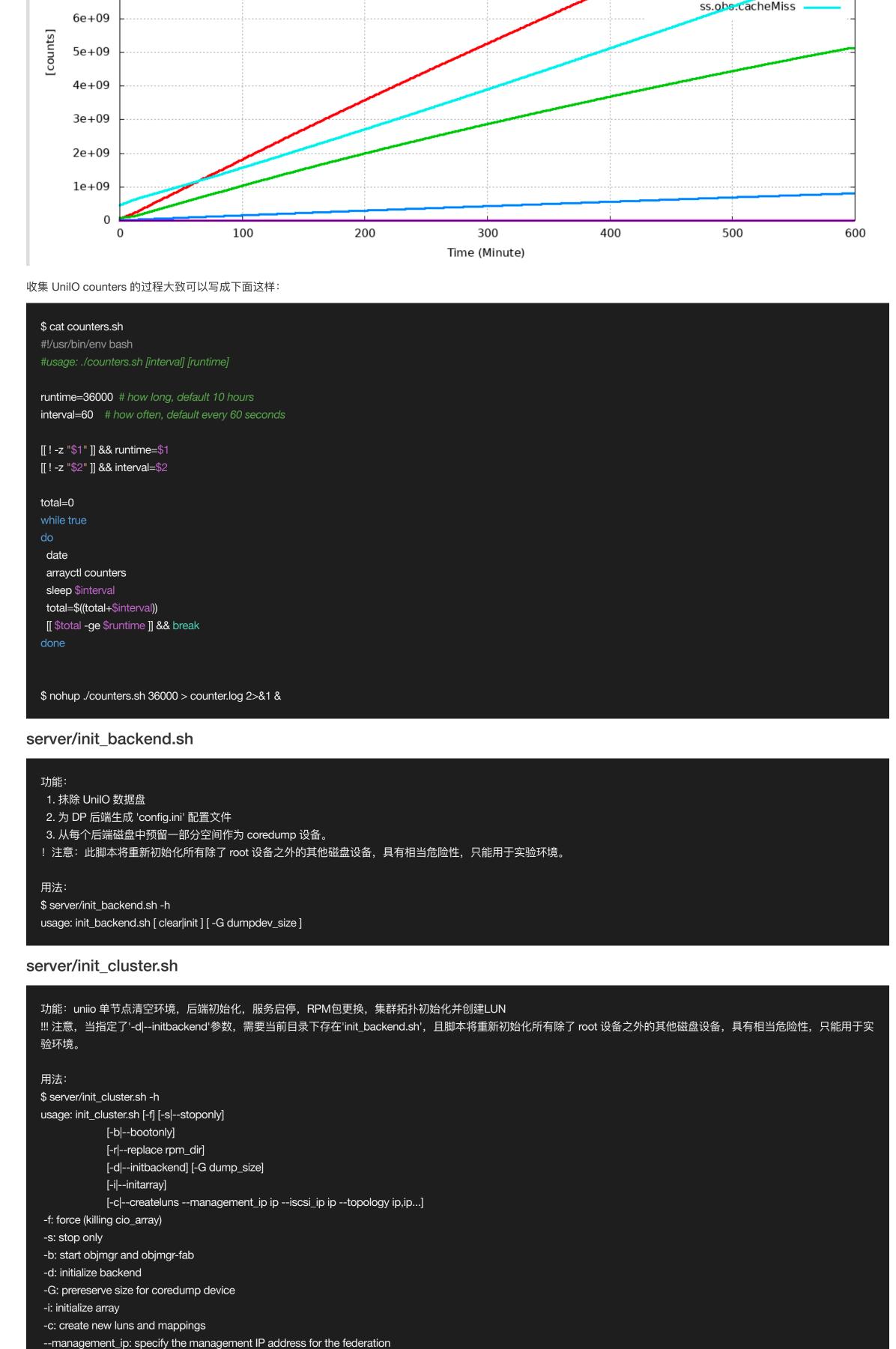
    server/init_cluster.sh

  • auto perftest scripts

    auto/perfauto.py

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:
                                                                     iops chart chart
      350
                                                                                                                                     read iops
                                                                                                                                     write iops
      300
                                                                                                                                     total iops
      250
      200
   IOPS (K)
      150
      100
       50
                                                                                             100
                                                                                                              120
                                                                                                                               140
                                                                                                                                                160
                                                                         Time (Minute)
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:
                        # gather all types of cpu data for 60 seconds and generate flame graphs. prefix 'this'
   collect_cpu.sh
   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:
                                                                     Flame Graph
                                              I.. IOThr.. I.. I.. I.. IOThr.. I.. I.. I.. cioG
    Function: IOThread_N1_3 (4,933 samples, 1.97%)
server/counterana.py
  功能:分析 'arrayctl counters' 命令所记录的 UnilO 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。
            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。
             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。
  7. 使用 '-g -c' 参数,可以将多个counter的数据绘制到同一个图形,方便比较counter走势。注意使用了'-c'参数时,多个counter应该具有同样的单位,否则图形会失去意义
   (将时间和次数相比是没有意义的)。如果只使用 '-g' 参数,则默认为每个counter生成一张图形。
  8. 使用 '-g -d' 参数,可以观察两次采样之间的差值。差值观察对于一些累计的总是增长的counter较为有用。可以观察到每次采样区间counter新增或减少的数量。
   用法:
  $ 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 UnilO counter log files.
  options:
   -e pattern:
               filter of counter names
            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:
                                 # report all counters in 'counter.log' (massive lines will slow down the analysis)
   counterana.py counter.log
   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
            the unit of a counter (counts, uSec, KiB)
             trends of the sample value from the first sample to the last in [UP|DOWN|FLAT|NOCHANGE|SPIKES]
   trends:
             result of linear regression(the 'a' in y=ax+b). how fast the sample value increase|decreases
   slop:
   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。
                                                 ss.obs.cacheMigrateFromWriteToRead chart
             1.4e + 09
                                                                                                  ss.obs.cacheMigrateFromWriteToRead
        ss.obs.cacheMigrateFromWriteToRead [counts
             1.2e + 09
```

**UnilO Performance Scripts** 



1e + 09

8e+08

6e + 08

4e + 08

2e+08

0

building aggregated array ... done.

1e+10

9e + 09

8e + 09

7e+09

/root/fred/ss.obs.cacheHitWrite\_more.plotdata.png

--iscsi\_ip: specify the management IP address for the federation

收集和cpu数据收集。性能测试任务会调用到'runfio.sh'和'counters.sh'。

功能:协调 UnilO Federation 服务器, fio 客户端,以及编译服务器,完成端到端的性能测试。

1. '-c' 选项指定一个任务配置文件。文件里配置了所涉及的客户端,服务器,以及编译服务器的访问方式。

5. '-i' 不要升级或者替换二进制,直接重新初始化uniio集群。'perfauto.py'会调用'init\_cluster.sh'来初始化集群。

--topology: specify the node IP addresses for the federation

auto perftest scripts

auto/perfauto.py

在目标服务器上替换这些RPM包。

100

5. 或者,也可以将多个相关的 counter 放在同一个图中进行比较:

\$ uio\_scripts/server/counterana.py counter.log -e 'ss.obs.cache(?=Miss|Hit)' -gc

200

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

300

Time (Minute)

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.cacheMiss[600][counts][UP]: min=454818307.0 max=7578260188.0 mean=3942654687.8 stddev=2070806872.7 stddev:mean=52.5% slop=11952484.988

combined chart

400

500

ss.obs.cacheHitWrite

ss.obs.cacheHitRead

ss.obs.cacheHitGhost

ss.obs.cacheHitOthe<u>r</u>

600

7. '-p --cpudata' 在fio运行期间,脚本每隔一小时在uniio服务器上调用'collect\_cpu.sh'收集cpu数据。 8. '-p --fill=sec' 在执行fio性能测试之前,先用纯写给LUN填数据,时间由sec指定。 9. '--fullmap' 跟'--createluns'一起使用,指定在ISCSI映射时是否让所有客户端看见所有的LUN。默认情况下,每个客户端都看见不同的LUN,这样读写不会互相覆盖。 用法: \$ 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 UnilO nodes, build server and fio clients for performance test. options: -c, --config: config file path (.json) force stop uniio node (kill cio\_array) -f, --force: -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 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 配置文件例子: "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"],

2. '-u' 选项可以自动编译并自动升级 UnilO Federation 服务器,默认情况下,perfauto.py 会编译所有的相关的库(uniio, uniio-ui, sysmgmt, nasmgmt),并生成RPM包,然后

3. '-u --binonly' 表示不要替换RPM包,而只替换uniio的二进制文件(cio\_array, cio\_array.sym)。可以指定一个本地文件路径,脚本会将这个文件上传到UnilO Federation 服务

4. '--binonly=xxx' 选项除了可以用一个本地文件替换服务器上的文件,还可以重新编译所需的二进制文件cio\_array和cio\_array.sym。如果'--binonly=xxx'所指定的不是一个路 | 径,那么脚本就会认为这是指定的一个git分支名称,或者commit哈希。这样脚本就不会上传本地文件,而是从编译服务器上去重新编译一个二进制文件。默认情况下,脚本 会从 '-c' 所指定的配置文件中读取git分支名。一旦'--binonly=xxx'指定了git分支名,那么就有2个含义:a. 只替换uniio的二进制文件,不替换所有的RPM。2. 用'--binonly'所指

6. '-p' 选项表示开始一次端到端性能测试。脚本会根据配置文件生成fio的任务文件,创建并映射lun到客户端,并且协同所有的客户端启动fio,同时根据情况启动counter日志

器上,用这个文件替代 UnilO Federation 服务器上的 '/opt/uniio/sbin/cio\_array'。这样就可以避免重新编译,节省时间。例如: '-u --binonly=./replacefile/cio\_array'

定的分支名覆盖配置文件中指定的分支名。另外,'--binonly=xxx' 还可以是'--binonly=conf',表示只替换二进制文件,但编译仍然使用配置文件里指定的git分支。

"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" : ""