Lab. Performance evaluation of virtualization technologies

High Performance Infrastructures (HPC Master)

Roberto R. Expósito

Department of Computer Engineering

Universidade da Coruña

Lab. Performance evaluation of virtualization technologies

- This lab consists of evaluating the performance of two different virtualization technologies: **VMs vs Containers**
- This document show you how to conduct such evaluation using two popular virtualization technologies:
  - **VirtualBox**, as an example of hypervisor-based virtualization
    - **Vagrant** can be used to ease the deployment of the VM
  - **Docker**, as an example of OS-level container-based virtualization
- Both technologies will be compared using the same system resources
  - 2 CPU cores and 2 GB of memory
- You may use any virtualization solution that you consider appropriate
  - Hypervisor-based virtualization: KVM, Hyper-V, Parallels Desktop, VMware Workstation/Fusion, UTM…
  - OS-level virtualization: LXC, Singularity, Podman, udocker...
  - **Anyway, you must execute the same set of benchmarks**

Lab. Performance evaluation of virtualization technologies

- The evaluation will be performed at multiple levels:
  - **CPU**: for evaluating single-core/multi-core performance
  - **Memory**: for evaluating sustainable memory bandwidth
  - **Disk I/O**: for evaluating sequential/random I/O performance
- Using different benchmarking tools:
  - Geekbench
    - https://www.geekbench.com
  - NAS Parallel Benchmarks (NPB)
    - https://www.nas.nasa.gov/publications/npb.html
  - STREAM
    - https://github.com/jeffhammond/STREAM
  - Flexible I/O tester (FIO)
    - https://github.com/axboe/fio
- Under a GNU/Linux RHEL-based OS: **Rocky Linux 8.6 (Green Obsidian)**
  - https://rockylinux.org

IMPORTANT INSTRUCTIONS

Lab. Performance evaluation of virtualization technologies

- When executing the benchmarks, try to provide the “best” conditions
  - A single virtualization platform and benchmark running at the same time
  - Minimize any other workload on your computer while running them
- **To complete this lab, you must upload a single file in Aula CESGA:**
  1. A PDF document named **IAP-Virt.pdf**
     - First of all, describe **briefly the features of your computer**
       - CPU: model, number of cores, speed
       - Memory: amount, speed and type (DDR4, DDR5...)
       - Disk: model, interface (SATA, PCIe...), type (HDD, SDD)
       - Operating system version
     - **For each benchmark**
       - Include the corresponding **screenshots** that **demonstrates the execution of the corresponding benchmark** for each virtualization platform

         - **DO NOT crop your screenshots**, all relevant information must be clearly appreciated without any confusion
       - **Summarize the results** obtained for both virtualization platforms (tables are strongly recommended) and **compare them in a few words**

VIRTUALBOX

Lab. Performance evaluation of virtualization technologies

- If you do not have VirtualBox in your system, install it:
  - Check VirtualBox website at: https://www.virtualbox.org/wiki/Downloads
  - Install version 6.1.38 or higher
- If you want to use Vagrant (**optional, recommended**), install it:
  - Check Vagrant website at: https://www.vagrantup.com/downloads
  - Install version 2.3.1 or higher
- **If you are using Vagrant**, download the Vagrant project from Aula CESGA
  - It is **mandatory** to change the **hostname** of your VM in the Vagrantfile
    - Change **HOSTNAME_VM** variable by replacing **xxx** with the initials of your name and surname
      - Example for student "**R**oberto **R**ey **E**xpósito":
        - HOSTNAME_VM = "**rre**-iap2223"

- **Otherwise**, download the VM image and **import** it to VirtualBox ← **Follow these instructions only if you are NOT using Vagrant**
  - Be sure to configure the VM with 2 CPU cores and 2 GB of memory
  - You must also change the VM hostname to use your prefix

Lab. Performance evaluation of virtualization technologies

- **If you are using Vagrant:**
  - Deploy and provision the VM: **vagrant up**
    - During deployment, the VM is provisioned with all the software you need to use for evaluating the performance
  - After deployment, connect to the VM: **vagrant ssh**

```
    default:   scl-utils-1:2.0.2-14.el8.x86_64
    default:   source-highlight-3.1.8-17.el8.x86_64
    default:   tbb-2018.2-9.el8.x86_64
    default:   tcl-1:8.6.8-2.el8.x86_64
    default:   unzip-6.0-45.el8_4.x86_64
    default:   xz-devel-5.2.4-3.el8.1.x86_64
    default:   zip-3.0-23.el8.x86_64
    default:
    default: Complete!
    default: 46 files removed
[rober@oceania Virtualization]$ vagrant ssh
[vagrant@xxx-iap2223 ~]$ uname -a
Linux xxx-iap2223 4.18.0-305.25.1.el8_4.x86_64 #1 SMP Tue Nov 2 10:32:37 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
[vagrant@xxx-iap2223 ~]$ ls
Geekbench-5.4.3-Linux  Geekbench-5.4.3-Linux.tar.gz  NPB3.4.2  NPB3.4.2.tar.gz  STREAM  STREAM.tgz  fio-3.28  fio-3.28.tgz
[vagrant@xxx-iap2223 ~]$
```

Be sure to set the VM hostnames in the Vagrantfile as required before

- **Otherwise**, start the VM using VirtualBox and login into it
  - User/password: vagrant/vagrant
  - Change the VM hostname and reboot it to changes take effect
    - sudo hostnamectl set-hostname **xxx**-iap2223

Lab. Performance evaluation of virtualization technologies

- **Geekbench**
  - Geekbench is a cross-platform benchmark that provides workloads that simulate real-world scenarios
    - It automatically uploads the results to the Geekbench Browser, so you need an active Internet connection on your computer
  - Chdir to Geekbench directory
  - Run the benchmark
    - ./geekbench_x86_64
  - When finished, a link with the results is provided
    - **You must include this link in your PDF report**
    - Visit the link to write down the results for single-core and multi-core performance
    - The output of the execution can be quite large, so it is enough to take an screenshot with only the last part of the output **showing the URL** for your results (see next slide)

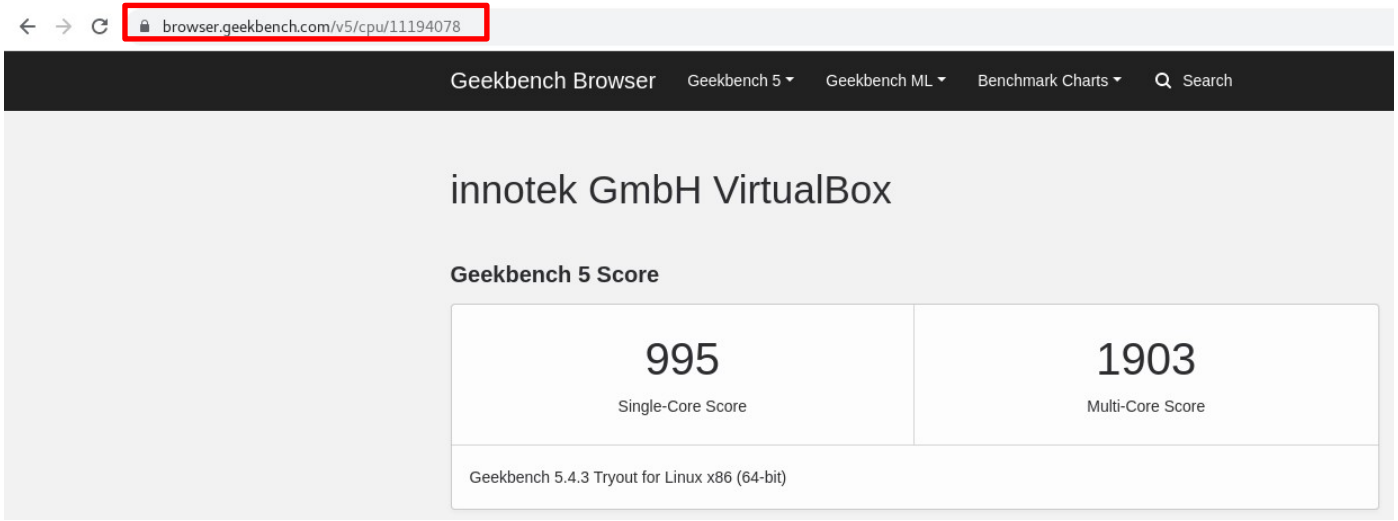
Lab. Performance evaluation of virtualization technologies

- **Geekbench**

```
  Running Speech Recognition
  Running Machine Learning


Uploading results to the Geekbench Browser. This could take a minute or two
depending on the speed of your internet connection.

Upload succeeded. Visit the following link and view your results online:

  https://browser.geekbench.com/v5/cpu/11194078

Visit the following link and add this result to your profile:

  https://browser.geekbench.com/v5/cpu/11194078/claim?key=402796

[vagrant@xxx-iap2223 Geekbench-5.4.3-Linux]$
```

You must include this link in your PDF

Be sure to show your **prompt** in all your screenshots



Lab. Performance evaluation of virtualization technologies

- **NAS Parallel Benchmarks (NPB)**
  - NPB are a set of paralell codes that mimics the computation and data movement from computational fluid dynamics applications
  - Chdir to NPB directory
  - We will use the **OpenMP implementation** of the NPB suite
    - Chdir to NPB3.4-OMP
  - **Before compiling, activate GCC 11 compiler**
    - To compile some NPB bencharmks, it is required at least GCC 9 and the default compiler in Rocky Linux 8.6 is GCC 8.5.0
    - To activate GCC11: scl enable gcc-toolset-11 bash
    - To check the version: gcc -v
  - Compile CG, BT and MG benchmarks using **CLASS B**
    - make cg CLASS=B
    - make bt CLASS=B
    - make mg CLASS=B
  - Binaries are located under the bin subdirectory

Lab. Performance evaluation of virtualization technologies

- **NAS Parallel Benchmarks (NPB)**
  - Run the benchmarks using **one thread**
    - **export OMP_NUM_THREADS=1**
    - ./bin/cg.B.x
    - ./bin/bt.B.x
    - ./bin/mg.B.x
  - When finished, results are provided in **Mop/s** and **runtime** (seconds)
    - Mop/s = Millions of operations per second
  - Run again the benchmarks using **two threads**
    - **export OMP_NUM_THREADS=2**
    - ./bin/cg.B.x
    - ./bin/bt.B.x
    - ./bin/mg.B.x
  - **When finished, write down all your results in Mops/ and seconds**

Lab. Performance evaluation of virtualization technologies

- **NAS Parallel Benchmarks (NPB)**
  - Example: NPB-OMP CG execution using one and two threads

```
CG Benchmark Completed.
Class           =                        B
Size            =                    75000
Iterations      =                       75
Time in seconds =                    46.72
Total threads   =                        1
Avail threads   =                        1
Mop/s total     =                 1170.99
Mop/s/thread    =                 1170.99
Operation type  =           floating point
Verification    =               SUCCESSFUL
Version         =                    3.4.2
Compile date    =              25 Nov 2021

Compile options:
   FC           = gfortran
   FLINK        = $(FC)
   F_LIB        = (none)
   F_INC        = (none)
   FFLAGS       = -O3 -fopenmp
   FLINKFLAGS   = $(FFLAGS)
   RAND         = randi8


Please send all errors/feedbacks to:

NPB Development Team
npb@nas.nasa.gov

[vagrant@xxx-iap2223 NPB3.4-OMP]$
```

```
CG Benchmark Completed.
Class           =                        B
Size            =                    75000
Iterations      =                       75
Time in seconds =                    24.57
Total threads   =                        2
Avail threads   =                        2
Mop/s total     =                 2227.09
Mop/s/thread    =                 1113.55
Operation type  =           floating point
Verification    =               SUCCESSFUL
Version         =                    3.4.2
Compile date    =              25 Nov 2021

Compile options:
   FC           = gfortran
   FLINK        = $(FC)
   F_LIB        = (none)
   F_INC        = (none)
   FFLAGS       = -O3 -fopenmp
   FLINKFLAGS   = $(FFLAGS)
   RAND         = randi8


Please send all errors/feedbacks to:

NPB Development Team
npb@nas.nasa.gov

[vagrant@xxx-iap2223 NPB3.4-OMP]$
```

Lab. Performance evaluation of virtualization technologies

- **STREAM**
  - STREAM is a simple synthetic benchmark designed to measure sustainable memory bandwidth and the corresponding computation rate for simple vector kernels
  - Examples:
    - Copy: a(i) = b(i))
    - Triad: a(i) = b(i) + c(i) * scalar
  - The STREAM source code uses a variable (**STREAM_ARRAY_SIZE**) to define the length of the arrays that are used in the tests
    - The arrays are of type ***double*** and STREAM uses three arrays (a,b,c)
    - Total memory required for this benchmark is 3*8*N bytes (N being STREAM_ARRAY_SIZE)
  - To effectively evalute the memory bandwitdh, **each array must be at least 4 times** the size of the **last level cache**
  - Default value for N (10M) is large enough for caches up to 20 MB

Lab. Performance evaluation of virtualization technologies

- **STREAM**
  - Chdir to STREAM directory
  - **If your cache size is <= 20 MB, compile the benchmark as follows:**
    - gcc -O3 **-fopenmp** stream.c -o stream
  - **Else (i.e your cache size is > 20 MB), calculate N as follows:**
    - N = (4 * CACHE_SIZE_IN_BYTES) / 8
      - E.g. for a cache size of 30 MB: N = (4 * $30*2^{20}$) / 8 = 15728640
    - Modify STREAM_ARRAY_SIZE at compile time
      - gcc -O3 **-fopenmp -DSTREAM_ARRAY_SIZE=N** stream.c -o stream
  - Run the benchmark using 2 threads
    - export OMP_NUM_THREADS=2 && ./stream
  - When finished, the best **bandwidths rates** are provided for each function
    - **Write down your results**

Lab. Performance evaluation of virtualization technologies

- **STREAM**

```
[vagrant@xxx-iap2223 STREAM]$ gcc -O3 -fopenmp stream.c -o stream
[vagrant@xxx-iap2223 STREAM]$ export OMP_NUM_THREADS=2 && ./stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 10000000 (elements), Offset = 0 (elements)
Memory per array = 76.3 MiB (= 0.1 GiB).
Total memory required = 228.9 MiB (= 0.2 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Number of Threads requested = 2
Number of Threads counted = 2
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 4806 microseconds.
   (= 4806 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           33437.4     0.004820     0.004785     0.004895
Scale:          22785.8     0.007051     0.007022     0.007103
Add:            24899.4     0.009676     0.009639     0.009705
Triad:          25375.2     0.009637     0.009458     0.010044
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
[vagrant@xxx-iap2223 STREAM]$
```

Lab. Performance evaluation of virtualization technologies

- **FIO**
  - FIO is a versatile I/O workload generator highly flexible that enables to replicate real-world environments (e.g. sequential/random read/write tests using different block sizes)
  - Four different tests will be performed
    - Sequential read & write performance
      - Sequential performance is key for large block sizes (e.g. 2 MB)
    - Random read & write performance
      - Random performance is key for small block sizes (e.g. 4 KB)
  - Chdir to FIO directory
  - Compile and install FIO as follows:
    - ./configure && make
    - sudo make install

Lab. Performance evaluation of virtualization technologies

- **FIO**
  - **Sequential read** performance test (file size 8GB, block size 2MB):
    - fio --ioengine=libaio --direct=1 --gtod_reduce=1 --name=test --filename=test **--bs=2M** --iodepth=1 **--size=8G** --runtime=120 **--rw=read** && rm test
  - **Sequential write** performance test (file size 8GB, block size 2MB):
    - fio --ioengine=libaio --direct=1 --gtod_reduce=1 --name=test --filename=test **--bs=2M** --iodepth=1 **--size=8G** --runtime=120 **--rw=write** && rm test
  - **Random read** performance test (file size 2GB, block size 4KB):
    - fio --ioengine=libaio --direct=1 --gtod_reduce=1 --name=test --filename=test **--bs=4k** --iodepth=64 **--size=2G** --runtime=120 **--readwrite=randread** && rm test
  - **Random write** performance test (file size 2GB, block size 4KB):
    - fio --ioengine=libaio --direct=1 --gtod_reduce=1 --name=test --filename=test **--bs=4k** --iodepth=64 **--size=2G** --runtime=120 **--readwrite=randwrite** && rm test
  - When finished, **bandwidth** and **IOPS** results are provided
    - IOPS = I/O Operations Per Second
    - **Write down your results for both metrics**

Lab. Performance evaluation of virtualization technologies

- **FIO**
  - Example: sequential read performance test

```
[vagrant@xxx-iap2223 fio-3.28]$ fio --ioengine=libaio --direct=1 --gtod_reduce=1 --name=test --filename=test --bs=2M
test: (g=0): rw=read, bs=(R) 2048KiB-2048KiB, (W) 2048KiB-2048KiB, (T) 2048KiB-2048KiB, ioengine=libaio, iodepth=1
fio-3.28
Starting 1 process
test: Laying out IO file (1 file / 8192MiB)
Jobs: 1 (f=1): [R(1)][100.0%][r=2434MiB/s][r=1217 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=7369: Thu Nov 25 12:30:18 2021
  read: IOPS=945, BW=1890MiB/s (1982MB/s)(8192MiB/4334msec)
    bw (  MiB/s): min=  516, max= 2472, per=98.64%, avg=1864.39, stdev=778.15, samples=8
    iops        : min=  258, max= 1236, avg=932.13, stdev=389.12, samples=8
  cpu           : usr=0.00%, sys=14.01%, ctx=12289, majf=0, minf=521
  IO depths     : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit     : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete   : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=4096,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency    : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
   READ: bw=1890MiB/s (1982MB/s), 1890MiB/s-1890MiB/s (1982MB/s-1982MB/s), io=8192MiB (8590MB), run=4334-4334msec

Disk stats (read/write):
    dm-0: ios=3970/0, merge=0/0, ticks=4132/0, in_queue=4132, util=97.60%, aggrios=16384/0, aggrmerge=1/0, aggrticks=
  sda: ios=16384/0, merge=1/0, ticks=6488/0, in_queue=6487, util=95.69%
[vagrant@xxx-iap2223 fio-3.28]$
```

Lab. Performance evaluation of virtualization technologies

- **FIO**
  - Example: random write performance test

```
[vagrant@xxx-iap2223 fio-3.28]$ fio --ioengine=libaio --direct=1 --gtod_reduce=1 --name=test --filename=test
te && rm test
test: (g=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=64
fio-3.28
Starting 1 process
test: Laying out IO file (1 file / 2048MiB)
Jobs: 1 (f=1): [w(1)][100.0%][w=37.8MiB/s][w=9688 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=7382: Thu Nov 25 12:33:08 2021
  write: IOPS=6966, BW=27.2MiB/s (28.5MB/s)(2048MiB/75259msec); 0 zone resets
    bw (  KiB/s): min=12223, max=40888, per=100.00%, avg=35773.90, stdev=6493.07, samples=117
    iops        : min= 3055, max=10222, avg=8943.45, stdev=1623.31, samples=117
  cpu           : usr=0.53%, sys=8.21%, ctx=502796, majf=0, minf=9
  IO depths     : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
     submit     : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete   : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.1%, >=64=0.0%
     issued rwts: total=0,524288,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency    : target=0, window=0, percentile=100.00%, depth=64

Run status group 0 (all jobs):
  WRITE: bw=27.2MiB/s (28.5MB/s), 27.2MiB/s-27.2MiB/s (28.5MB/s-28.5MB/s), io=2048MiB (2147MB), run=75259-752

Disk stats (read/write):
    dm-0: ios=0/530555, merge=0/0, ticks=0/183602, in_queue=183602, util=99.95%, aggrios=0/529949, aggrmerge=
  sda: ios=0/529949, merge=0/708, ticks=0/109335, in_queue=109336, util=99.91%
[vagrant@xxx-iap2223 fio-3.28]$
```

DOCKER

Lab. Performance evaluation of virtualization technologies

- If you do not have Docker in your system, install it:
  - https://docs.docker.com/desktop/#download-and-install
  - DO NOT install Docker on the Linux VM used before!
    - It should be installed on your computer to provide a fair comparison
- Test your Docker installation by executing a simple container
  - docker run --rm hello-world

```
[rober@oceania ~]$ docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:cc15c5b292d8525effc0f89cb299f1804f3a725c8d05e158653a563f15e4f685
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

Lab. Performance evaluation of virtualization technologies

- Download the Docker image from Aula CESGA
  - It contains all the needed software
- Import the image on your computer
  - docker load -i docker-rocky-8-mhpc.tar
- Run a Docker container and chdir to /root
  - docker run --rm **--hostname xxx-iap2223 --cpus=2 --cpuset-cpus="0,1" --memory=2G** -it rocky-8-mhpc
- **Notes on the container configuration**

  - It is **mandatory** to change the **hostname** of your container
    - Change **hostname** parameter in “docker run” by replacing **xxx** with the initials of your name and surname
  - Be sure that you give 2 CPU cores and 2 GB of memory to the container
    - If your CPU has SMT enabled (e.g. Intel HyperThreading), try to assign a **cpuset** which uses different physical cores
- Proceed as before to run all the benchmarks within the container

Lab. Performance evaluation of virtualization technologies

```
[rober@oceania ~]$ docker load -i docker-rocky-8-mhpc.tar
03644e7c508b: Loading layer [==================================================>]  752.6MB/752.6MB
Loaded image: rocky-8-mhpc:latest
[rober@oceania ~]$ docker image ls
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
rocky-8-mhpc            latest    e0171a938dbf   6 minutes ago   935MB
rockylinux/rockylinux   8.6       523ffac7fb2e   11 days ago     196MB
[rober@oceania ~]$ docker run --rm --hostname xxx-iap2223 --cpus=2 --cpuset-cpus="0,1" --memory=2G -it rocky-8-mhpc
[root@xxx-iap2223 /]# cd root/
[root@xxx-iap2223 ~]# ls
Geekbench-5.4.3-Linux  NPB3.4.2  STREAM  anaconda-ks.cfg  anaconda-post.log  fio-3.28  original-ks.cfg
[root@xxx-iap2223 ~]#
```