iocost & resctl-bench
What is a cgroup IO controller?

- What is cgroup?
  - A hierarchical grouping of processes in a system.

- What is IO?
  - Here, it means block IOs - the things that get written to and read from secondary storage devices like an SSD.

- What is controller?
  - A mechanism which attaches to the cgroup hierarchy and divvies up available system resources.
Why is IO control challenging?

- Conventionally available metrics are not descriptive enough
  - Bandwidth, IOPS, latencies interact in non-trivial ways which is difficult to describe succinctly.
  - Makes single-metric-based control difficult to configure. e.g. A given bandwidth config can be too high and too low at the same time for blk-throtl.

- SSDs can be very erratic
  - Periodic or unpredictable slowdowns or even stalls are not uncommon.
  - Difficult to tell much from specs or simple benchmarks.

- SSDs can be very performant
  - Can easily reach hundreds of thousands of operations per second. Can't do anything too expensive.

- Intertwined with the rest of the system
  - Priority inversions.
How does iocost solve the challenges?

- Each IO is assigned a cost according to a model
  - Currently linear model derived from offline benchmark is used.
  - Can be extended.
- Planning path allocates resources to different cgroups according to weights
  - Easy to configure across different classes of devices and workloads.
  - The slower and complex path which runs periodically at millisecond timescale.
- Issue path controls IOs in a cheap and scalable way
  - Control decisions are local to each cgroup without any cross-cgroup communication.
- Integrates with FS and MM to avoid priority inversions
  - Do first, charge later.
Does it work? (750K IOPS rand read)
Does it work? (2:1 latency sensitive random reads)
Deployment at Meta
Deployment at Meta

![Graph showing the number of container failures per day over a month. The x-axis represents time over one month, with 'Week 1' and 'Week 3' as landmarks. The y-axis represents the number of container failures, ranging from 0K to 24K. The graph shows a downward trend in failures from Week 1 to Week 3.]
The parameters

- iocost works well as long as it’s configured well

```bash
# cat /sys/fs/cgroup/io.cost.model
259:0 ctrl=user model=linear rbps=1965196223 rseqiops=197218 rrandiops=150355 wbps=937386870 wseqiops=11516 wrandiops=31462
# cat /sys/fs/cgroup/io.cost.qos
259:0 enable=1 ctrl=user rpct=0.00 rlat=1708 wpct=0.00 wlat=22167 min=100.00 max=100.00
```

- but that’s a lot of numbers
- The kernel tree has tools/cgroup/iocost_coef_gen.py script which does simple fio benchmarks.
  - But didn’t you say devices are erratic and simple metrics don’t capture the behavior well?
resctl-bench

- A scenario based benchmark which observes end-to-end behavior.
  - A latency sensitive and resource intensive main workload.
  - A memory and IO intensive adversarial workload.
  - Can you isolate?
- resctl-bench repeats the above scenario at different throttling point
  - Is isolation good enough at this throttling point?
- and produces a detailed report on the observed behavior and suggested parameters
[iocost-tune 2.2.5 result] 2022-06-29 17:45:06 - 17:45:06

System info: kernel="5.18.0-rc6"
   nr_cpus=52 memory=13.6G swap=4.5G swappiness=60
   mem_profile=16 (avail=12.4G share=12.0G target=11.0G)

IO info: dev=nvme0n1(259:0) model="WDC CL SN730 SDBQNTY-256G-2020" firmware ="11120120" size=238G
   iosched=none wbt=off iocost=on other=off
   iocost model: rbps=2406258697 rseqiops=129795 rrandiops=124482
      wbps=429280383 wseqiops=20317 wrandiops=14912
   iocost QoS: rputc=95.00 rlat=38732 wpct=95.00 wlat=626000 min=31.93 max=31.93

Solutions

=================

[naive] vrate=75-100, rpct=99, wpct=99
   info: scale=100.0% MOF=1.376@16 aMOF=1.174 aMOF-delta=0.204 isol-05=95.42%
   rlat: 50-mean= 141u 50-99= 606u 50-100= 2.7m 99-mean= 9.2m 99-99=40.3m 100-100= 306m
   wlat: 50-mean= 9.6m 50-99=43.2m 50-100=75.6m 99-mean=58.3m 99-99=96.6m 100-100= 297m
   model: rbps=2406258697 rseqiops=129795 rrandiops=124482 wbps=429280383 wseqiops=20317 wrandiops=14912
   qos: rpct=99.00 rlat=9225 wpct=99.00 wlat=58339 min=75.00 max=100.00

[bandwidth] aMOF=max-vrate
   info: scale=100.0% MOF=1.376@16 aMOF=1.174 aMOF-delta=0.204 isol-05=95.42%
   rlat: 50-mean= 141u 50-99= 606u 50-100= 2.7m 99-mean= 9.2m 99-99=40.3m 100-100= 306m
   wlat: 50-mean= 9.6m 50-99=43.2m 50-100=75.6m 99-mean=58.3m 99-99=96.6m 100-100= 297m
   model: rbps=2406258697 rseqiops=129795 rrandiops=124482 wbps=429280383 wseqiops=20317 wrandiops=14912
   qos: rpct=0.00 rlat=9225 wpct=8.00 wlat=58339 min=100.00 max=100.00

[isolated-bandwidth] (aMOF=right-max).clamp(isolation, bandwidth)
   [isolation] aMOF-delta=min
   info: scale=74.22% MOF=1.376@16 aMOF=1.340 aMOF-delta=0.044 isol-05=95.42%
   rlat: 50-mean= 141u 50-99= 606u 50-100= 2.7m 99-mean= 9.2m 99-99=40.3m 100-100= 306m
   wlat: 50-mean= 9.6m 50-99=43.2m 50-100=75.6m 99-mean=58.3m 99-99=96.6m 100-100= 297m
   model: rbps=1785940583 rseqiops=96335 rrandiops=92391 wbps=318614644 wseqiops=15079 wrandiops=11068
   qos: rpct=0.00 rlat=9225 wpct=0.00 wlat=58339 min=100.00 max=100.00
That sounds like a lot

- It is but we just need a few results per SSD model.
- We can collect and build a database.
- https://github.com/iocost-benchmark/iocost-benchmarks
Running resctl-bench

- Requires btrfs filesystem with swap on it.
- Can run on non-root device but better to run on root filesystem.
- Installable images are generated in
- Takes ~6 hours. Can be interrupted and restarted.
- Needs a few results to be merged for good graph fitting.
resctl-demo flasher
Reboot Into Firmware Interface

Boot in 4 s.
Choose disk to install resctl-demo

WARNING: the disk will be overwritten! Choose Quit to cleanly exit.

- /dev/sda  Samsung SSD 860 EVO M.2 250GB 232.9G S5GFN60N302600X
- /dev/sdb  Samsung SSD 870 EVO 1TB 931.5G S75BN9W7031902
- /dev/sdc  Samsung SSD 850 EVO 500GB 465.8G S3PTNF0J1445SL
- /dev/nume0n1 Samsung SSD 980 PRO 1TB 931.5G S5P2NU0W129061M

shell  Drop to a shell
quit  Quit (Shutdown Machine without modifications)

< OK >  < Cancel >
Choose disk to install resctl-demo

You have chosen to install resctl-demo to /dev/nvme0n1 (Samsung SSD 980 PRO 1TB 931.5G S5P2M00W129061M)
WARNING: the disk will be overwritten! Choose no to go back to the main menu.

< Yes >  < No >
resctl-demo
version: 20240313.185133.resctl-demo.ed0953e

Installing resctl-demo to /dev/nume0n1: do not turn your computer off.
You will be prompted to restart your computer after installation.

bmaptool: info: discovered bmap file '/mnt/flasher-storage/resctl-demo-image.img.bmap'
bmaptool: info: block map format version 2.0
bmaptool: info: 7812500 blocks of size 4096 (29.8 GiB), mapped 1097632 blocks (4.2 GiB or 14.0%)
bmaptool: info: copying image 'resctl-demo-image.img.gz' to block device '/dev/nume0n1' using bmap file 'resctl-demo-image.img.bmap'
bmaptool: info: 20% copied
Installation complete

Boot to installed image or shutdown

- pivot
- shutdown

Boot to installed image
Shut down the system
An SSH server is running on port 22.

resctl-demo login: demo (automatic login)

Welcome to the Resource Control Demo!
This test environment is based on Debian Bookworm.

To start the resource control demo, please run:
$ ./start-resctl-demo

To start the resource control benchmark, please run:
$ ./start-resctl-bench

Power off the instance when you are finished - `sudo poweroff`.

[  4.859660] nouveau 0000:09:00.0: AMD-Vi: Event logged [IO_PAGE_FAULT domain=0x0013 address=0xfed5000 flags=0x0000]
[  4.872367] nouveau 0000:09:00.0: disp: chid 0 stat 00001000 reason 1 [PUSHBU]
[  4.885144] nouveau 0000:09:00.0: disp: chid 1 stat 00001000 reason 1 [PUSHBU]

Operating on block device model: Samsung SSD 980 PRO 1TB

demo@resctl-demo:$ [  7.347499] nouveau 0000:09:00.0: DRM: core notifier timeout
[  9.347521] nouveau 0000:09:00.0: DRM: core notifier timeout
[ 11.347522] nouveau 0000:09:00.0: DRM: wndu-0: timeout
demo@resctl-demo:$ ./start-resctl-bench
Starting resctl-bench...
hash_size=1.2M rps_max=2000 mem_actual=18.8G chunk_pages=25 fake_cpu_load
mem-profile 16G memavail 14.5G, memshare 12.0G, mem_target 11.0G
workload_mem_low=9.06G ballon_size=2.5G
Applying hashd parameters
hash_size=1.2M rps_max=2000 mem_actual=18.8G chunk_pages=25 fake_cpu_load
protection: Stabilizing hashd at 100.0% for Probing 18.06 (1/10)
protection: Running for 15.0s to measure the baseline
protection: Starting memory hog
protection: Probing 18.06 (1/10) failed after 0.17s (hashd failed while waiting, hashd-A Failed)
mem-profile 16G memavail 14.5G, memshare 12.0G, mem_target 11.0G
workload_mem_low=9.06G ballon_size=2.5G
Applying hashd parameters
hash_size=1.2M rps_max=2000 mem_actual=17.9G chunk_pages=25 fake_cpu_load
protection: Stabilizing hashd at 100.0% for Probing 17.9G (2/10)
protection: Running for 15.0s to measure the baseline
protection: Starting memory hog
protection: Probing 17.9G (2/10) failed after 0.17s (hashd failed while waiting, hashd-A Failed)
mem-profile 16G memavail 14.5G, memshare 12.0G, mem_target 11.0G
workload_mem_low=9.06G ballon_size=2.5G
Applying hashd parameters
hash_size=1.2M rps_max=2000 mem_actual=17.0G chunk_pages=25 fake_cpu_load
protection: Stabilizing hashd at 100.0% for Probing 17.0G (3/10)
protection: Running for 15.0s to measure the baseline
protection: Starting memory hog
protection: Probing 17.0G (3/10) failed after 0.17s (hashd failed while waiting, hashd-A Failed)
mem-profile 16G memavail 14.5G, memshare 12.0G, mem_target 11.0G
workload_mem_low=9.06G ballon_size=2.5G
Applying hashd parameters
hash_size=1.2M rps_max=2000 mem_actual=16.0G chunk_pages=25 fake_cpu_load
protection: Stabilizing hashd at 100.0% for Probing 16.0G (4/10)
protection: Running for 15.0s to measure the baseline
protection: Starting memory hog
protection: Memory hog terminated after 6.0s, Probing 16.0G (4/10) finished
protection: Probing 16.0G (4/10) succeeded, isol-95=92.50% >= 90.00%
locost-qos011: Running storage benchmark with QoS parameters:
locost-qos011: rps=95.0 rlat=16776 upct=95.0 wlat=71959 [min=120.28] [max=120.28]
Applying locost parameters
model: rbps=2540000000 sseqiops=364653 randiops=304477 ubps=2314893629 useqiops=66718 wrandiops=84113
gos: rps=95.0 rlat=18776 upct=95.0 wlat=71959 min=120.28 max=120.28
workload_mem_low=9.06G ballon_size=2.5G
storage: Measuring supportable memory footprint and IO latencies (1/1)
rd-hashd-bench [INFO] [0/5] lat: 72.1 rps:1559.5 slope: +0.01% error_slope: +0.39%
6 hours later
bandwidth: Isolatable memory size is 13.08% < supportable, sizing may be difficult.
* isolated-bandwidth: Isolatable memory size is 9.18% < supportable, sizing may be difficult.

tar: Removing leading '/' from member names
resctl-bench finished successfully!
Results saved to USB stick as Samsung_SSD_980_PRO_1TB/resctl-bench-result_2024_03_15-04_01_20

cp: error copying 'resctl-benchmark.tar.gz' to '/mnt/results/logs/resctl-benchmark.tar.gz': No space left on device

[24208.423710] INFO: task umount:96451 blocked for more than 122 seconds.
[24208.430870] Tainted: G E 5.10.0-rc6 #1
[24208.437014] "echo 0 > /proc/sys/kernel/hung_task_timeout_secs" disables this message.
[24331.393663] INFO: task umount:96451 blocked for more than 245 seconds.
[24331.310000] Tainted: G E 5.10.0-rc6 #1
[24331.316966] "echo 0 > /proc/sys/kernel/hung_task_timeout_secs" disables this message.
[24454.183632] INFO: task umount:96451 blocked for more than 368 seconds.
[24454.190826] Tainted: G E 5.10.0-rc6 #1
[24454.196992] "echo 0 > /proc/sys/kernel/hung_task_timeout_secs" disables this message.
demo@resctl-demo:$
Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue.

Go to Labels

4 Open  47 Closed

- CI is broken  #83 opened last month by eywler
- Image is broken  #82 opened last month by eywler
- resctl-demo dependencies need to be updated  #81 opened on Jan 31 by eywler
- Investigate making it easy for Fedora folks to run locost-benchmark and upload models  #76 opened on Sep 6, 2023 by eywler

ProTip! Adding notlabel will show everything without a label.
Issue: Benchmark submission

Submit an iocost benchmark. If this doesn't look right, choose a different type.

Add a title

Samsung 970 PRO

Add a description

1. Update the issue title with some information about the device you are submitting for.
2. Attach your benchmark in `.json.gz` format by dragging & dropping onto this issue.
3. Submit issue.
4. Await response from the github actions bot!

```
(resctl-bench-result_2024_03_15-04_01_20.json.gz)
```

Remember, contributions to this repository should follow our Github Community Guidelines.
htejun commented 5 minutes ago

resctl-bench-result_2024.03.15-04.01.20.json.gz

htejun added the benchmark-submission label 5 minutes ago

github-actions bot pushed a commit that referenced this issue 4 minutes ago

Automated update from issue 84 #85

github-actions bot mentioned this issue 4 minutes ago

Automated update from issue 84 #85

You can find the PDFs for the new merges here: https://github.com/iocost-benchmark/iocost-benchmarks/actions/runs/8297951627
**Links**

- iocost paper

- resctl-demo and resctl-bench
  - [https://github.com/facebookexperimental/resctl-demo](https://github.com/facebookexperimental/resctl-demo)

- resctl-bench result repository
  - [https://github.com/iocost-benchmark/iocost-benchmarks](https://github.com/iocost-benchmark/iocost-benchmarks)