CERN openlab

Development of an HPC benchmark

María Belén Guaranda

Supervisors: David Southwick, Maria Girone

summer 2022

Why benchmarking?

Identify potential areas of improvement

Better resource allocation

Informed architectural decisions

The concept

program.py

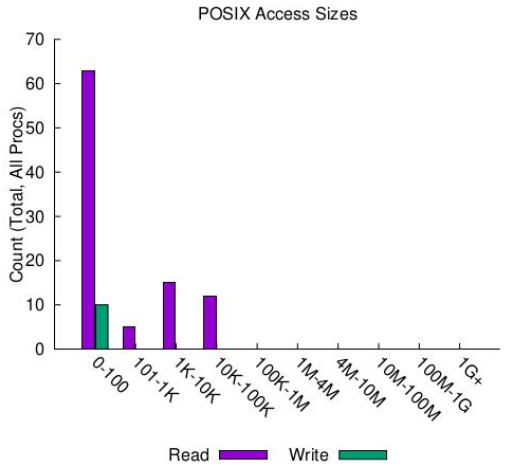
I/O reqs

bandwidth information to diagnose

I/O characterization

Darshan

- HPC I/O profiling characterization tool.
- Output is a binary file.



File Count Summary
(estimated by POSIX I/O access offsets)

type	number of files	avg. size	max size
total opened	45	7.4K	46K
read-only files	42	7.9K	46K
write-only files	1	390	390
read/write files	0	0	0
created files	1	390	390

Example

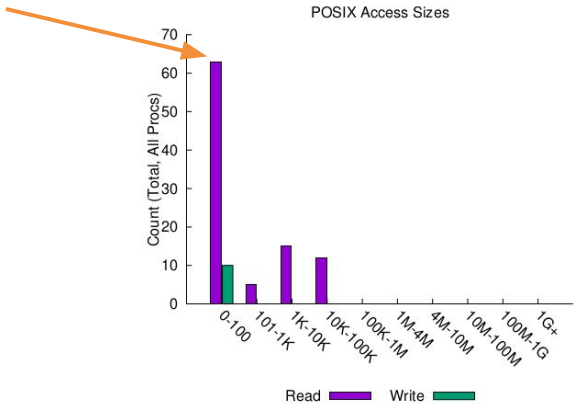
```
#<module>        <rank>  <record id>      <counter>       <value> <file name>      <mount pt>      <fs type>
650 POSIX   0        15483593433758586291     POSIX_RENAME_TARGETS    0       /home/belen/github/cern/test_dir/output.9       /       ext4
651 POSIX   0        15483593433758586291     POSIX_RENAMED_FROM      0       /home/belen/github/cern/test_dir/output.9       /       ext4
652 POSIX   0        15483593433758586291     POSIX_MODE      0       /home/belen/github/cern/test_dir/output.9       /       ext4
653 POSIX   0        15483593433758586291     POSIX_BYTES_READ        14      /home/belen/github/cern/test_dir/output.9       /       ext4
655 POSIX   0        15483593433758586291     POSIX_MAX_BYTE_READ     13      /home/belen/github/cern/test_dir/output.9       /       ext4
656 POSIX   0        15483593433758586291     POSIX_MAX_BYTE_WRITTEN  0       /home/belen/github/cern/test_dir/output.9       /       ext4
```

After using darshan-parser utility, we get a text file.

I/O characterization

Darshan's counters

- POSIX_ACCESS1_ACCESS



- POSIX_CONSEC_WRITES (or reads)
- POSIX_BYTES_READ (or write)

Benchmark

IOR/mdTest

- Tests performance of parallel file systems using various access patterns.

Example

```
# Total file size is 16GiB
srun ior -t 16m -b 64m -s 16 -F -C -e -i 5 -o /mnt/ceph/users/mbguarandacabezas/ior_output
```

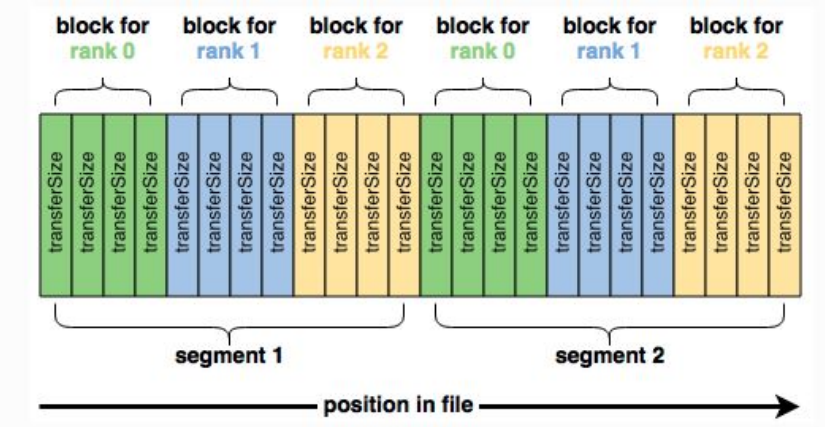
```
Results:

access    bw(MiB/s)  IOPS       Latency(s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)   close(s)   total(s)   iter
------    ---------  ----       ----------  ---------- ---------  --------   --------   --------   --------   ----
write     7011       438.57     0.463954    65536      16384      0.008900   2.33       0.707985   2.34       0
read      6176       386.27     0.533662    65536      16384      0.180169   2.65       0.791301   2.65       0
remove    -          -          -           -          -          -          -          -          0.107673   0
write     7942       496.86     0.475753    65536      16384      0.027062   2.06       0.450860   2.06       1
read      5923       370.38     0.453971    65536      16384      0.099232   2.76       1.43       2.77       1
remove    -          -          -           -          -          -          -          -          0.102075   1
write     8014       501.37     0.442369    65536      16384      0.151156   2.04       0.321556   2.04       2
read      6044       377.98     0.504938    65536      16384      0.125744   2.71       1.17       2.71       2
remove    -          -          -           -          -          -          -          -          0.102360   2
write     6909       432.18     0.483409    65536      16384      0.009997   2.37       0.763071   2.37       3
read      5654       353.66     0.628601    65536      16384      0.215685   2.90       1.24       2.90       3
remove    -          -          -           -          -          -          -          -          0.114988   3
write     2820.63    176.35     0.453071    65536      16384      0.265339   5.81       4.11       5.81       4
read      6259       391.42     0.540062    65536      16384      0.179251   2.62       1.21       2.62       4
remove    -          -          -           -          -          -          -          -          0.115358   4
Max Write: 8014.41 MiB/sec (8403.72 MB/sec)
Max Read:  6258.60 MiB/sec (6562.61 MB/sec)

Summary of all tests:
Operation   Max(MiB)   Min(MiB)  Mean(MiB)     StdDev   Max(OPs)   Min(OPs)  Mean(OPs)     StdDev    Mean(s) Stonewall(s) St
b$
write        8014.41    2820.63    6539.55    1914.80     500.90     176.29     408.72     119.67    2.92480         NA
671088$
read         6258.60    5654.25    6011.10     212.01     391.16     353.39     375.69      13.25    2.72909         NA
671088$
Finished            : Fri Jul 29 05:59:32 2022
```


6

IOR parameters



block size * number of process (or ranks) * number of segments = total bytes read or written.

- Number of tasks (processes) = 1 ⟶ each segment has **one** block
- Transfer size: arithmetic **mean** of POSIX_ACCESS1_ACCESS
- Block size: **mean** of consecutive ops * transfersize
- Number of segments: proportional to number of files

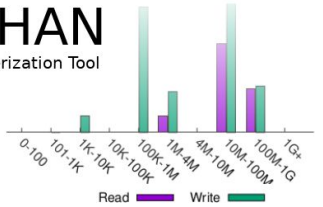
Tests setup

- Ceph filesystem of 1TB
- Python program
- Broadwell node connected over 10Gb/s Ethernet to the central storage resources.
- Python line profiler

FLATIRON INSTITUTE

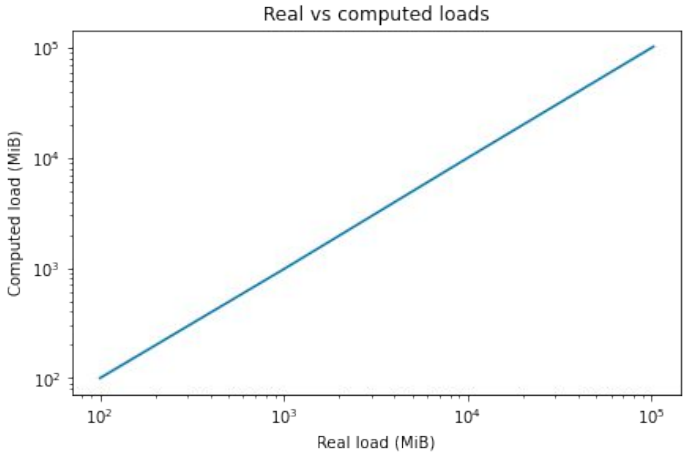
slurm workload manager

DARSHAN
HPC I/O Characterization Tool

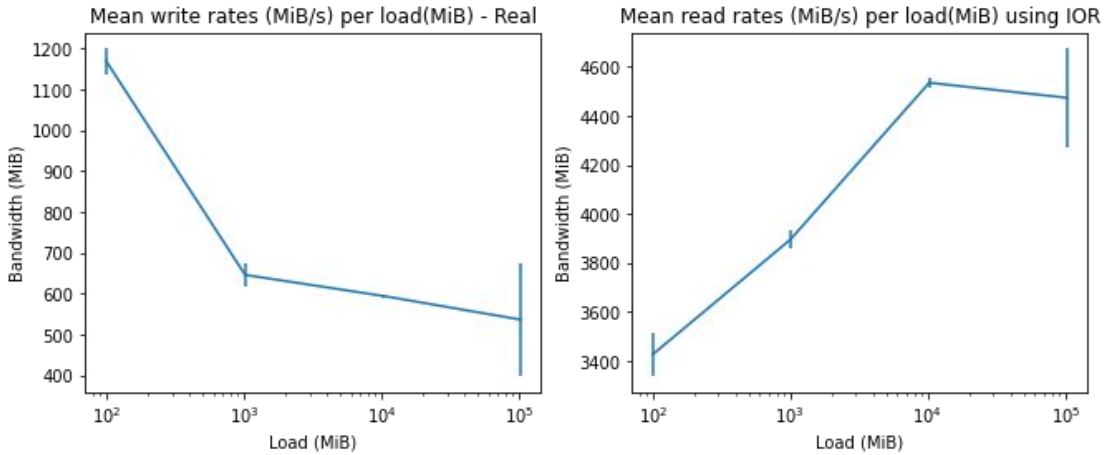


Tests results

Read



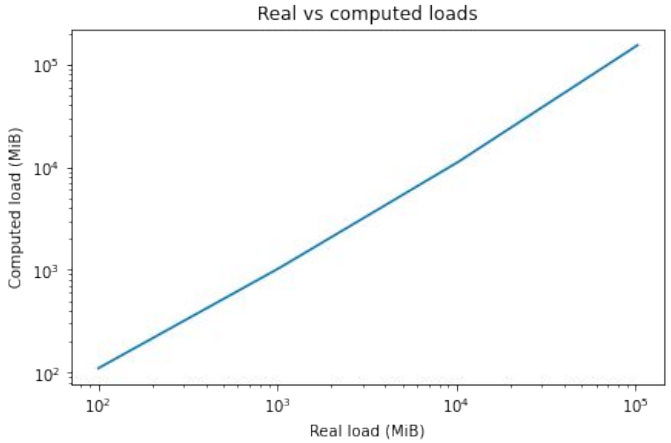
Total computed load is approximately the same as the original one.



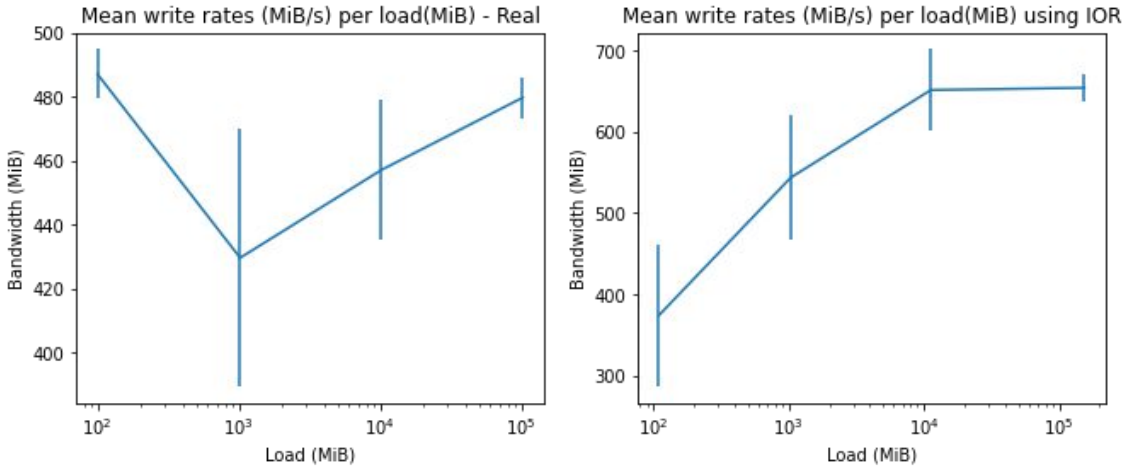
Bandwidths are different in trend; overestimated.

Tests results

Write



Total computed load is approximately the same as the original one.



Bandwidths have different trends; over and underestimated.

Conclusions & future work

🕐 It takes a lot of time to test.

📄 Documentation can be tricky.

🔬 In general, more experiments need to be done:

- On multiprocess programs
- Use more nodes
- Monitor cache

Thanks

David Southwick, Maria Girone, HPC team at CERN

mguarand@fiec.espol.edu.ec

Appendix

IOR parameters

Number of segments:

or written

```
excess_io = (blocksize * total_files) - bytes_read
excess_segments = floor(excess_io/blocksize)
segment_num = total_files - excess_segments
```

Semantics of segment is not well defined.