Irène Joliot-Curie
Laboratoire de Physique des 2 Infinis

Experience with HepScore at IJCLAB

Vamvakopoulos Emmanouil

HEPscore Workshop@CERN

19 Sep 2022

IJCLab
Irène Joliot-Curie
Laboratoire de Physique des 2 Infinis

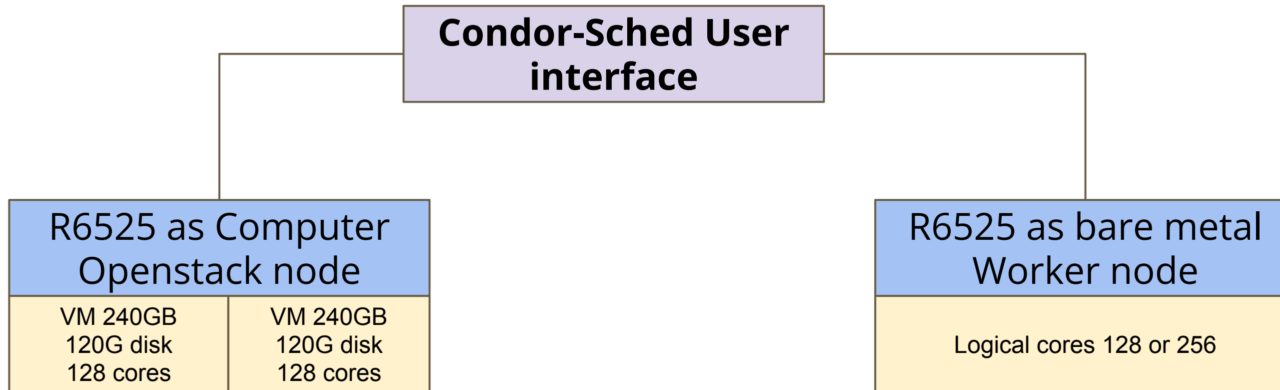
Usage HEP Benchmark Suite and HEPscore

- Participation in *Task Force measurement campaign*, since June 2021
- Run hepscore on various hardware and virtual machines
- Obtain the necessary site's license to run HEPSPEC06 and SPEC2017

Hardware under tests

- Dell PowerEdge R6525 (2x) AMD EPYC 7702 64-Core @2.0 GHz - 512 GB Ram
  - And virtual machines on EPYC hypervisor (128 vcores 256G RAM)
- Dell PowerEdge FC640 (2x) Intel Xeon Silver 4216 CPU @ 2.10GHz - 128GB Ram
- New deliveries of hardware
  - HP DL365 with AMD EPYC 7702 64-Core and 512GB

Environment of tests: A Tandem configuration



- No Numa node take into account (for the moment)
- We could have more than 1 node per type

Run of the Campaign

IJCLab Irène Joliot-Curie
Laboratoire de Physique des 2 Infinis

- Run the tests in steps (script by script) - with singularity
  - Minor change of script due to condor scratch path
  - Sanity check the wallclock time, job efficiency and I/O Load of the server
  - Sometimes we use the prmon (stage-in/out via condor), perf and flame graphs [*] for further validation of POWHEG benchmark component I/O performance (we saw a large system I/O on bare metals)
- Use a particular certificate (e.g. CERN CA) to push the results to CERN's Active@MQ
- Enthusiastic with the data population mechanisms.
- The most difficult part on servers' benchmarking is the allocation of the hardware/conditions and the population/aggregation of the results.
- Elasticsearch and Kibana looks decent machinery for data process and final visualization.

[*] https://www.brendangregg.com/flamegraphs.html

Conclusions

- We are running smoothly the benchmark components during the measurement campaign.
- We are enthusiastic ! with the data population and aggregation mechanism.
- If we have in the future Jupyter notebook direct access to data will be very helpful for custom graphs and better inspection of the data.
- Sanity check of the wallclock, job efficiency and system I/O are important to detect corner cases in hardware configuration, runtime environment condition or benchmark nature (e.g. cpu bound or I/O bound).

Backup slides ...

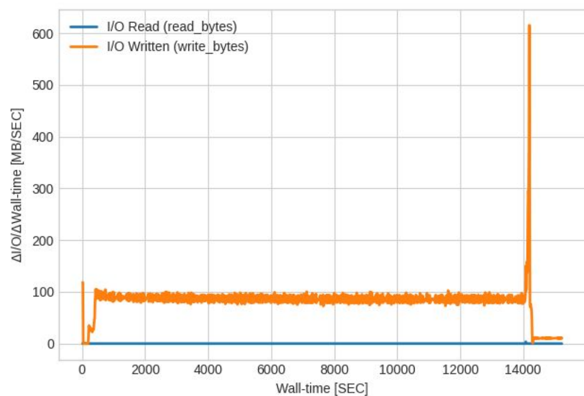
Atlas-gen-bmk based on POWHEG event generator

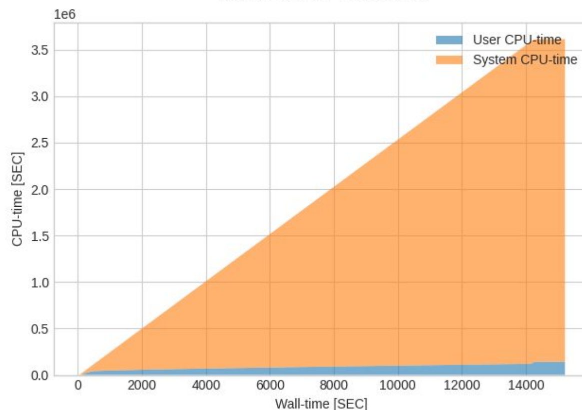
IJCLab Irène Joliot-Curie
Laboratoire de Physique des 2 Infinis

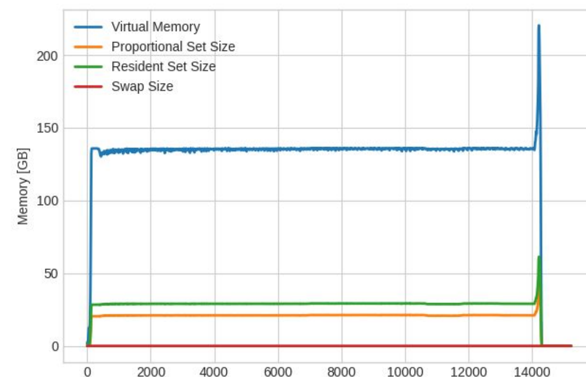
- Run the benchmark component alone
- Image at docker://gitlab-registry.cern.ch/hep-benchmarks/hep-workloads/atlas-gen-bmk'
- Use condor to dispatch the job and record syscpu and user cpu time
- Monitoring the job with prmon
- Use the linux perf tool and Flame Graphs in order to find the expensive syscalls
    - (e.g. perf record -F 99 -a -g -- sleep 240)

R6525 as bare metal

Scratch on RAID1 (Perc 745p)

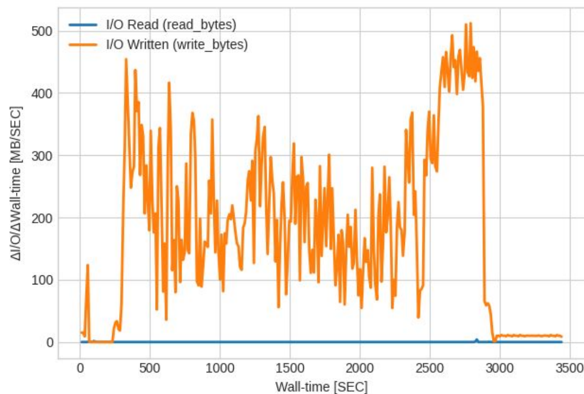


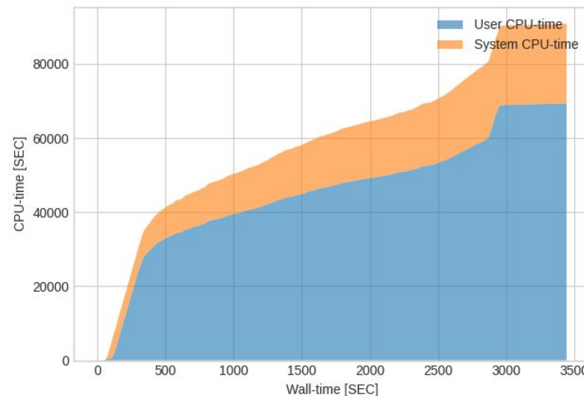


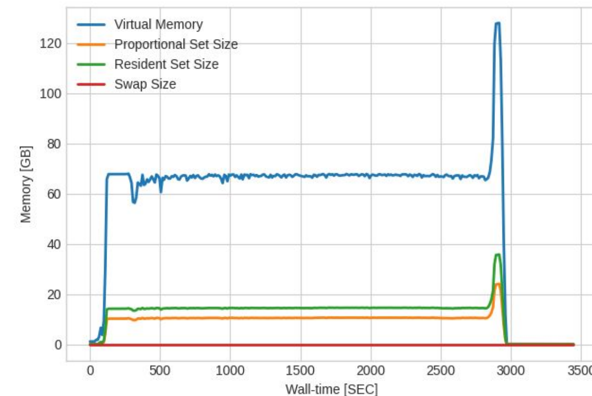


- Wallock of htcondor job 15131 sec
- Job efficiency 90% and system cpu time 96%
- Memory consumptions look ok, pps < 25GB
- Work flow exhibits write operations, suspicious flat write rate ~100MB/sec
- Score (avg) 16.37

2xVM 128 vCPU/128GB on R6525
as hypervisor Scratch on RAID1 (Perc 745p)



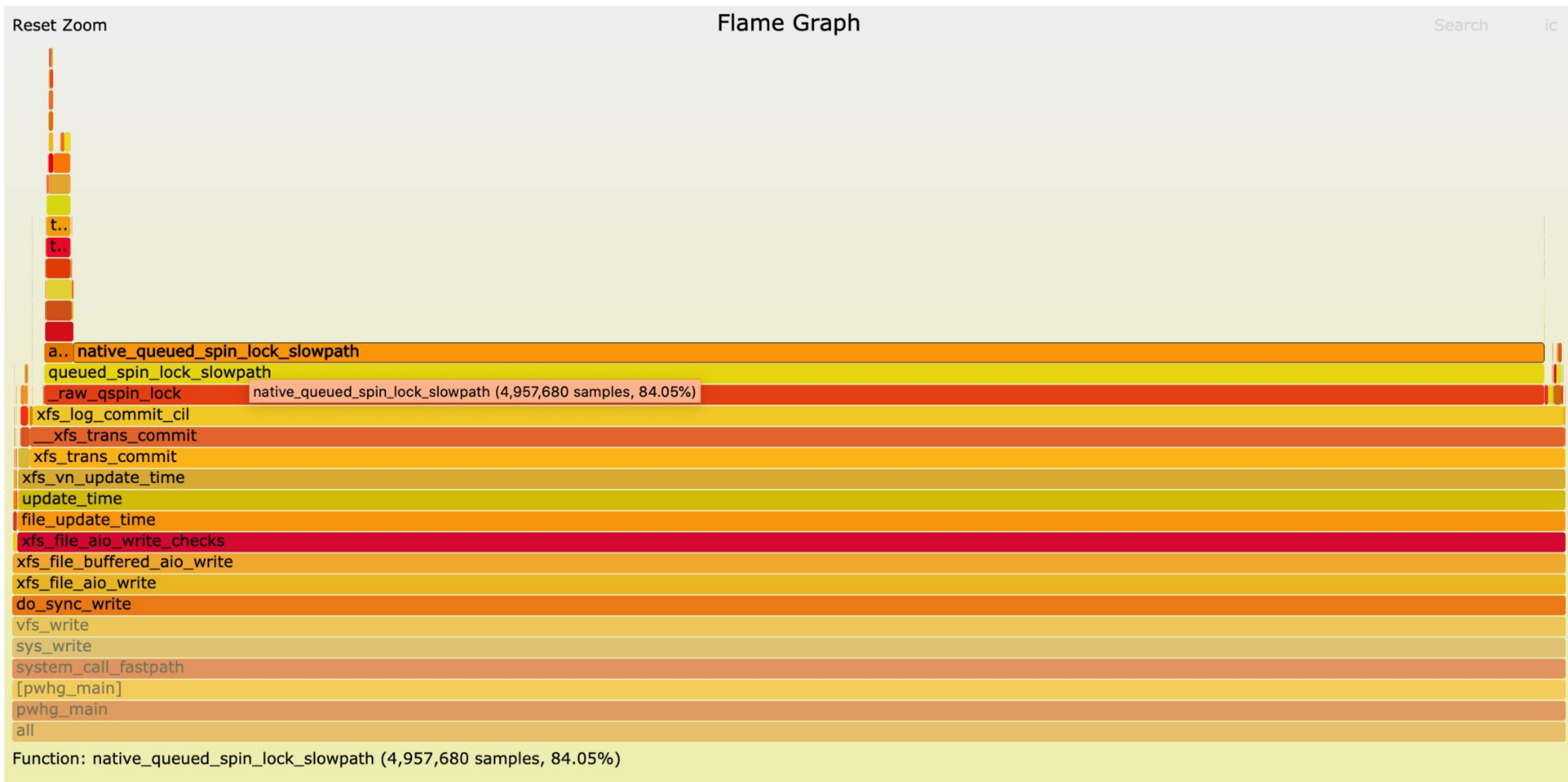




- Wallock of htcondor job 3445 sec
- Job efficiency 19% and system cpu time 23%
- Memory consumptions look ok pps < 20GB
- Work flow exhibits on write operation, suspicious oscillated write rate
- Score (avg) 18.83 (?)

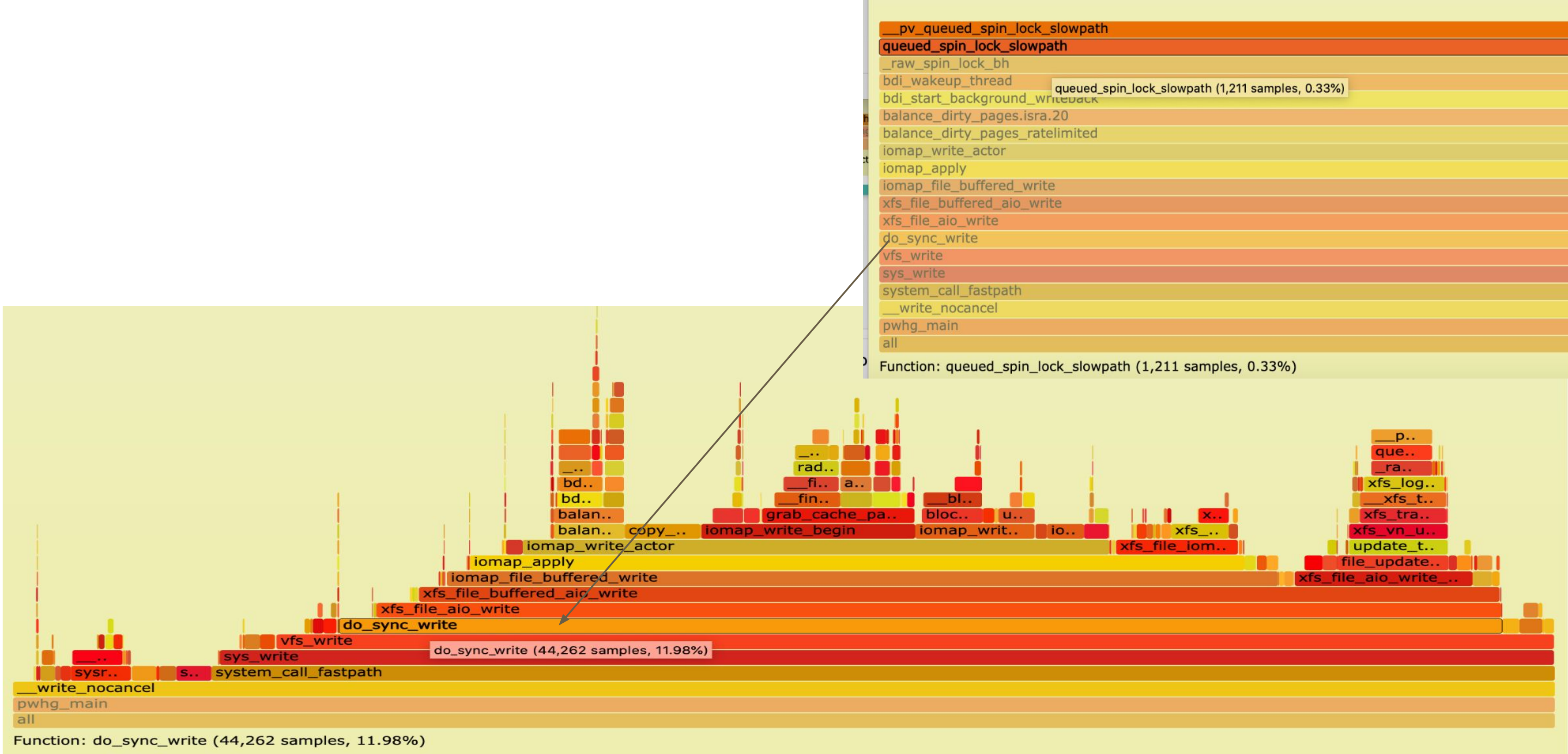
Flame Graph for R6525 as bare metal

Scratch on RAID1 (Perc 745p)



Flame graphs on 2xVM 128 vCPU/128GB on R6525

as hypervisor Scratch on RAID1 (Perc 745p)



Conclusions on POWHEG I/O

- Work load Atlas-gen-bmk based on POWHEG cause a spinlocks congestion on backlog XFS call during the write access pattern.This contention drives to very high System cpu utilization up to 96%, I/O write rate appears saturate for the real hardware.
- On Virtual environment we do not see the same behaviour as paravirtualized spin lock have different virtualization-friendly implementation (for example, block the virtual CPU rather than spinning).
- Same behaviour could reproduce via a xfs loop file fs on top of tmpfs. Thus is not hardware dependency
- We check the same for a ext4 loop file fs on top of tmpfs. Thus is not hardware dependency, we do not have any spinlocks contention but the system cpu time is quite high up to 87%
- We consider that the write access pattern of Atlas-gen-bmk based on POWHEG should be analyzed and understand better the interplay with XFS and ext4 filesystem calls
- We can check again on non journal FS

HEPSCORE details

- Installation on the fly with hep-benchmark-suite -wheels-py37-v2.1
- Hepscore v1.2 with default configuration
- 3 interactions per test component
- Singularity-3.7.3-1
- cvmfs-2.8.1-1.el7.x86_64
- Kernel-3.10.0-1160.25.1.el7.x86_64
- Results are published to cern MQ + ES/Kibana (usage of plain X509 key/certificate)

Paravirtualized spinlocks as not identical implementation with the original kernel spinlocks

CONFIG_PARAVIRT_SPINLOCKS=y, Paravirtualized spinlocks allow a pvops backend to replace the spinlock implementation with something virtualization-friendly (for example, block the virtual CPU rather than spinning). It has a minimal impact on native kernels and gives a nice performance benefit on paravirtualized KVM / Xen kernels. If you are unsure how to answer this question, answer Y.