HEP Benchmarks

High-Energy Physics workloads as benchmarks of computing architectures

D. Giordano (CERN / IT-CM-RPS)

ROOT PPP
25 June 2020
Intro

- Presentation largely based on recent report at HSF/WLCG Workshop
  - [https://indico.cern.ch/event/908146/timetable/#sc-7-1-hep-benchmarks-for-cpus](https://indico.cern.ch/event/908146/timetable/#sc-7-1-hep-benchmarks-for-cpus)

- Focus
  - Upcoming “remote” Summer Student activity
  - Motivations of HEP Benchmarks
  - Technical aspects
Summer Student project

- Mainly proposed by Xavi, Maria, Axel
  - Analysis benchmarks for grid and HPC
    - CERN measures the performance of computers based on typical computing patterns that happen “in real life”. One such pattern is analysis. This project proposes to extract existing example analyses to create a benchmarking utility for computers or HPC systems. This utility can then serve to describe a system’s performance for running typical analyses. Example analyses already exist in the context of the main analysis software of high energy physics, ROOT (https://root.cern), for instance leveraging CERN’s open data. The student would convert such analyses into a stand-alone program, and make it accessible as a benchmark.

- Cover one of the use cases of the HEP Benchmark Suite
  - Reason why I’m here 😊
An idea on hold since quite some time

- Danilo P. proposed this synergy in Feb 2019 at the HEPiX Benchmarking WG meeting

We want to improve the understanding of ROOT parallelised analysis in the O(100) cores regime

- We need to
  a. Consolidate, automate and expand our benchmarking suite (well understood in the O(10) cores regime)
  b. Have fat servers at disposal, profile and analyse

1) Can our benchmarks be useful in a context which is more general than ROOT?
2) Can we get help to improve our benchmarking suite?
3) Can we get help to run on extreme machines?

- https://indico.cern.ch/event/782598/timetable/
HEP Benchmarks

- Why are we doing this activity?
  - WLCG has to abandon (at some point in the future) HS06
    • EoL support, targets only CPUs, scales well with HEP WLs mainly on Intel x86 CPUs

- Transition to a **field-specific** benchmark (**HEPscore**) is demonstrated
  - The HEPiX Benchmarking WG has extensively documented design, build/run process, code and results

- **Opportunity** for the WLCG community to review the concepts and challenges of pledging, accounting, procurement, and **define the policies**
  - Technically several pledging/accounting scenarios are possible ⇒ Policies shall drive implementations

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>HS06</th>
<th>New Benchmark</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86 CPUs (y. 2010-2020)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>New CPUs (AMD) and/or arch (ARM/..)</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>New Exp Sw</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>CPU + GPU/FPGA/…</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
HEP CPU benchmarks evolution

1980's
MIPS (M Instr Per Sec)
VUPS (VAX units)
CERN units

1990’s – 2000’s
SI2k (SPEC INT 2000)
INTEGER benchmarks
200 MB footprint

2009
HS06 (SPEC CPU 2006 all_cpp)
INTEGER + FP benchmarks
1 GB footprint/core
32-bit
x86 servers
single-threaded/process on multi-core

2019
2 GB footprint/core (or more)
64-bit
multi-threaded, multi-process
multi-core, many-core
vectorization (SSE, … AVX512)
x86 servers, HPCs
ARM, Power9, GPUs…

HEP software (and computing) evolves… so shall do HEP CPU benchmarks!

Requirements:

– Probe the compute resources as the HEP applications do (not trivial!)
– Include heterogeneous resources
– Summarize performance using a single number, at least for accounting purposes
Benchmarking CPUs using HEP workloads

By construction, using HEP workloads directly is guaranteed to give

- A score with **high correlation** to the throughput of HEP workloads
- A CPU usage pattern that is similar to that of HEP workloads

“The first step in performance evaluation is to select the right measures of performance, the right measurement environments, and the right techniques.”

Criteria to build the HEP Benchmarks

- **Reproducibility** of results
  - Run the same processing sequence
    - same configuration, random seeds, input data

- **Robustness** of the running application
  - Do not fail, and notify in case of failures

- **Usability**
  - Especially outside the restricted group of experts

- **Portability**
  - Adopting container technology (Docker and Singularity so far)

- **Traceability** of the build process
  - Experiment sw, data, configuration
  - Images are built, tested and distributed via gitlab
HEP Benchmarks project

Three components [https://gitlab.cern.ch/hep-benchmarks](https://gitlab.cern.ch/hep-benchmarks)

- **HEP Workloads, HEP Workloads GPU (new)**
  - Common build infrastructure
  - Individual HEP workloads

- **HEP Score**
  - Orchestrate the run of a series of HEP workloads
  - Compute & Report the HEPscore value
    - “Single-number” benchmark score

- **HEP Benchmark Suite**
  - Automate execution of multiple benchmarks
    - HEPscore, SPEC CPU2017, HS06, …
  - Publish results
    - Simplify the sharing, tracking and comparison of results
HEP Workloads

- **Standalone containers** encapsulating **all and only** the dependencies needed to run each workload as a benchmark
  - Runs the Experiment executable with a configurable number of threads (MT) or processes (MP)

- Components of each HEP Workload
  - SW repository (OS and CVMFS) & Input data
  - Orchestrator script (benchmark driver)
    - Sets the environment, runs (many copies of) the application, parses the output to generate scores (json)

- All HEP workload types are currently available as **container images** in gitlab-registry, with more than one Experiment code per workload type
  - Run each workload via a single command line:
    > docker run $IMAGE_PATH
  - Standalone docker containers available in gitlab registry
Individual HEP workload container images are built, tested and distributed via gitlab

- Enabling technology: cvmfs tracing and export of all and only the libraries accessed by the workload

Can be executed both via Docker and Singularity
Extensive validation process

- Validating reproducibility, robustness, run duration, disk space needed
- Continuously running in a number of virtual & physical machines
- Evaluated a different number of events per WL to shorten the runtime

Current default configuration (still under study)

<table>
<thead>
<tr>
<th>WL</th>
<th># threads or proces. (default)</th>
<th># Evts/thread (default)</th>
<th>Duration of a single WL run on ref machine [hh:mm]</th>
<th>Wdir size (per running copy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas gen</td>
<td>1 (SP)</td>
<td>200</td>
<td>~12</td>
<td>50MB</td>
</tr>
<tr>
<td>Atlas sim</td>
<td>4 (MP)</td>
<td>10</td>
<td>~1:32</td>
<td>100 MB</td>
</tr>
<tr>
<td>CMS gen-sim</td>
<td>4 (MT)</td>
<td>20</td>
<td>~0:15</td>
<td>70 MB</td>
</tr>
<tr>
<td>CMS digi</td>
<td>4 (MT)</td>
<td>50</td>
<td>~0:09</td>
<td>400 MB</td>
</tr>
<tr>
<td>CMS reco</td>
<td>4 (MT)</td>
<td>50</td>
<td>~0:15</td>
<td>100 MB</td>
</tr>
<tr>
<td>LHCb gen-sim</td>
<td>1 (SP)</td>
<td>5</td>
<td>~0:40</td>
<td>15 MB</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td>~3:30</td>
<td></td>
</tr>
</tbody>
</table>

Atlas gen - sim
- # threads or proces. (default): 1 SP
- # Evts/thread: 200
- Duration of a single WL run on ref machine: ~12
- Wdir size: 50MB

Atlas sim
- # threads or proces. (default): 4 MP
- # Evts/thread: 10
- Duration of a single WL run on ref machine: ~1:32
- Wdir size: 100 MB

CMS gen-sim
- # threads or proces. (default): 4 MT
- # Evts/thread: 20
- Duration of a single WL run on ref machine: ~0:15
- Wdir size: 70 MB

CMS digi
- # threads or proces. (default): 4 MT
- # Evts/thread: 50
- Duration of a single WL run on ref machine: ~0:09
- Wdir size: 400 MB

CMS reco
- # threads or proces. (default): 4 MT
- # Evts/thread: 50
- Duration of a single WL run on ref machine: ~0:15
- Wdir size: 100 MB

LHCb gen-sim
- # threads or proces. (default): 1 SP
- # Evts/thread: 5
- Duration of a single WL run on ref machine: ~0:40
- Wdir size: 15 MB

Full CPU socket VMs under test
HEP Score running mode

Several similarities with the HS06 running mode

- Run HEP Workloads in sequence
  - 3 times per WL, a **container** per WL run, then retain **median** WL score
  - Total running time 3x 3h (with the current workload configuration)

- The available CPU cores are saturated spawning a number of parallel WL copies
  - The **score** of each WL is the **cumulative** event throughput of the running copies
  - When possible the initialization and finalization phases are excluded
    - Otherwise a long enough sequence of events is used

- A WL **speed factor** is computed as ratio of the WL score on the machine under test w.r.t. the WL score obtained on a fixed reference machine
  - CPU Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz (32 cores, SMT ON)

- **HEPscore** is the **geometric mean** of the WLs’ **speed factors**
  - A configurable **weighted geometric mean** would allow to differently weight some workloads or experiments
To summarise

- HEPscore is the tool to
  - Run HEP reference and reproducible WLs on compute resources
  - Compute the single number representing the “average” performance of the processor. This number is used for procurement/pledging/accounting
    - The WLs have to be representative of the experiment job mix
  - The current WLs are mainly probing the performance of CPU/GPU/…
    - If I/O is the limiting factor, another mix of application should be used and define HEPscore-IO
And Analysis?

- Analysis currently represents a tiny fraction of accounted resources.
- But this could change in the future.
  - Worth having it available in the HEP benchmark suite.

Software workloads: who are the big players in WLCG?...

- MC Full Simulation (Geant) Approx. ~25% to ~90% in the four experiments
- MC and Data Reconstruction Approx. ~30% for ATLAS and CMS
- MC Event Generation Approx. ~10% for ATLAS, less for others

In addition: Derivation, Fast Simulation, Analysis…

… and what is their status on GPUs?
Desiderata about the choice of analysis WL

- Be representative of what the experiments run in their analysis jobs
  - Possibly having more than one experiment represented

- Be reproducible: fixed input dataset, fixed configuration, etc

- Be standalone: everything pre-installed in the container

- Possibly target not only CPUs but also accelerators
Discussion…

- Final remark
  - WLCG (MB / GDB / WG) will define the policies for the successor of HS06
  - The Benchmarking WG is providing a versatile tool: HEPscore