HEP Benchmarks for CPUs and beyond

High-Energy Physics workloads as benchmarks of computing architectures

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

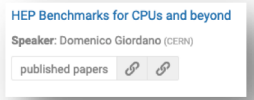
on behalf of the HEPiX CPU Benchmarking WG
hepix-cpu-benchmark@hepix.org

CERN HSF WLCG Virtual Workshop
on New Architectures, Portability, and Sustainability

11-13 May 2020

HEPiX CPU Benchmarking Working Group

- The working group was formed in **2007**
  - **Mandate**: *In charge of defining and maintaining a consistent and reproducible CPU benchmark to describe WLCG experiment requirements*
- WG activity has been re-launched in 2016 to work on
  - next generation HEPiX CPU benchmark, a.k.a. the ”HS06 successor”
  - new running environments: clouds (on-premise and/or commercial clouds) and now HPC
  - new computing architectures
- Plans and achievements shared within the WLCG community (HEPiX Workshops, GDB meetings)
  - Publications linked in the today agenda

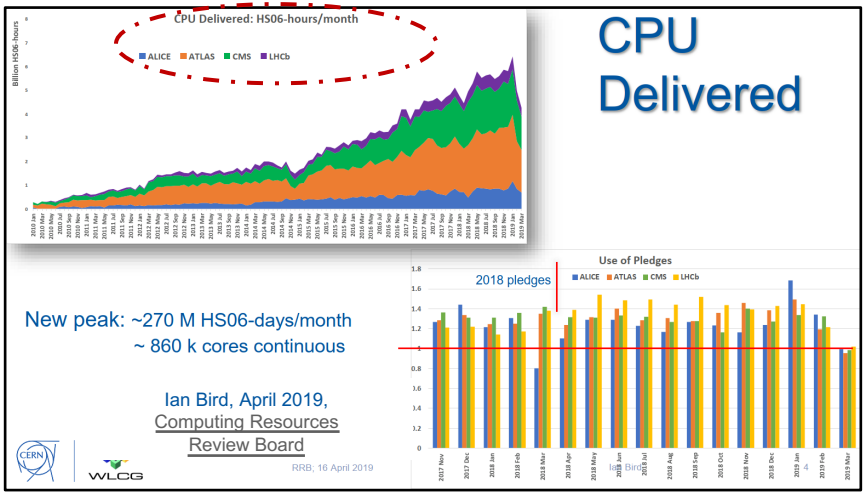


Benchmarking computing resources in WLCG

- Provide Experiments, Funding Agencies, Computing Sites with a ***'score'*** metric **summarizing** the performance of a given CPU in accomplishing some HEP computing work

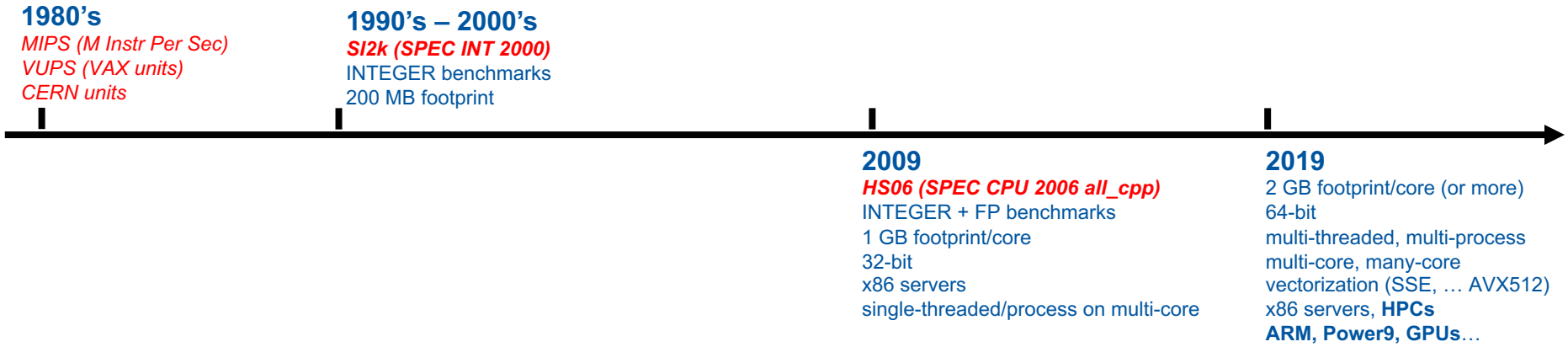
- This 'score' is used to **procure**, **pledge** and **account** the WLCG compute resources

- Since 2009, the WLCG benchmark 'score' is **HEP-SPEC06** (**HS06**)
  - Based on the industrial standard SPEC CPU2006
    - Consortium including AMD, ARM, Dell, Fujitsu, HPE, IBM, Inspur, Intel, Nvidia and Oracle, …



- Soon Central Processing Units (x86 in particular) won't be the only computing resources to take into account to describe WLCG experiment requirements
  - *Most of the considerations you will read about CPUs are valid for heterogeneous resources*

HEP CPU benchmarks evolution



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

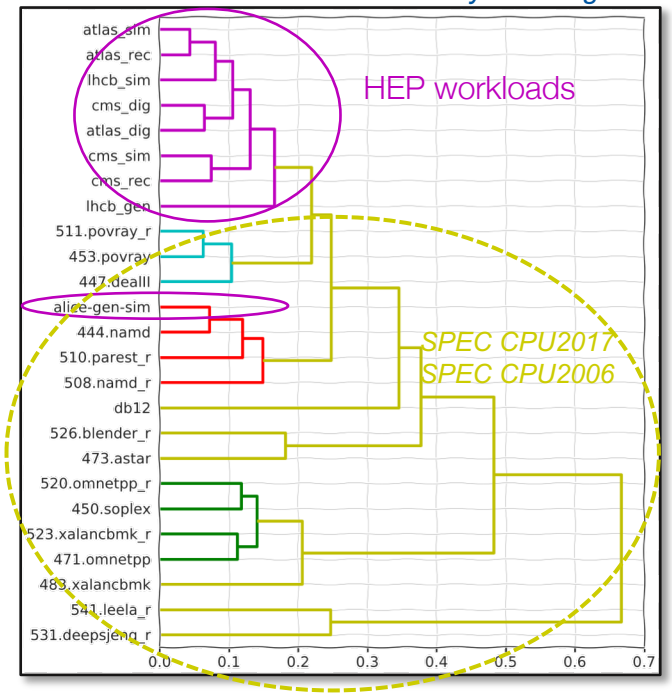
Which benchmark shall WLCG adopt after HS06?

- SPEC CPU 2017, the industrial standard successor of SPEC CPU 2006, has not benefits for WLCG
  - Larger suite, more complex code, shaped for multi-core and multi-threads,
  - Same application fields as SPEC CPU2006
  - Studies on **hardware performance counters** (front-end bound, back-end bound, bad-speculation, retiring) show that

    *HEP workloads have same characteristics and **differ** more with respect to HS06 and SPEC CPU 2017 workloads (see backup slides for details)*

- Opportunity to look elsewhere …

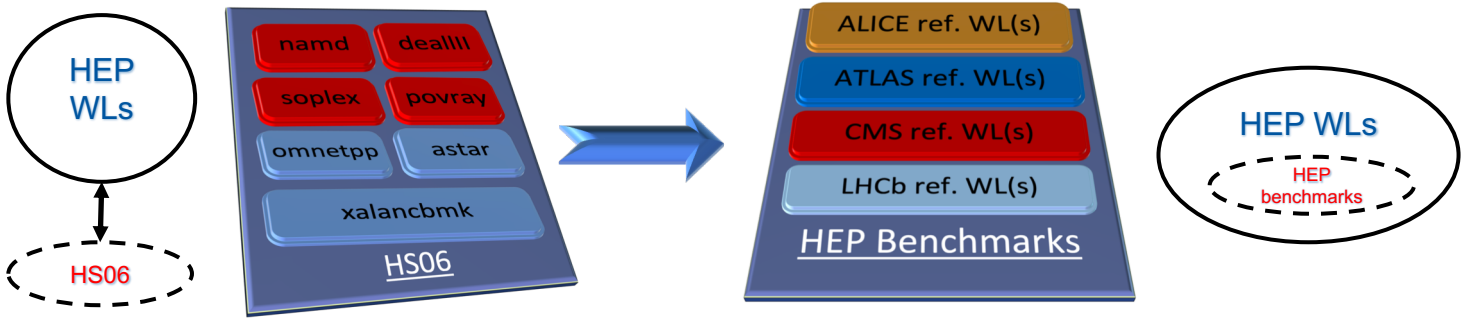
Workloads' similarity dendrogram



Benchmarking CPUs using HEP workloads

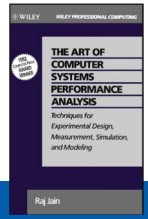
By construction, using HEP workloads directly is guaranteed to give

- 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."

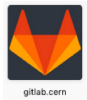
by **Raj Jain** , Wiley Computer Publishing, John Wiley & Sons, Inc, 1992



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)

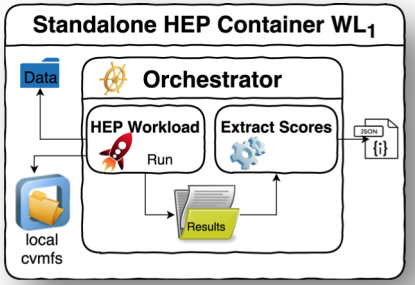
- **Traceability** of the build process
  - Experiment sw, data, configuration
  - Images are **built**, **tested** and **distributed** via gitlab

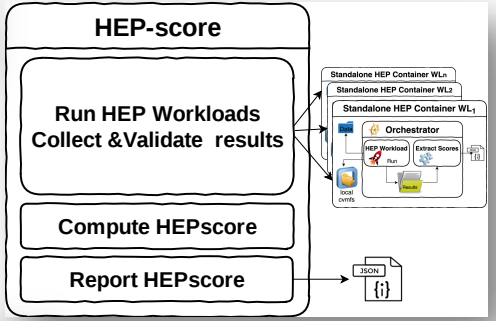


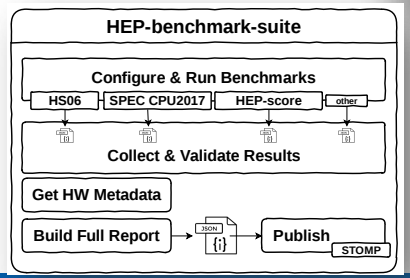
HEP Benchmarks project

Three components https://gitlab.cern.ch/hep-benchmarks

- *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

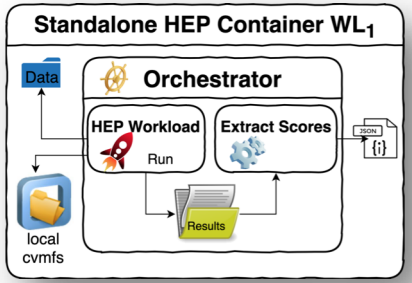




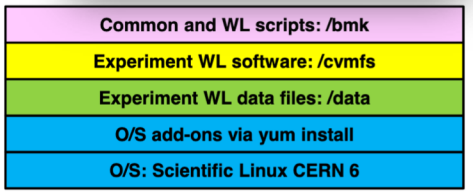


HEP Workloads

- **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)



Container images are made up of layers

- 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

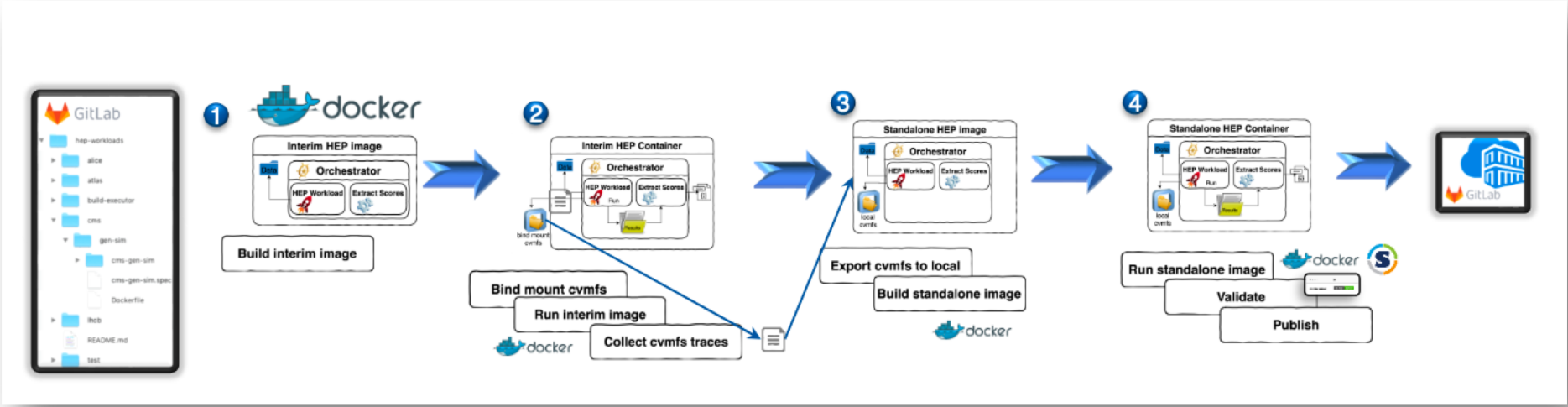
Workload	ATLAS gen	ATLAS sim	ATLAS digi-reco	CMS gen-sim	CMS digi	CMS reco	LHCb gen-sim
Robustness	✅	✅	✅	✅	✅	✅	✅
Reproducibility	0.8%	2%	0.6%	1.5%	1%	1%	1%
Memory	✅	✅	✅	✅	✅	✅	✅
Image size (unpacked)	1.5 GB	2.0 GB	6GB	10 GB	6.5 GB	5.5 GB	2.6 GB
Readiness	✅	✅	✅	✅	✅	✅	✅

✅ okay
❌ blocker

Reproducibility, evaluated as spread in repeated measurements $(score_{max} - score_{min})/score_{mean}$

Build pipeline

- 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

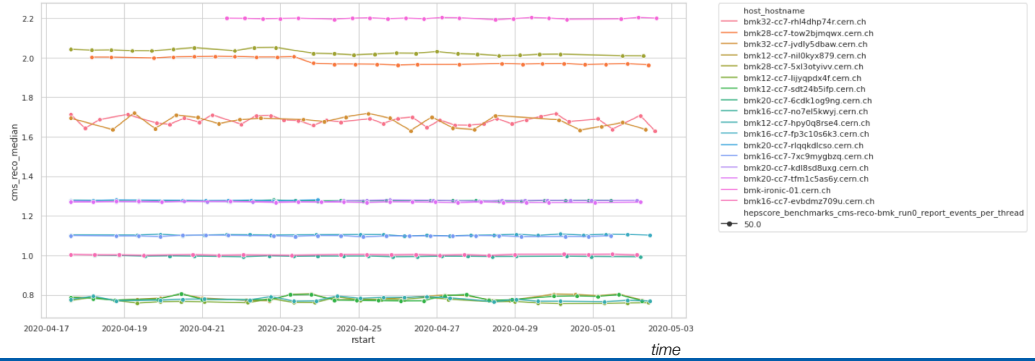
Current default configuration (still under study)

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

LHCb gen-sim score [evts/sec] in multiple runs

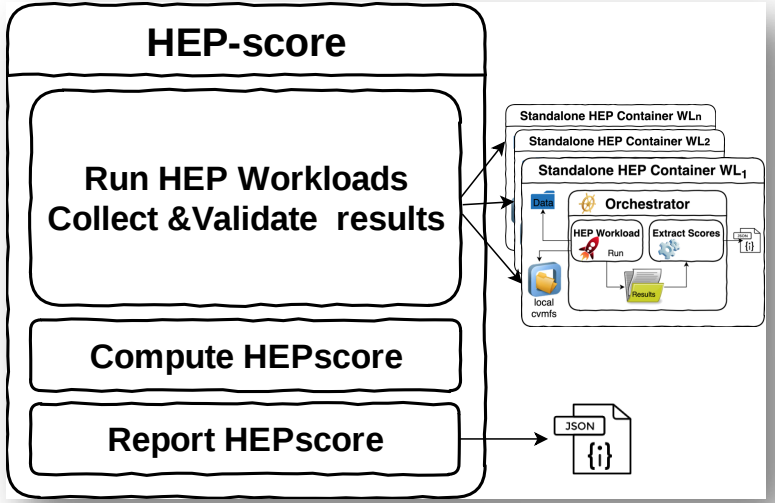
Full CPU socket VMs under test

CMS reco score [evts/sec] in multiple runs



HEP Score: combine the HEP Workloads' scores

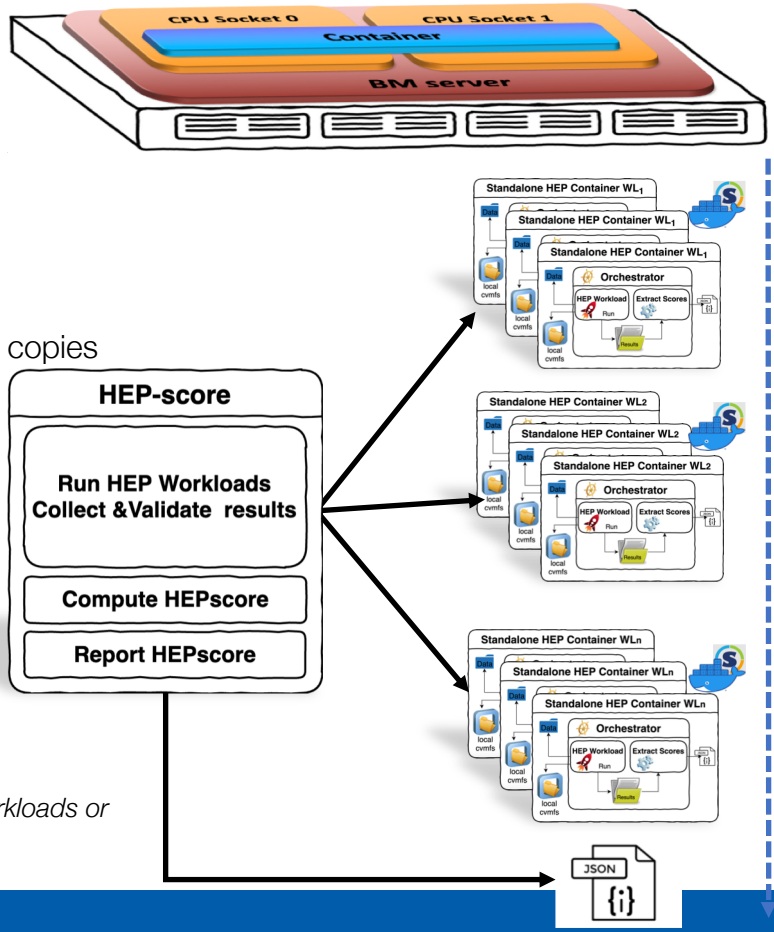
- Orchestrate the run of a series of HEP Workloads
- Compute & Report the HEPscore value
  - Default config. defines the HEPscore value
  - Other config. to perform specific studies
- HEP score does not include HEP Workloads' sw
  - HEP Workloads' sw is "*isolated*" in dedicated containers
  - Enable the utilization of **additional WLs**, as long as they comply with the expected API
  - Can be extended to other workloads, running on GPUs for instance



HEP Score running mode

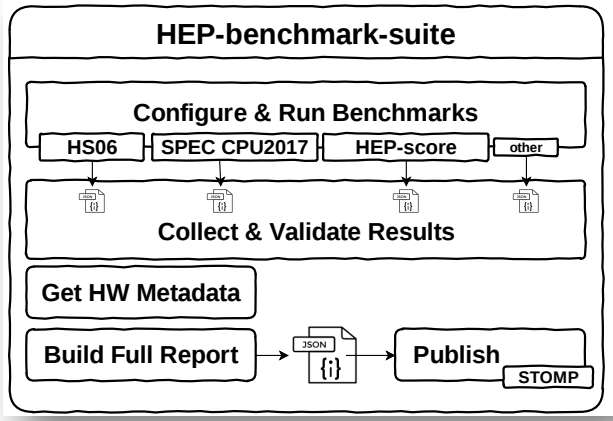
Several similarities with the HS06 running mode

- 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*



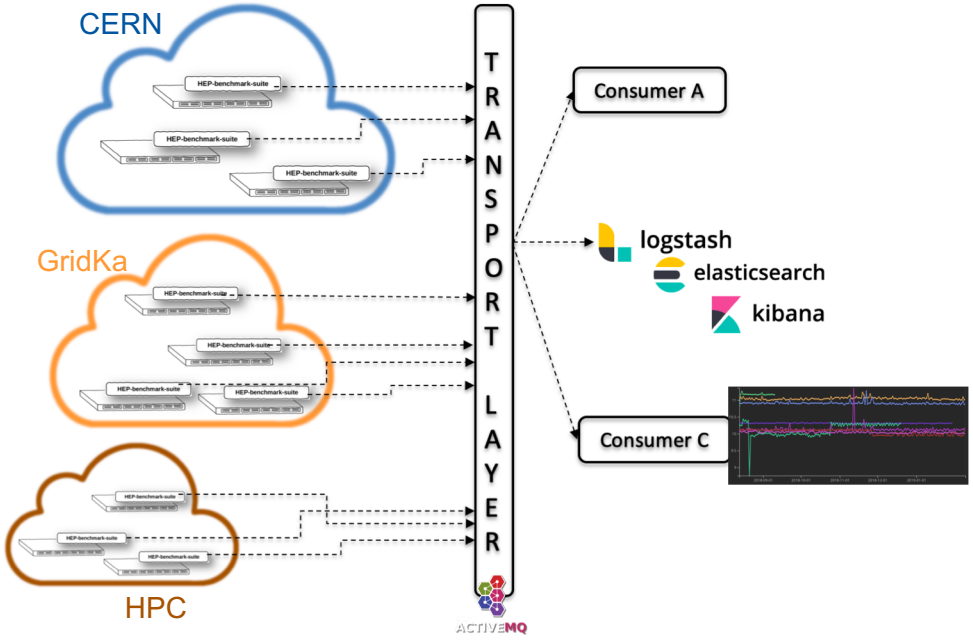
HEP Benchmark Suite

- Move from a 10-years-old procedural approach (designed for HS06) to a **toolkit**
  - A sort of “Push the button” & Run & Get Results
- Benefits:
  - Control the execution of several benchmarks
    - HEPscore, SPEC CPU2017, HS06, KV, DB12, …
    - NB: does NOT distribute sw under proprietary license (such as HS06), just requires that this code is pre-installed on the machine
  - Simplify the sharing, tracking and comparison of results
    - Metadata track data-centre name, CPU model, host, kernel version, …
- Distributed as an **docker** image
  - Several examples available in this gitlab repository



Centralise the benchmark data storage

- Global results published to a messaging system in json format
  - Ideal for monitoring and offline analysis
- Adoption
  - @ CERN to continuously benchmark available CPU models
  - Used in CERN commercial cloud procurements (HNSciCloud)
  - Tested by other site managers (GridKa, RAL, INFN-Padova, CC-IN2P3, …)
  - The HEP Benchmark Suite will drive the CPU benchmark of all nodes in the CERN data centre



Benchmarks for heterogeneous resources

- As Benchmarking WG, we consider essential to start the investigation of **HEP Benchmarks** for the **CPU+GPU** system
  - How to value pledged HPC resources? How to procure CPU+GPU systems?

- First demonstrators of **standalone container** for **GPU** benchmarking under preparation
  - Patatrack
    - Based on CMS (Pixel, Calo) reconstruction with GPUs
  - cern.ch/SixTrack
    - Computes trajectories of charge particles in synchrotrons

- Other **production** applications running on GPU are welcome

Challenges: HEP Benchmark Suite on HPC

Assumptions valid for WLCG sites (and vendor servers) are not valid anymore in HPC centres

Install and run the benchmark by site admins with root privileges

Run the benchmark suite and the HEP WLs with privileged **docker-in-docker**

👍 We could rely on **singularity** and **singularity-in-singularity**, provided that HPC sites install **Singularity 3.5.3+** and **enable user namespaces**

- Singularity 3.5.3+ fixes a few bugs preventing these releases from being able to be used (either with setuid or user namespaces) with the workloads

⚠️ Unfortunately this configuration is not available in most of the HPC centres

- Will the situation change after the OSG & EGI security recommendations?
- Meanwhile we are restructuring the code to avoid singularity-in-singularity

Enable user namespaces via `sysctl`:

```
root@host # echo "user.max_user_namespaces = 15000" \
    > /etc/sysctl.d/90-max_user_namespaces.conf
root@host # sysctl -p /etc/sysctl.d/90-max_user_namespaces.conf
```

From Dave Dykstra <dwd@fnal.gov>
Subject **Baseline recommendation for Singularity changed**
To wlcg-operations@cern.ch <wlcg-operations@cern.ch>
Reply | Reply All | Forward | Archive | Junk | Delete | More
08.05.20, 21:00

Earlier this week OSG & EGI security sent out a recommendation for WLCG sites to enable unprivileged user namespaces on RHEL 7 worker nodes. If that is done, WLCG VOs will run singularity out of cvmfs instead of from the local host, so sites are then encouraged to remove the singularity rpm (or singularity-runtime rpm if still using singularity-2.6.1) from worker nodes unless they have other VOs that require it.

The WLCG baseline table at
https://twiki.cern.ch/twiki/bin/view/LCG/WLCGBaselineTable
has now been changed to say the Singularity version is unprivileged user namespaces or 2.6.1. The 2.6.1 version is very old but as far as I know still works and has no known security holes.

The instructions for enabling unprivileged user namespaces are at
https://opensciencegrid.org/docs/worker-node/install-singularity/#enabling-unprivileged-singularity

Dave

Ongoing work

- Validation studies
  - Run at large scale on multiple production nodes, different CPU models and data centres
  - Report foreseen for upcoming GDB meeting
- Consolidation of the code base
- Focus on development and testing on heterogeneous resources in the next months
  - Strengthen by an additional FTE project associate in WLCG/openlab
  - Test in HPC centres
  - Inclusion of new CPU/GPU workloads
- Would you like to contribute? Contact: hepix-cpu-benchmark@hepix.org

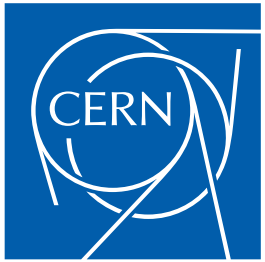
Conclusions

- We are building a **domain specific HEP benchmark** directly from HEP workloads, using the throughput [event/s] as key metric
  - The technology aspects have been addresses and solved to a large extent
  - Ready to include new HEP software when available

- The full **HEP Benchmarks** chain is in place
  - Examples are available at https://gitlab.cern.ch/hep-benchmarks/hep-benchmark-suite

- The Suite supports also HS06: facilitate a smooth transition to a different benchmark

- Approach and implementation are valid also for non-x86 resources
  - Adopting the same Experiments workloads that exploit the heterogeneous resources


CERN

Current WLCG benchmark: *HEP-SPEC06 (HS06)*

- Based on SPEC CPU2006
  - Standard Performance Evaluation Corporation was founded in 1988
  - SPEC CPU2006: **Industry-standard**, CPU-intensive, benchmark suite
  - Current SPEC CPU subcommittee members include AMD, ARM, Dell, Fujitsu, HPE, IBM, Inspur, Intel, Nvidia and Oracle [*]

[*] https://www.spec.org/cpu2017/press/release.html

- HS06 is a subset of SPEC CPU® 2006 benchmark, tuned for HEP
  - 7 C++ benchmarks recompiled with gcc optimizer switches of LHC experiments' software
  - In **2009,** proven **high correlation** with HEP workloads

Bmk	Int vs Float	Description
444.namd	CF	92224 atom simulation of apolipoprotein A-I
447.dealII	CF	Numerical Solution of Partial Differential Equations using the Adaptive Finite Element Method
450.soplex	CF	Solves a linear program using the Simplex algorithm
453.povray	CF	A ray-tracer. Ray-tracing is a rendering technique that calculates an image of a scene by simulating the way rays of light travel in the real world
471.omnetpp	CINT	Discrete event simulation of a large Ethernet network.
473.astar	CINT	Derived from a portable 2D path-finding library that is used in game's AI
483.xalancbmk	CINT	XSLT processor for transforming XML documents into HTML, text, or other XML document types

The 7 C++ HS06 benchmarks

Quantitative comparison with WLCG workloads

- Unveil the **dissimilarities** between HEP workloads and the SPEC CPU benchmarks
  - Using the **Trident** toolkit
    - analysis of the hardware **performance counters**

Characterization of the resources utilised by a given workload

Percentage of time spent in

- **Front-End** – fetch and decode program code
- **Back-End** – monitor and execution of uOP
- **Retiring** – Completion of the uOP
- **Bad speculation** – uOPs that are cancelled before retirement due to branch misprediction