HEP-Benchmarks

High Energy Physics workloads as benchmarks of computing architectures

D. Giordano (CERN)

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

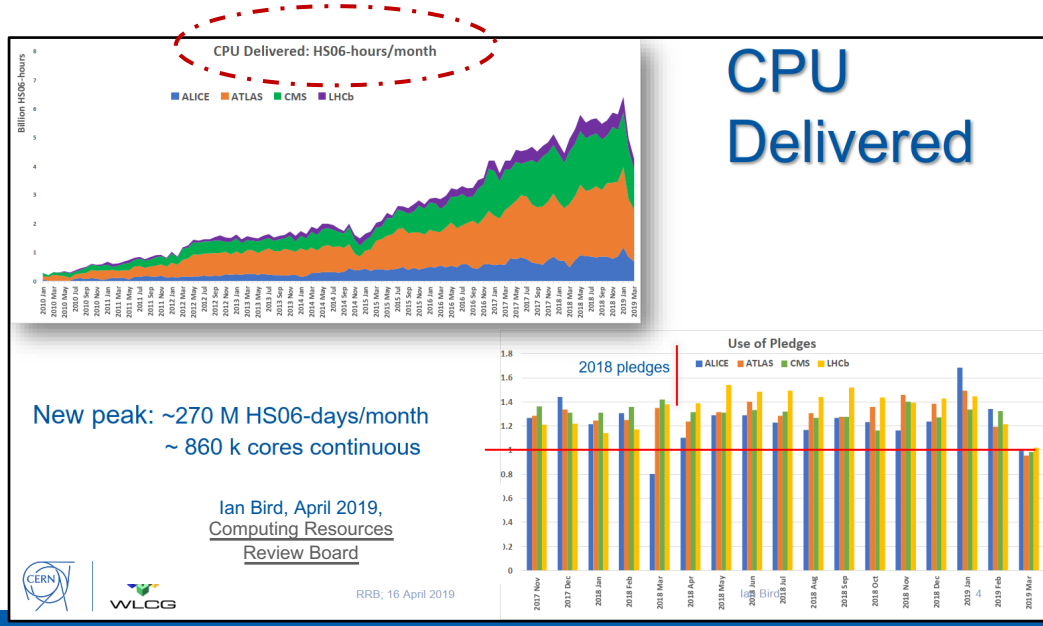
CERN Openlab Technical Workshop
22-23 Jan 2020

[*] In charge of defining and maintaining a consistent and reproducible CPU benchmark to describe WLCG experiment requirements

Benchmarking CPU 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

- Used to **procure**, **pledge** and **account** the WLCG compute resources
- Since 2009, the WLCG benchmark 'score' is **HEP-SPEC06** (**HS06**)



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 [*]

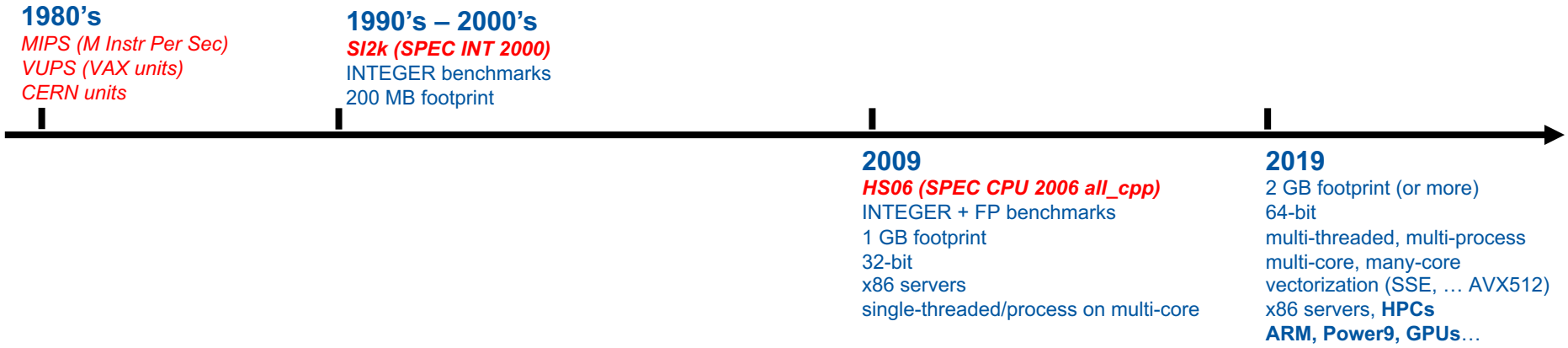
[*] https://www.spec.org/cpu2017/press/release.html

- 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

Follow the HEP sw evolution

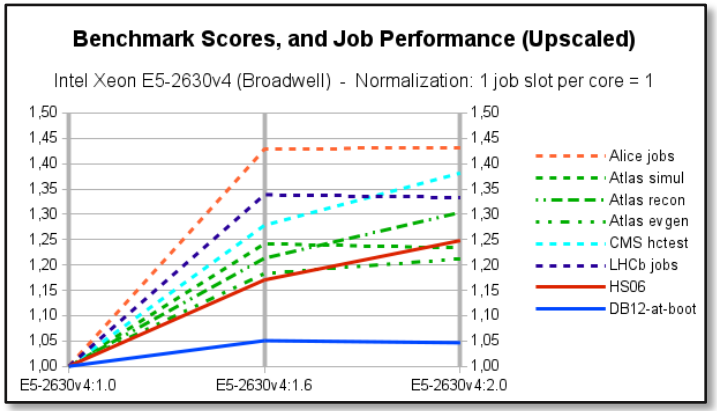


HEP software (and computing) evolves… so shall do HEP CPU benchmarks!

- Challenges:
  - Probe the compute resources as the HEP applications do
  - Include **heterogeneous resources**
  - Summarize performance using a **single number**, at least for accounting purposes

Identify the right performance metric

- *Is HS06 still representative of the WLCG workloads?*
  - Found that it is still good but **not perfectly** correlated with the performance of today HEP applications



- *Which benchmark shall WLCG adopt after HS06?*

> - By end of day on **January 9, 2018 US Eastern Time**, SPEC will retire SPEC CPU2006.
>   - After this day, further submissions not already under review will not be published by SPEC and technical support for SPEC CPU2006 will end.
>   - For publication on SPEC's website by January 9th, 2018, results need to be submitted to SPEC by the **December 26, 2017 3AM US Eastern Time** submission deadline. Note that, per above, corresponding SPEC CPU2017 results are also needed.

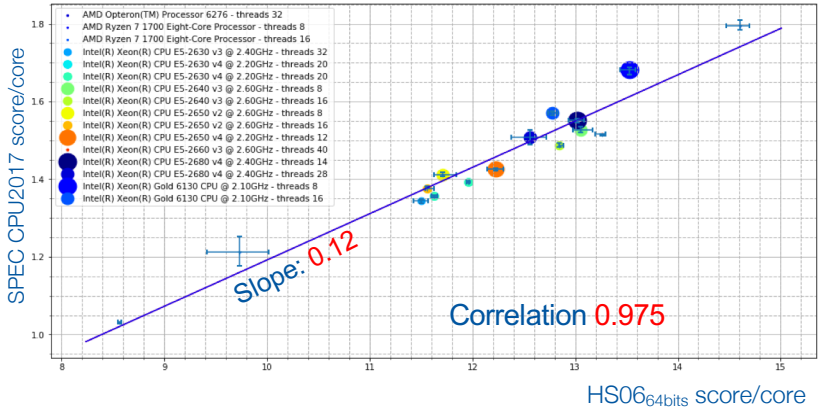
https://www.spec.org/cpu2006/

  - Is SPEC CPU2017 the right candidate?
    - Larger suite, more complex code, shaped for multi-core and multi-threads

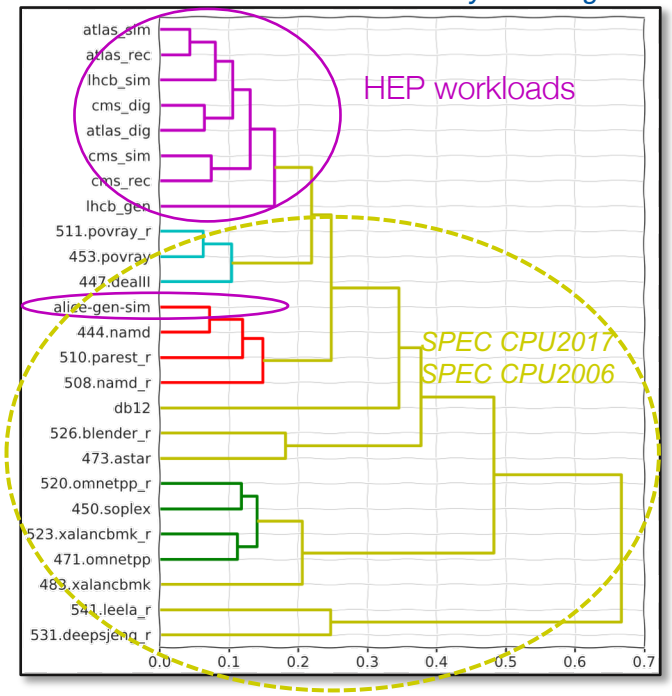
Is SPEC CPU2017 the right candidate?

- SPEC CPU 2017 Vs HS06
  - *Measured extremely high correlation between the two C++ benchmark suites*

  => No advantage for WLCG in moving now to SPEC CPU 2017



Workloads' similarity dendrogram

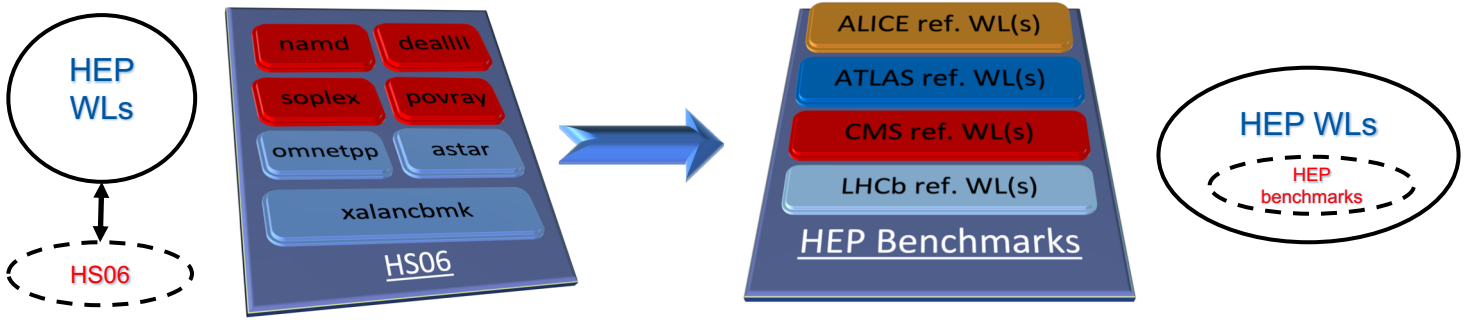


- Studies on hardware **performance counters**
  - *HEP workloads have same characteristics and **differ** more respect to HS06 and SPEC CPU 2017 workloads* (see backup slides for details)

Benchmarking CPUs using HEP workloads

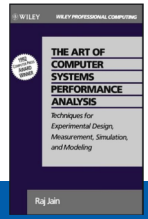
By construction, using HEP workloads directly is guaranteed to give

- 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



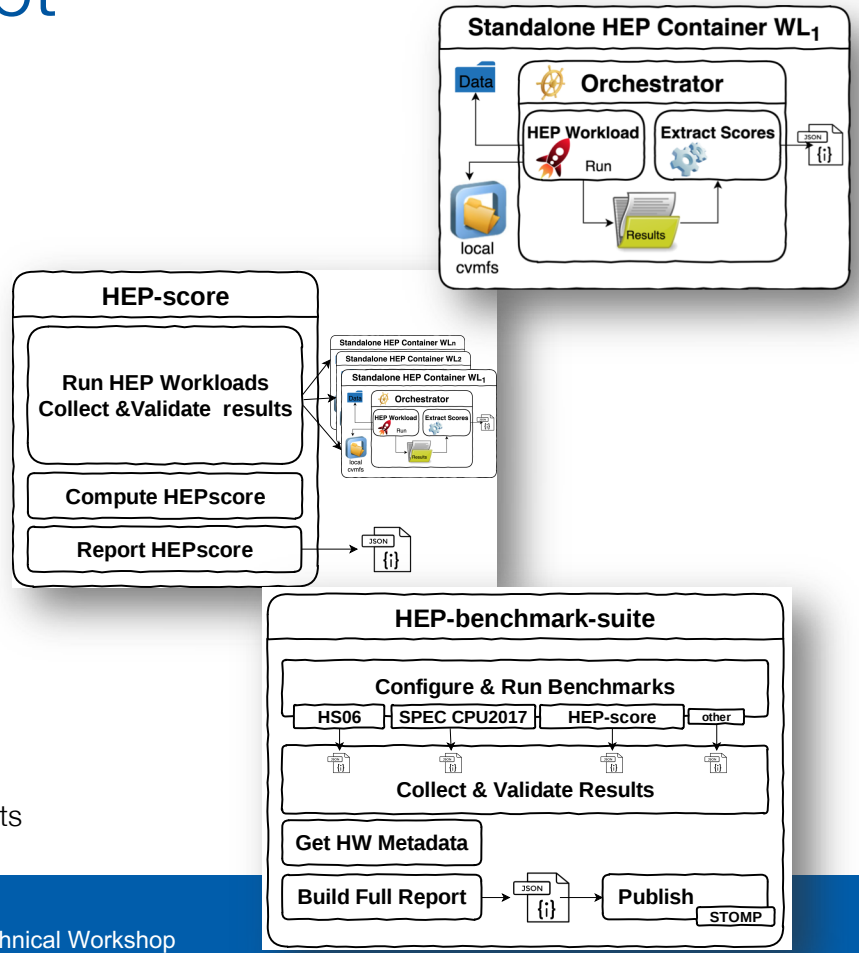
Criteria to build the HEP Benchmark

- **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**)
- **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*
  - 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



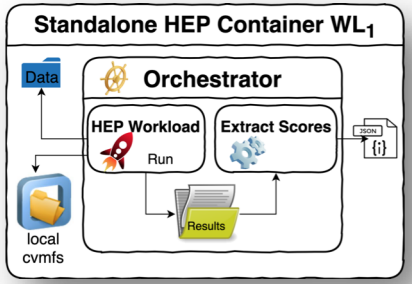
HEP Workloads

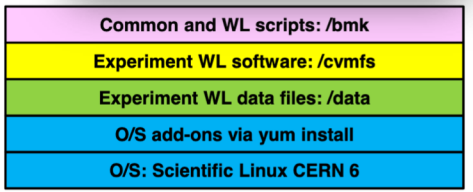
- **Standalone containers** encapsulating **all and only** the dependencies needed to run each workload as a benchmark

- Components of each HEP Workload
  - SW repository & 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
  - Run each workload via a single command line:

```
>docker run $IMAGE_PATH
```





Container images are made up of layers

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 $(score_{max} - score_{min})/score_{mean}$

Benchmarks for heterogeneous resources

- In the future WLCG resources will likely include HPCs with GPUs
  - Essential to investigate a **HEP Benchmark** 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 (see previous reports from F. Pantaleo & V. Khristenko)
    - Based on CMS reconstruction with GPUs (Pixel track reconstruction, Calorimeter reconstruction)
  - cern.ch/SixTrack (R. De Maria)
    - Computes trajectories of charge particles in synchrotrons

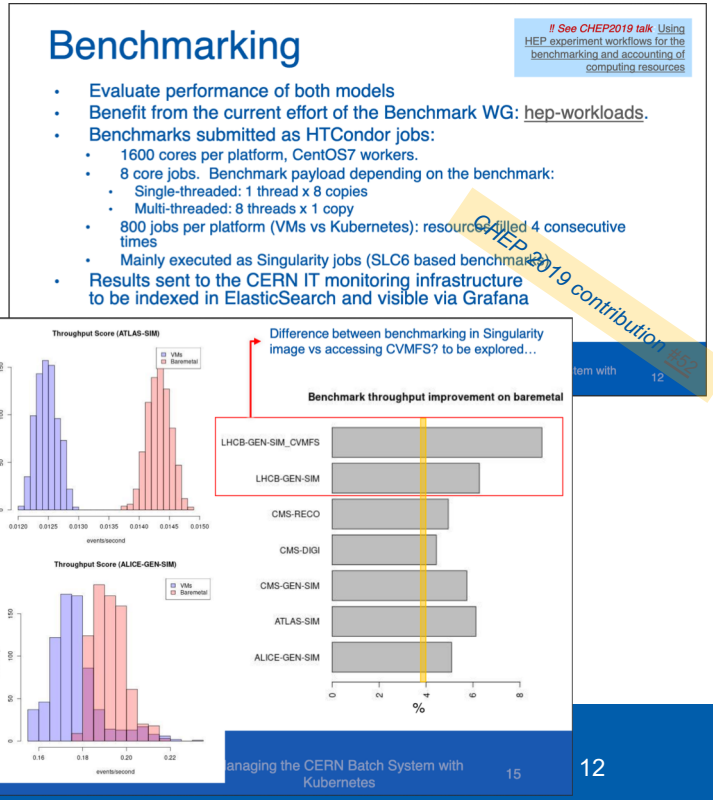
- Other **production** applications running on GPU are welcome

Early adopters

- The HEP Benchmark Suite will drive the CPU benchmark of all node in the CERN data centre
  - Embedded into **Openstack Ironic** enrollment

- HEP Workloads are being used to test/improve the **CERN Batch** infrastructure
  - First large scale tests
  - Leverage the HEP Workloads reproducibility

- Successful examples of runs in WLCG sites and HPC centres



Ongoing work

- ❑ Validation studies
- ❑ Inclusion of new CPU/GPU workloads
- ❑ Consolidation of the code base
- ❑ Run at large scale on production nodes

All those areas would greatly profit from new contributors

– Contact: hepix-cpu-benchmark@hepix.org

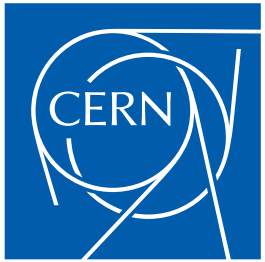
Conclusions

- HEP software is evolving and so shall do HEP CPU benchmarks
  - After 10 years, HS06 no longer describes well enough HEP workloads

- Our solution: build a **new benchmark** directly from HEP workload throughputs
  - Enabling technologies: Docker/Singularity containers and cvmfs tracing mechanism
  - **Individual containers released for all workloads provided by the LHC experiments**
    - See https://gitlab.cern.ch/hep-benchmarks/hep-workloads

- The full **HEP Benchmarks** chain is in place, and under test

- Outlook: can extend the idea and implementation to HPCs and non-x86 resources


CERN

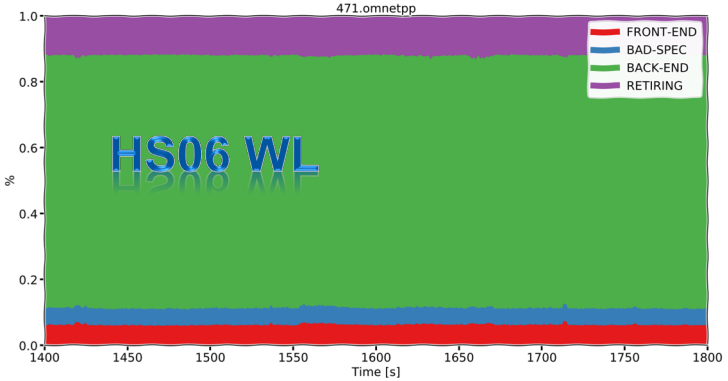
Quantitative comparison with WLCG workloads

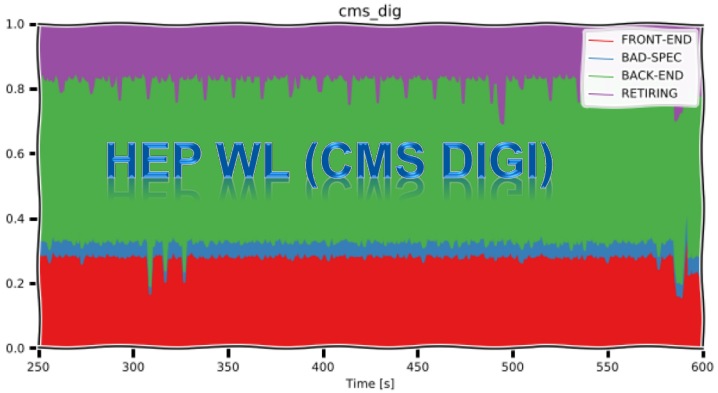
- Unveil the **dissimilarities** between HEP workloads and the SPEC CPU benchmarks
  - Using the **Trident** toolkit
    - analysis of the hardware **performance counters**

Characterization of the resources utilised by a given workload

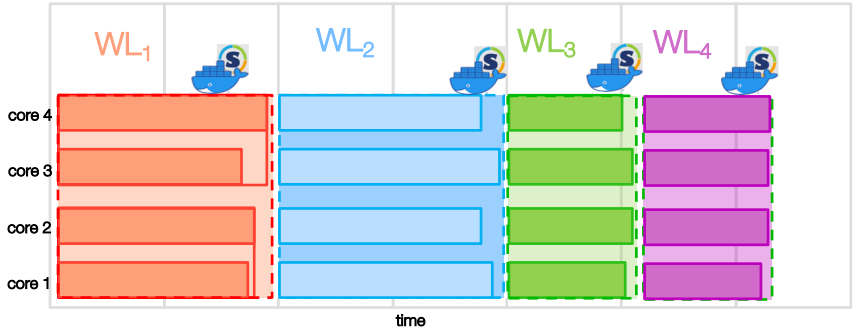
Percentage of time spent in

- **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





HEP Score running mode



- HEP-score triggers HEP Workloads' runs in sequence
  - A **container** per WL
  - **3** times per WL, in sequence, and the **median** WL score is retained
- Each container runs the Experiment executable with a configurable number of threads (MT) or processes (MP)
- The available cores are saturated spawning a ***computed*** number of parallel copies
- The **score** of each WL is the cumulative event throughput of the running copies
  - When possible the initialization and finalization phases are excluded from the computation
  - Otherwise a long enough sequence of events is used
- A WL ***speed factor*** is computed as ratio of the WL score on a given machine w.r.t. the WL score obtained on a fixed reference machine
- **HEPscore** is the **geometric mean** of the WLs' ***speed factor***