Benchmarking Working Group Status Report

D. Giordano (CERN/IT)

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

HEPiX Autumn 2020 Online workshop
13 October 2020

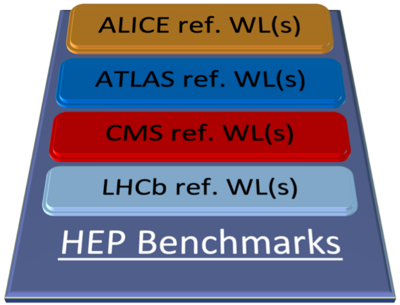
Intro

- This report focuses on the HEP Benchmarks project https://gitlab.cern.ch/hep-benchmarks
  - Main activity of the WG in the last year
  - With several new / young contributors

Scenarios	HS06	HEPscore
x86 CPUs (y. 2010-2020)	✓	✓
New CPUs models and/or arch	?	✓
New Exp Sw	?	✓ (w/ new reference WLs)
CPU + GPU/FPGA/…	✗	✓ (same speed definition: event/s)

- In short
  WLCG has to change the benchmark HS06 sooner or later
  - Motivations extensively presented at the last HEPiX Workshop '19
  - Briefly: HS06 end of technical support (2017), targets only CPUs, we don't know if will continue to scale well w.r.t. new HEP sw

- **Field-specific** (HEP) workloads guarantee by construction
  - A score with **high correlation** to the throughput of HEP workloads
  - A usage pattern that is similar to that of HEP workloads

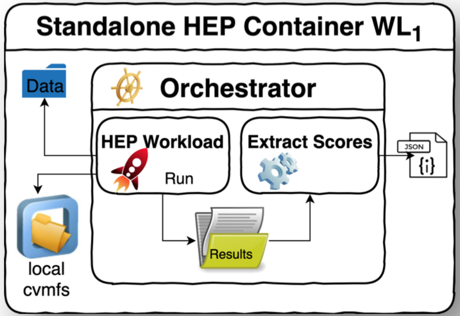


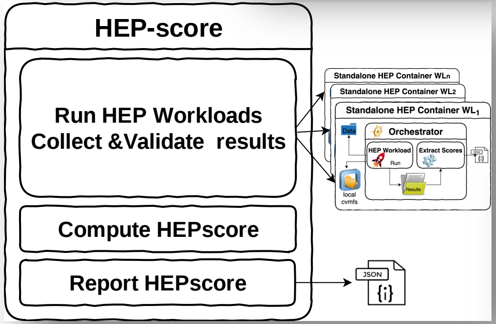
HEP Benchmarks project

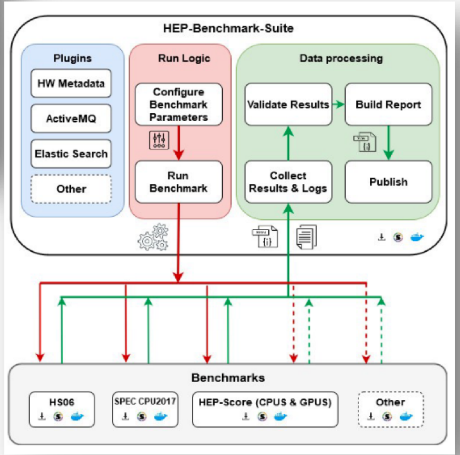
Three main components being developed since ~2 years

!!! Released under GPLv3 licence !!!

- *HEP Workloads*
  - Individual **reference** HEP workloads
  - Common build infrastructure
- *HEP Score*
  - Orchestrate the run of a series of HEP workloads
  - Compute the **HEPscore** value
  - Report whole set of WL results
- *HEP Benchmark Suite*
  - **Meta-orchestrator** of multiple benchmark suites
    - HEPscore, HS06, SPEC CPU2017…

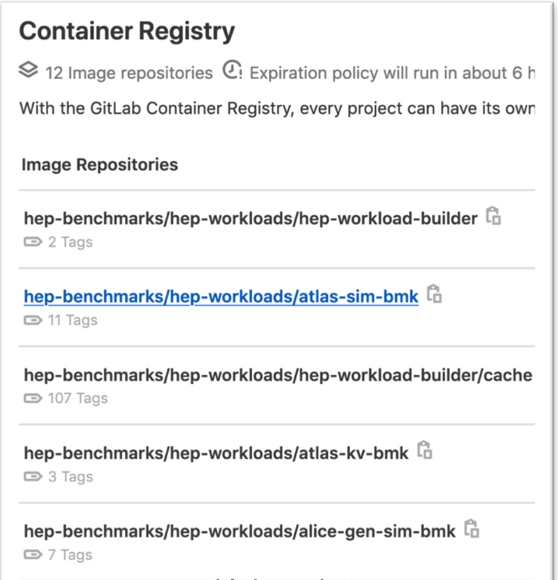






HEP Workloads

HEP Workloads



gitlab registry

- **Standalone containers** encapsulating <u>all and only</u> the dependencies needed to run each workload as a benchmark
  - Reduce library size using cvmfs tracing/shrinking technology
  - Report results in a structured json format (see next slides)

- Standalone containers available in gitlab registry and automatically **distributed via CVMFS** (*!!NEW!!*)

  /cvmfs/**unpacked.cern.ch**/gitlab-registry.cern.ch/hep-benchmarks/hep-workloads

  - Run a given workload via a single command line:

    >singularity run <IMAGE_PATH> <args>

    >docker run <IMAGE_PATH> <args>

Extensive validation process of WL

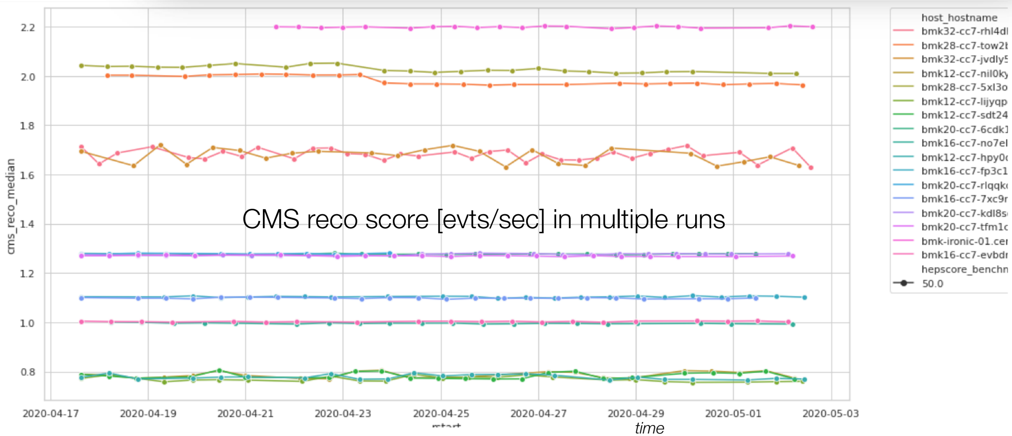
- Validating reproducibility, robustness, run duration, disk space
- Continuously running in a number of virtual & physical machines
- Evaluated a different number of events per WL to shorten the runtime

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)	1.65 GB	6.0 GB	6GB	5.4 GB	11 GB	8.4 GB	2.6 GB
Readiness	✅	✅	✅	✅	✅	✅	✅

✅ okay
❎ blocker

CPU Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz (32 cores, SMT ON)

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**	



Growing list of workloads

- Alice gen-sim
  - The current one is based on Geant3, not fully stable
  - New workload in preparation: ALICE O2 multi-core simulation
- Atlas digi-reco
  - waiting for multi-processing version with pile-up overlay

Packaging of standalone containers can be easily applied to other workloads and Experiments

HEP Score

HEP-Score vs HS06

- Similarities
  - Implement the **geometric mean** of the WL **speed factors**
  - 3 runs per WL and get the **median** value

- Differences
  - HEP-Score adopts a set of WLs suggested by the experiments
    - Event throughput [event/s] as key metric
    - The parameters can be tuned to represent the production job mix
      - N.B.: The final list must be defined following WLCG needs
  - HEP-Score: the geometric mean has been extend to the **weighted geometric mean**
    - Workloads can be **differently weighted** to represent different job mix
      - N.B.: To be defined as WLCG policy

$$\bar{x} = \left(\prod_{i=1}^{n} x_i^{w_i} \right)^{1/\sum_{i=1}^{n} w_i}$$

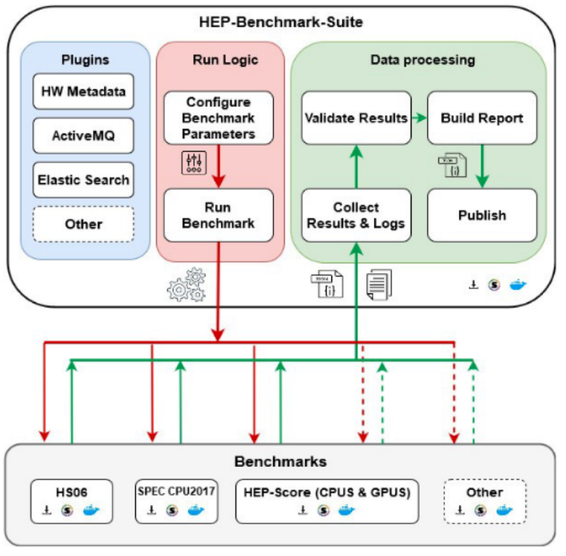
https://en.wikipedia.org/wiki/Weighted_geometric_mean

Status of HEP-Score

- Version v1.0 will be released in the coming weeks
  - C. Hollowell (BNL), C. Van Der Laan (CERN Intern)
- Several new features in v1.0
  - **Singularity** and docker engines are both supported
  - Access of **cvmfs unpacked** images
  - Better handling of disk space, configurable cleanup of the working directory
  - Optimised the report structure
  - Improved CI tests
- To install: `pip install --user git+https://gitlab.cern.ch/hep-benchmarks/hep-score.git@<tag>`
- To run: **Singularity**: `hep-score /SCRATCHDIR` **Docker**: `hep-score -d /SCRATCHDIR`

HEP Benchmark Suite

HEP Benchmark Suite

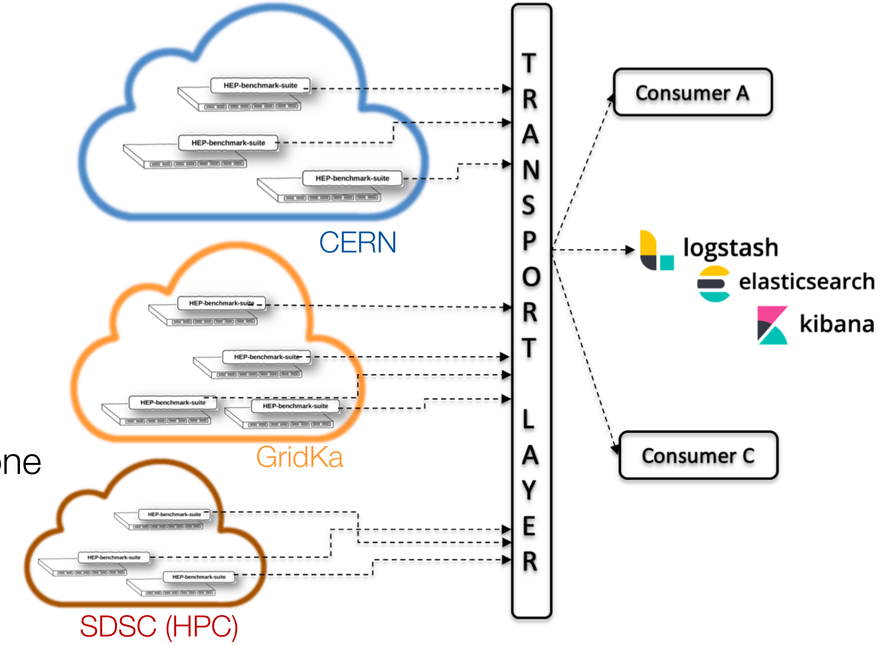


- ❑ Meta-orchestrator for the execution of several benchmarks
  - HS06, HEP-Score, …
- ❑ Version v2.0 will be released in the coming weeks
  - Fully rewritten in python, distributed via pip install
    - M. Fontes Medeiros & D. Southwick (CERN/IT)
  - Very few dependencies needed, install as unprivileged user
  - New metadata section with detailed HW information

Centralise the benchmark data storage

- ❑ The Hep Benchmark Suite is the first component of a data pipeline to collect results
  - Enable sharing, tracking, studies
  - Ideal for monitoring and offline analysis

- ❑ Adoption
  - @ CERN integration into **Openstack Ironic**
    - About to replace the current benchmarking done with an in-house built image
  - Tested by other sites (GridKa, RAL, INFN-Padova, …) and **HPC centre** (SDSC)



Access to HPC resources

Support from HPC sites has been crucial for testing and enhancement of HEP-benchmarks on HPC centers

- Access to large pool of nodes and various hardware configurations
- Ability to scale across clusters and partitions, integrating with job scheduling tools (SLURM)
- +Functional tests on Subatech & Cineca

CCIPL - Subatech

CINECA

https://ccipl.univ-nantes.fr/le-centre-de-calcul-intensif-des-pays-de-la-loire-438632.kjsp
https://fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/DEEP-EST/_node.html
https://www.sdsc.edu/support/user_guides/popeye-simons.html

SDSC SAN DIEGO SUPERCOMPUTER CENTER

CPU Nodes	Skylake 8168 (144x2), 6148 (72x2) Cascade 8268 (216x2)
Cores	~28K + 128 GPUs
DRAM/node	768 GB
GPU	4x NVIDIA V100/node

Sept. '20

AMD EPYC 7742 Compute Nodes	Configuration
Node count	728
Clock speed	2.25 GHz
Cores/node	2x64
DRAM/node	256 GB
NVMe/node	1 TB

- Early Access before general availability
- 13 racks of 56 CPU nodes + 4x4 GPU nodes
- network attach 12PB Lustre + 7PB Ceph

DEEP Extreme Scale Technologies

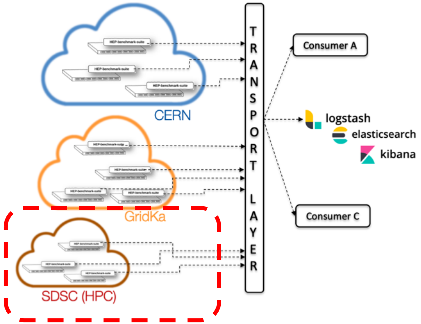
CPU Nodes	Skylake 6146 (50x2) Cascade 4125 (75), 8260M (16x2)
Cores	2568 + 91 GPUs + 16 FPGA
DRAM/node	192GB / 48GB / 348GB
GPU	1x NVIDIA V100/node

Benchmarking nodes in a remote site

```
[[redacted] ~]$ sbatch run_suite.sbatch
Submitted batch job 190918
```

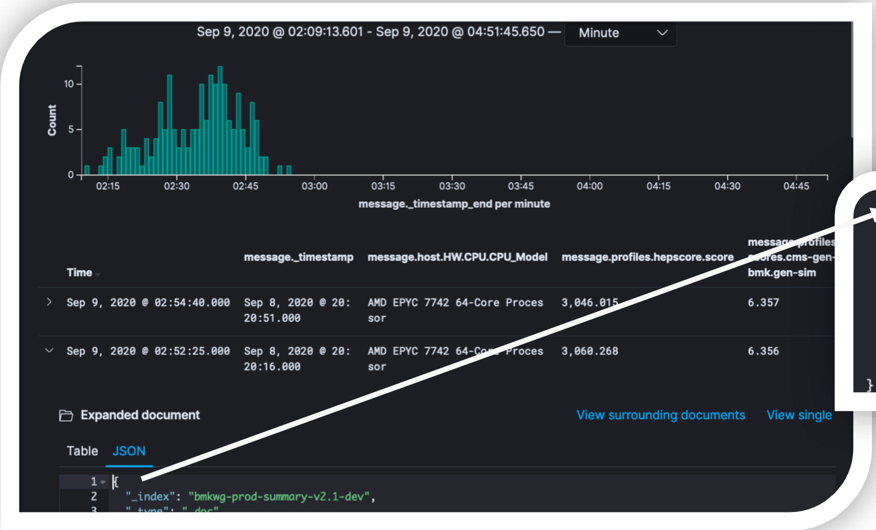
D. Southwick (CERN/IT)

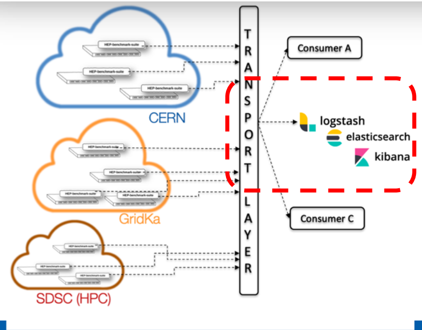
- ❑ Nodes in **HPC centre (SDSC)**
  - More constraints than WLCG sites
- ❑ Run with SLURM on full nodes
- ❑ Script defines
  - SW requirements
  - Benchmark configuration file (with secrets for the publication to the remote transport layer)




```
[redacted] ~]$ cat run_suite.sbatch
!/bin/bash
#
#SBATCH --exclusive --hint=multithread
#SBATCH --job-name=HEP-Benchmark-suite
#SBATCH --output=res%A-%j.out
#SBATCH --mail-type=END,FAIL
#SBATCH --mail-user=
#SBATCH --array=1-200

module purge
module load gcc singularity/3.5.3 python3/3.7.3

export SUITE_BRANCH=qa-v2.0
export RUNDIR=/tmp/HEP-benchmark-suite
export CONFPATH=$HOME/hpctest_full.yaml

echo "Running HEP Benchmark Suite on $SLURM_CPUS_ON_NODE Cores"
mkdir -p $RUNDIR
python3 -m pip install --user --upgrade git+https://gitlab.cern.ch/hep-benchmarks/hep-benchmark-suite.git@$SUI

# run
$HOME/.local/bin/bmkrun --config $CONFPATH --uid $SLURM_JOB_ID --rundir $RUNDIR -v
```

Benchmark results on the central DB @CERN





- HEP Benchmark Suite report
  - Metadata:
    - Host HW, SW configuration
    - User tags
  - Benchmark profiles
    - HEPscore in the example

Benchmark results on the central DB @CERN


```
"@timestamp": "2020-09-09T00:52:32.412Z",
"message": {
  "_timestamp_end": "2020-09-09T00:52:25Z",
  "_timestamp": "2020-09-08T18:20:16Z",
  "host": {↔},
  "profiles": {
    "hepscore": {↔}
  },
  "json_version": "v2.1-dev",
  "_id": "678200_2020-09-08T18:20:16Z"
}
},
```



```
"profiles": {
  "hepscore": {
    "status": "success",
    "app_info": {
      "registry": "gitlab-registry.cern.ch/hep-benchmarks/hep-workloads",
      "name": "HEPscore19",
      "reference_machine": "CPU Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz",
      "hepscore_ver": "1.0.0.0rc4.dev54",
      "hash": "0e13b73a5bb21e4ce12b85b54a39c965fecce1099f0f4c54545adacd603ca5ff"
    },
    "environment": {
      "date": "Tue Sep  8 14:20:17 2020",
      "singularity_version": "error",
      "system": "Linux worker5237 3.10.0-1127.10.1.el7.x86_64 #1 SMP W
    },
    "benchmarks": {↔},
    "settings": {
      "replay": false,
      "repetitions": 3,
      "container_exec": "singularity",
      "scaling": 355,
      "method": "geometric_mean"
    },
    "wl-scores": {
      "cms-digi-bmk": {↔},
      "cms-gen-sim-bmk": {↔},
      "cms-reco-bmk": {↔},
      "atlas-gen-bmk": {↔},
      "lhcb-gen-sim-bmk": {↔},
      "atlas-sim-bmk": {
        "sim": 0.6191,
        "sim_ref": 0.0641
      }
    },
    "score": 3060.2678,
    "score_per_core": 23.908
```



```
"benchmarks": {
  "cms-digi-bmk": {↔},
  "cms-gen-sim-bmk": {↔},
  "cms-reco-bmk": {↔},
  "atlas-gen-bmk": {↔},
  "lhcb-gen-sim-bmk": {↔},
  "atlas-sim-bmk": {
    "run1": {
      "start_at": "Tue Sep  8 15:01:11 2020",
      "report": {
        "wl-scores": {
          "sim": 0.62
        },
        "log": "ok",
        "wl-stats": {↔}
      },
      "duration": 2455,
      "end_at": "Tue Sep  8 15:42:06 2020"
    },
    "args": {
      "-e": 10,
      "-t": 4
    },
    "app": {↔},
    "run_info": {↔},
    "run0": {↔},
    "run2": {↔},
    "version": "v1.1"
  }
},
"settings": {
```


- HEP-Score report
  - Metadata: settings, environment, app_info
  - Exit status
  - Report of each individual benchmark
  - Median Score of each WL benchmark
  - Final Score

Prototypes

- **Standalone container** for **GPU** benchmarking
  - CMS HLT reconstruction (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
- HEP Analysis WL use case for LHC
  - Integrate ROOT **rootbench** in a standalone container reporting scores in a format compliant with HEP-Score
  - Openlab (remote) Summer Student project 2020 (see presentation)
  - In collaboration with the ROOT team

Plans for 2021

- Policy side: contribute to the new **WLCG Task Force** recommended by WLCG MB
  - In charge of defining the **policies** for the adoption of HEPscore as benchmark (pledges, accounting, procurement)
- Software side:
  - Provide more distribution options for HEP Benchmark Suite and HEP Score
    - distribute via python wheel and tarball (in addition to pip install)
  - Include WL containers for non-x86 CPU arch.
  - Include podman as alternative to docker
  - GPU workloads and Analysis workloads
    - From proof-of-concept container to containers fully compliant with the HEP Benchmarks design
    - Define HEP Score configuration for GPU vs CPU comparison
      - Using the same application software

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
  - The software is released under **GPLv3 licence**

- **Opportunity** for HEP community to review the concepts of pledging, accounting, procurement

Useful links

Recent publication (CHEP 2019)

Project repository: https://gitlab.cern.ch/hep-benchmarks

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

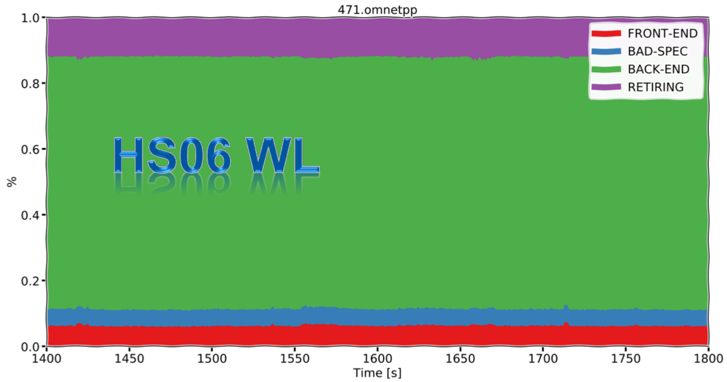
Quantitative comparison with WLCG workloads

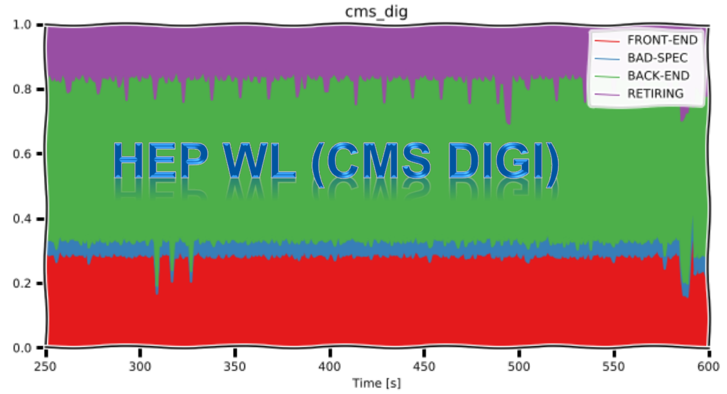
- ❑ Unveil the **dissimilarities** between HEP workloads and the SPEC CPU benchmarks
  - – Using the Trident toolkit
    - • analysis of the hardware **performance counters**

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





Benchmark comparing "speed factors"

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \cdots x_n}$$

https://en.wikipedia.org/wiki/Geometric_mean

- In order to compare servers **HS06** and **HEP-Score** implement the **geometric mean** approach. Needs:
  - a set of reference workloads (WLs)
  - a measure of performance per WL (m_i), that typically goes as [1/s] (eg. can be the event throughput)
  - a reference machine
- The score S of a server (**srv**) is defined as the **geometric mean** of the **speed factors** x_i(**srv**,ref) = m_i(**srv**)/m_i(**ref**) respect to the reference machine (**ref**)
  - i.e. "speed" is *normalised* respect to the reference machine "speed"
- The relative score between srv_A and srv_B is the ratio of the scores S(srv,ref) , this is still a geometric mean of speed factors

	WL_1		WL_2		WL_n		Score	S(A,B)
Ref. Srv	m_1(ref)	1 (by def)	m_2(ref)	1 (by def)	m_n(ref)	1 (by def)	$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$	
Srv A	m_1(A)	x_1(A,ref)	m_2(A)	x_2(A,ref)	m_n(A)	x_n(A,ref)	S(A,ref)	$\frac{S(A,ref)}{S(B,ref)}$
Srv B	m_1(B)	x_1(B,ref)	m_2(B)	x_2(B,ref)	m_n(B)	x_n(B,ref)	S(B,ref)	