CPU/GPU HEP Benchmarking

D. Giordano (CERN)

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

Journées LCG-France, CC-IN2P3

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

CERN

Outline

- ❑ Overview: CPU benchmarking in WLCG
  - – Why benchmarking?
  - – Current HEP-SPEC06 (HS06) and its limitations
- ❑ New approach: benchmarking using HEP experiment workloads
  - – Overview, implementation, status
  - – Applicability to HPCs and GPUs
- ❑ Conclusions

Why benchmarking CPU resources in WLCG?

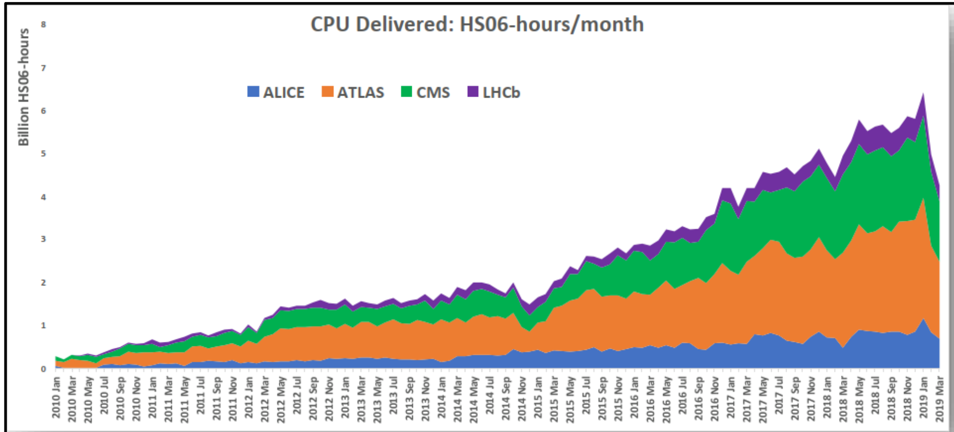
Two main use cases for WLCG:

- Accounting
  - Experiments request “X” CPU resources to do their computing for one year
  - Funding agencies and sites provision “X” CPU resources to the experiments
  - Resource review boards compare the “X” used to the “X” requested
- Procurement
  - Each site buys the CPU resources providing the best “X” per CHF/EUR/…

In addition:

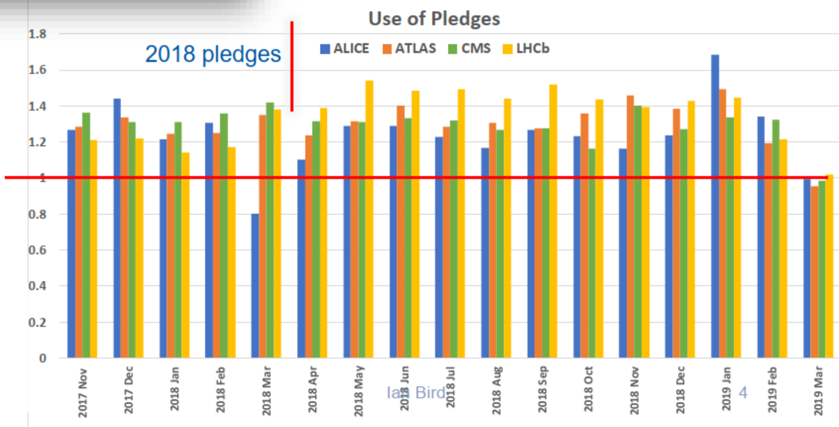
- Job scheduling
- Software optimizations

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



CPU Delivered

New peak: ~270 M HS06-days/month
~ 860 k cores continuous



Ian Bird, April 2019, Computing Resources Review Board

Is HS06 still representative of the WLCG WLs?

Which benchmark shall WLCG adopt after HS06?

Is HS06 still representative of the WLCG WLs?

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  - After this day, further submissions not already under review will not be published by SPEC and technical support for SPEC CPU2006 will end.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/

Follow the HEP sw 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
32-bit
x86 servers
single-threaded/process on multi-core

2019
2 GB footprint (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 do HEP CPU benchmarks!
- As time goes by, *WLCG computing is becoming more and more* ***heterogeneous***
- One of the challenges is how to summarize performance using a single number
  - Unfortunately, this is needed at least for accounting purposes

A reminder about HS06

- Subset of SPEC CPU® 2006 benchmark
  - SPEC's industry-standardized, CPU-intensive benchmark suite, stressing a system's processor, memory subsystem and compiler.

- HS06 is suite of 7 C++ benchmarks
  - In **2009,** proven **high correlation** with experiment workloads

    <<CPP showed a good match with average lxbatch e.g. for FP+SIMD, Loads and Stores and Mispredicted Branches>> [*]
  - Execution time of the full HS06 suite: O(4h)

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

Correlation	Generation	Simulation	Reconstruction	Total
Atlas	0.9969	0.9963	0.9960	0.9968
Alice pp MinBias	0.9994		0.9832	0.9988
Alice PbPb	0.9984		0.9880	0.9996
LhcB	0.9987			
CMS HiggsZZ	0.9982		0.9987	0.9983
CMS MinBias	0.9982		0.9974	0.9974
CMS QCD 80 120	0.9988		0.9987	0.9988
CMS Single Electron	0.9987		0.9942	0.9981
CMS Single MuMinus	0.9986		0.9926	0.9970
CMS Single PiMinus	0.9955		0.9693	0.9955
CMS TTbar	0.9985		0.9589	0.9987

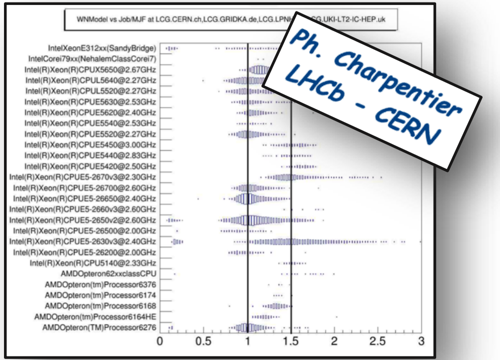
[*] Correlation of HEP-SPEC06 with several kinds of applications and different experiments

[*] "A comparison of HEP code with SPEC benchmarks on multi-core worker nodes"
J. Phys.: Conf. Ser. 219 (2010) 052009 CHEP-09

WG activities - short overview (1)

Fall 2015 *(@ GDBs)*

- Some experiments start reporting performance deviation respect to HS06
- Attention goes to **fast** benchmarks
  - LHCb DB12, Atlas KV, Root stress-test



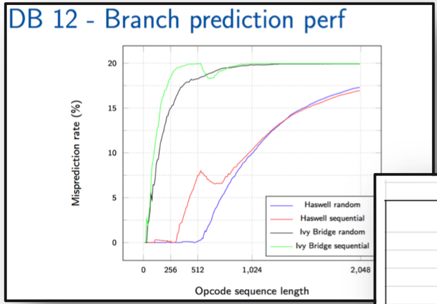
https://indico.cern.ch/event/319754

Spring 2016 *(HEPiX)*

- HEPiX CPU Benchmarking WG restarts coordinated studies

Up to Spring 2017 *(WLCG workshop)*

- Detailed analysis of fast benchmarks
  - In bare metal servers as well as VMs
  - ☞ *Excluded DB12 for lack of robustness*
- Understand systematics on HS06
  - ☞ *E.g. 32-bits Vs 64-bits correction factor*
- Increasing expectations in the future SPEC CPU benchmark

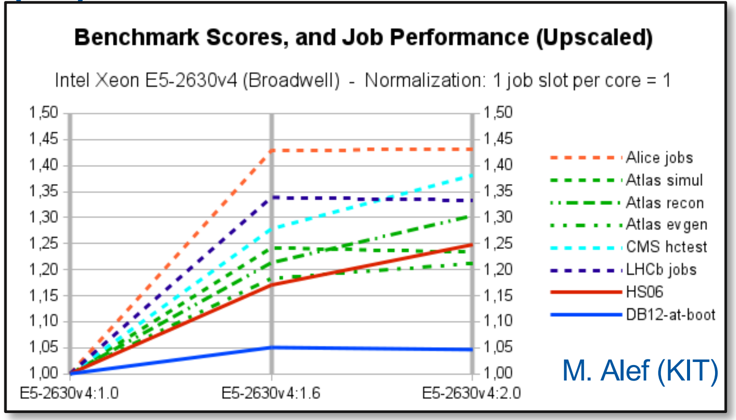


DB12 python

OS	version	32 procs ratio_to_python2.6	ratio 32/16
slc6-base	Python 2.6.6	1	1.00
CC7	Python 2.7.5	1.09	1.02
cc7-base	Python 2.7.5	1.09	1.04
python:2.7	Python 2.7.13	1.18	1.01
python:3	Python 3.6.0	1.05	0.98

https://indico.cern.ch/event/609911/contributions/2620190/

WG activities - short overview (2)



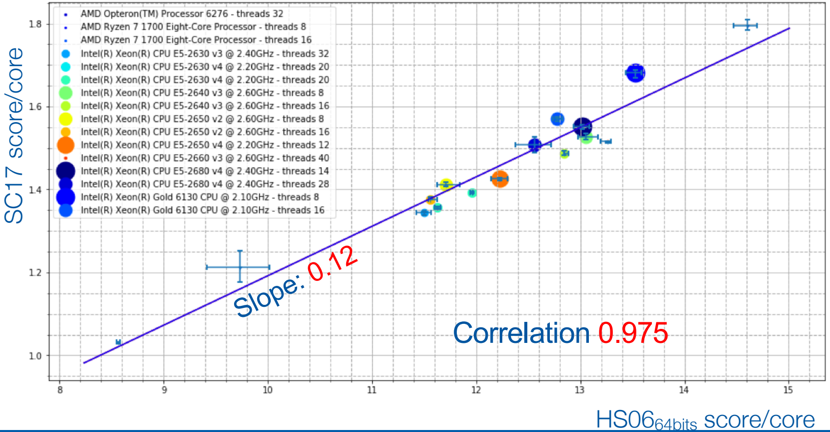
June 2017

- SPEC CPU 2017 finally available
  - Will it solve the HS06 “crisis”?

Up to Summer 2018 *(CHEP)*

- Comprehensive comparisons of HEP Workloads and HS06
  - ☞ *Confirmed discrepancies for Alice and LHCb*

- SPEC CPU 2017
  - Detailed studies w.r.t. HS06
  - ☞ *Extremely high correlation of the 2 benchmark suites*
  - ☞ *No advantage is moving to SPEC CPU 2017*



SPEC CPU2017

SPEC releases major new CPU benchmark suite

The SPEC CPU2017 benchmark suite features updated and improved workloads, use of OpenMP to accommodate more cores and threads, and optional metric for measuring power consumption

Gainesville, Va., June 20, 2017 -- The Standard Performance Evaluation Corp. (SPEC) today released the SPEC CPU2017 benchmark suite, an all-new version of the non-profit group's software for evaluating compute-intensive performance across a wide range of hardware systems.

The SPEC CPU2017 benchmark suite is the first major update of the worldwide standard CPU performance evaluation software in more than 10 years. The new suite includes updated and improved workloads with increased size and complexity, the use of OpenMP to allow performance measurement for parallelized systems with multiple cores and threads, and an optional metric for measuring power consumption.

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

Larger suite, more complex code, shaped for multi-core and multi-threads

https://www.spec.org/cpu2017/Docs/index.html#benchmarks

The Benchmarks

SPEC CPU2017 has 43 benchmarks, organized into 4 suites:

```
SPECrate 2017 Integer            SPECspeed 2017 Integer
SPECrate 2017 Floating Point     SPECspeed 2017 Floating Point
```

Benchmark pairs shown as:

```
5nn.benchmark_r / 6nn.benchmark_s
```

are similar to each other. Differences include: compile flags; workload sizes; and run rules. See: [OpenMP] [memory] [rules]

SPECrate 2017 Integer	SPECspeed 2017 Integer	Language[1]	KLOC[2]	Application Area
500.perlbench_r	600.perlbench_s	C	362	Perl interpreter
502.gcc_r	602.gcc_s	C	1,304	GNU C compiler
505.mcf_r	605.mcf_s	C	3	Route planning
520.omnetpp_r	620.omnetpp_s	C++	134	Discrete Event simulation - computer network
523.xalancbmk_r	623.xalancbmk_s	C++	520	XML to HTML conversion via XSLT
525.x264_r	625.x264_s	C	96	Video compression
531.deepsjeng_r	631.deepsjeng_s	C++	10	Artificial Intelligence: alpha-beta tree search (Chess)
541.leela_r	641.leela_s	C++	21	Artificial Intelligence: Monte Carlo tree search (Go)
548.exchange2_r	648.exchange2_s	Fortran	1	Artificial Intelligence: recursive solution generator (Sudoku)
557.xz_r	657.xz_s	C	33	General data compression

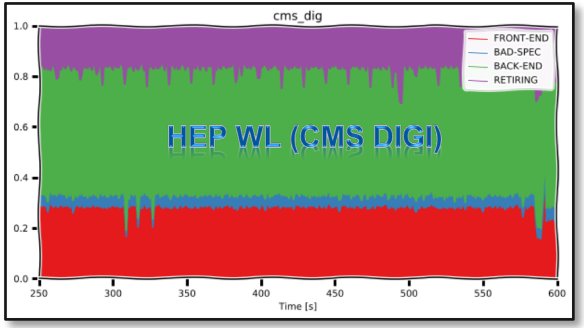
SPECrate 2017 Floating Point	SPECspeed 2017 Floating Point	Language[1]	KLOC[2]	Application Area
503.bwaves_r	603.bwaves_s	Fortran	1	Explosion modeling
507.cactuBSSN_r	607.cactuBSSN_s	C++, C, Fortran	257	Physics: relativity
508.namd_r		C++	8	Molecular dynamics
510.parest_r		C++	427	Biomedical imaging: optical tomography with finite elements
511.povray_r		C++, C	170	Ray tracing
519.lbm_r	619.lbm_s	C	1	Fluid dynamics
521.wrf_r	621.wrf_s	Fortran, C	991	Weather forecasting
526.blender_r		C++, C	1,577	3D rendering and animation
527.cam4_r	627.cam4_s	Fortran, C	407	Atmosphere modeling
	628.pop2_s	Fortran, C	338	Wide-scale ocean modeling (climate level)
538.imagick_r	638.imagick_s	C	259	Image manipulation
544.nab_r	644.nab_s	C	24	Molecular dynamics
549.fotonik3d_r	649.fotonik3d_s	Fortran	14	Computational Electromagnetics
554.roms_r	654.roms_s	Fortran	210	Regional ocean modeling

[1] For multi-language benchmarks, the first one listed determines library and link options (details)

[2] KLOC = line count (including comments/whitespace) for source files used in a build / 1000

same application area as in HS06

WG activities - short overview (3)



Summer 2018

– Proposal of building a set of **HEP reference workloads** (*WLCG MB*)
  - Enable feature studies of the experiments' workloads
  - Build a HEP benchmark suite

Fall 2018

– Collect instructions from LHC experiments to run reference WLs

– **Prototype** the build of HEP reference benchmarks in containers

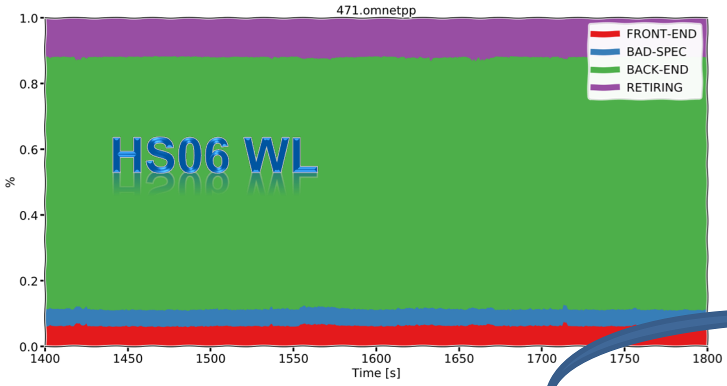
– Studies on hardware **performance counters** (using Trident)

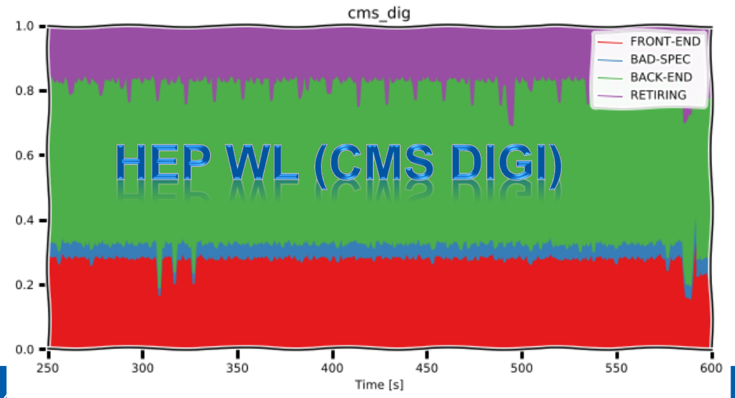
☞ *HEP WLs have same characteristics*
and differ more respect to HS06 and SPEC CPU 2017 workloads

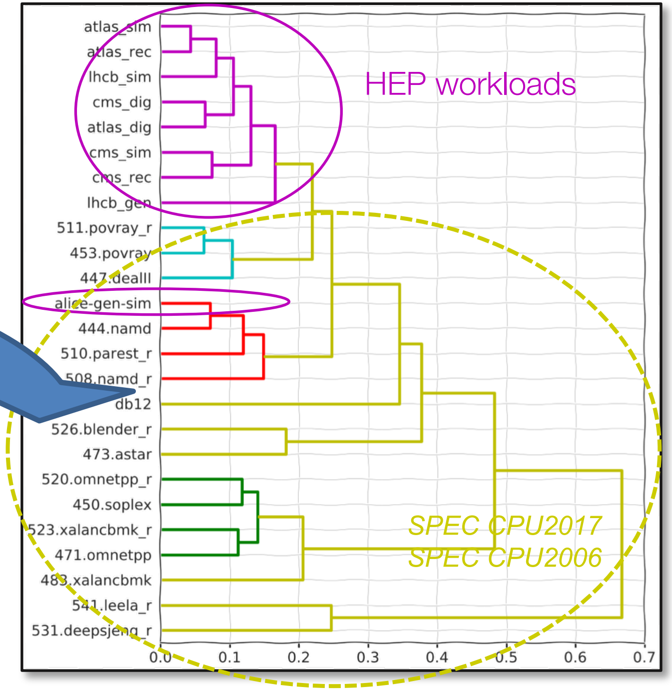
2019

– Start the **HEP Benchmarks project**

Unveil the dissimilarities between HEP WL and the SPEC CPU benchmarks



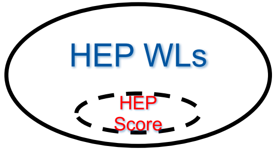


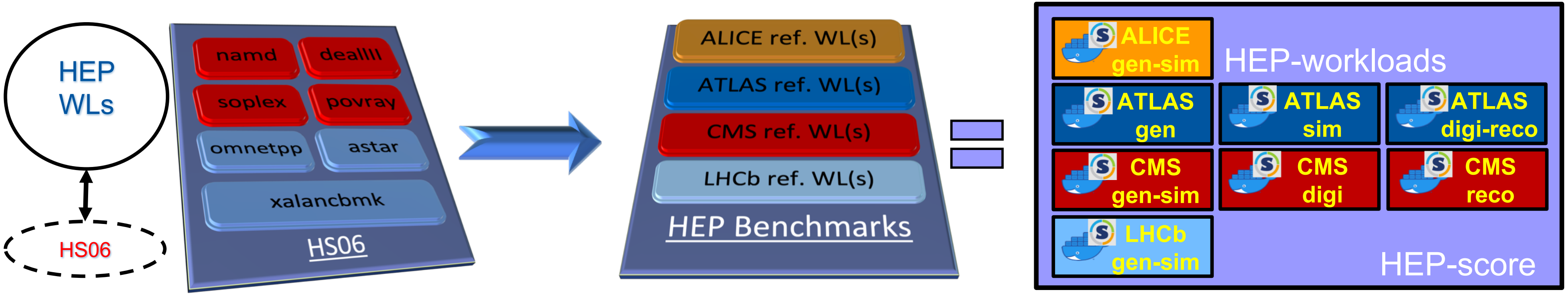


DENDROGRAM OF WL'S SIMILARITY

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





- It is NOT a replacement of HS06 for today
  - This is the future approach to be adopted when the correlation with HS06 will be definitely broken

HEP Benchmarks: project organization

- Current contributors
  - M. Alef (KIT), J. M. Barbet (CNRS-IN2P3), ***O. Datskova*** (CERN), *D. Giordano* (CERN), **C. Grigoras** (CERN), *C. Hollowell* (BNL), ***M. Javurkova-Pagacova*** (UMass)***, V. Khristenko*** (CERN), **D. Lange** (Princeton), M. Michelotto (INFN), ***L. Rinaldi*** (INFN)*,* ***A. Sciabà*** (CERN)***, A. Valassi*** (CERN)
    - Core team for common infrastructure development and overall testing
    - Experiment contacts for experiment-specific software, workloads, metrics

    - *Developers*
    - **Exp. experts**

  - Contact with industry, access to new HW
    - M. Girone (CERN IT-Openlab), L. Atzori (CERN IT-Procurement)

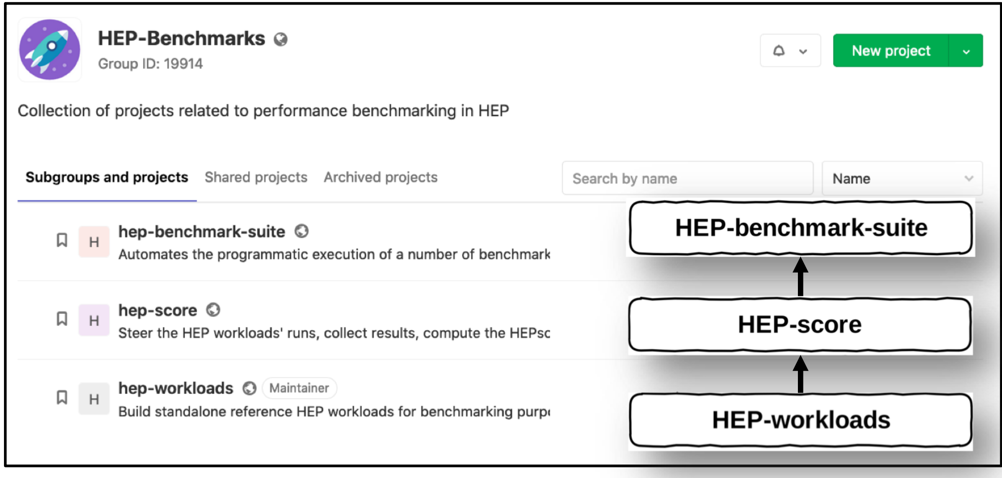
- Track work progress via Jira Project and Twiki
  - Weekly meetings and Jira Sprint reviews

Project repositories

Three repositories @ CERN Gitlab

- *hep-workloads*
  - Common build infrastructure
  - Individual HEP workloads
- *hep-score*
  - "Single-number" benchmark aggregator from several WLs
  - Steer the WLs' run
- *hep-benchmark-suite*
  - Automate execution of multiple benchmarks
  - Publish results

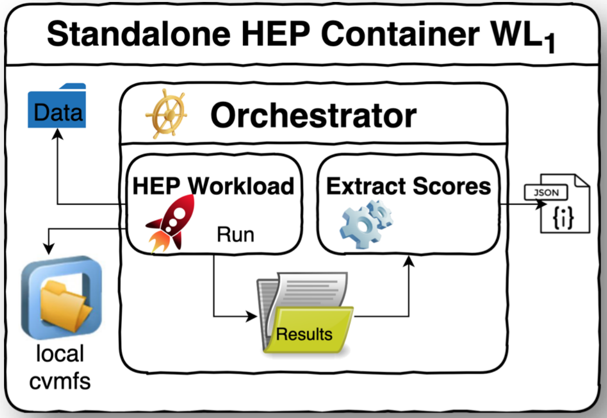
https://gitlab.cern.ch/hep-benchmarks



HEP Workloads

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

- Components of each HEP WL
  - SW repository (OS and CVMFS)
  - Input data (event and conditions data)
  - An orchestrator script (benchmark driver)
    - Sets the environment
    - Runs (many copies of) the application
      - Each copy may be multi-process or multi-threaded
    - Parses the output to generate scores (json)

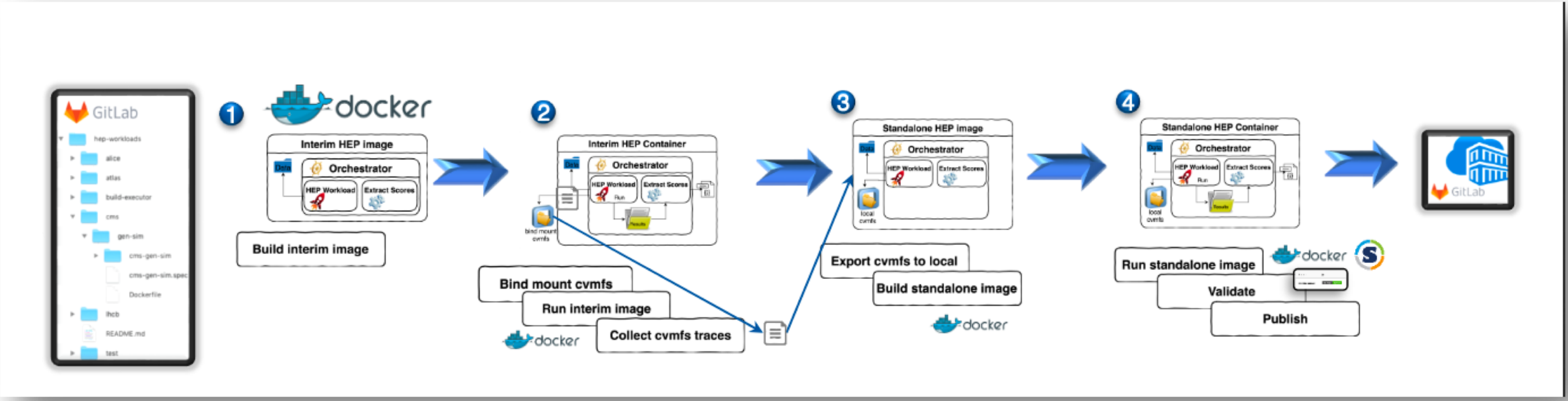


Common and WL scripts: /bmk
Experiment WL software: /cvmfs
Experiment WL data files: /data
O/S add-ons via yum install
O/S: Scientific Linux CERN 6

Container images are made up of layers

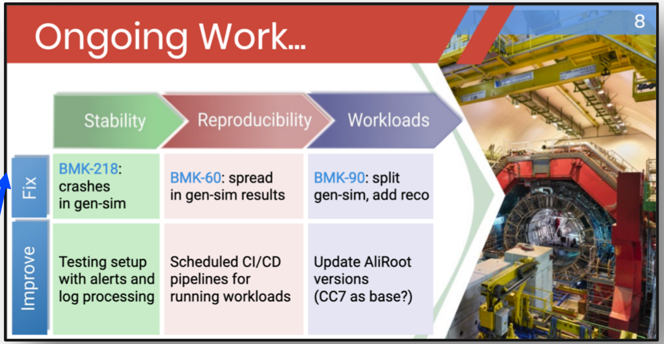
Solid Build Infrastructure

- ❑ Individual HEP workload container images are **built**, **tested** and **distributed** via gitlab



- ❑ Can be executed both via **Docker** and **Singularity**

Readiness of HEP Workloads



- ❑ Criteria:
  - – Reproducibility, evaluated via a “pessimistic” spread:
    - $(score_{max} – score_{min})/score_{mean}$
  - – Robustness
    - • Do not crash. Properly reports failures
  - – Runtime duration
  - – Memory utilization (< 2 GB/core)
  - – Image size (mainly dominated by input data)

Workload	ALICE gen-sim	ATLAS gen	ATLAS sim	ATLAS digi-reco	CMS gen-sim	CMS digi	CMS reco	LHCb gen-sim
Robustness	✘	✔	✔	✔	✔	✔	✔	✔
Reproducibility	0.5%	0.8%	2%	0.6%	1.5%	1%	1%	1%
Runtime duration	~ 12'	~ 20'	~ 3h	~ 25'	~ 70'	~ 15'	~ 30'	~ 40'
Memory	✔	✔	✔	✔	✔	✔	✔	✔
Events_per_thread	20	500	20	30	100	100	100	5
Image size (unpacked)	1.8 GB	1.5 GB	2.0 GB	6GB	10 GB	6.5 GB	5.5 GB	2.6 GB
Readiness	✘	✔	✔	✔	✔	✔	✔	✔

✔ okay
✘ blocker

- ❑ All workload types are covered, with more than one experiment code

Optimizations

- Current focus is on shortening the **running time** without affecting the **precision**
- A single round of all HEP-WLs takes 6h35'.
  - Will be shortened to 3h30'
    - Can it be reduced more? W.I.P
- As a reference HS06 runs in ~3h, executing 3 rounds of all benchmarks

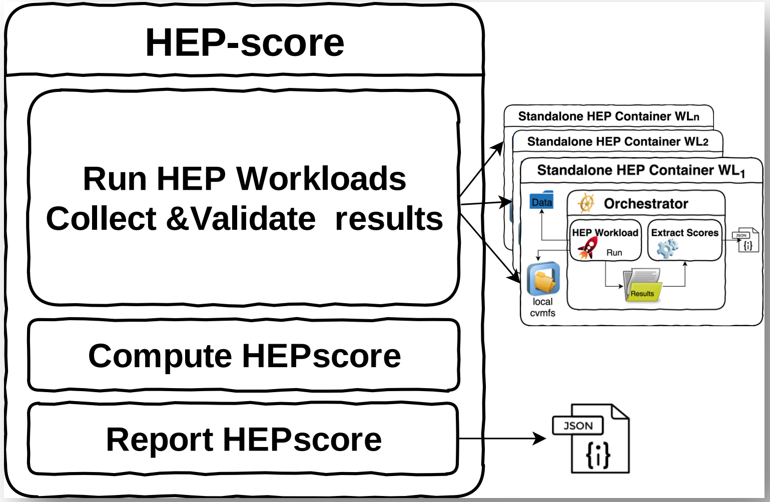
WL	Events: Default (smaller)		Duration of a single run [hh:mm]	
Atlas gen	500	(200)	~0:20	(~12)
Atlas sim	20	(10)	~3:10	(~1:32)
Atlas digi-reco	30		~0:30	
CMS gen-sim	100	(20)	~1:13	(~0:15)
CMS digi	100	(50)	~0:16	(~0:9)
CMS reco	100	(50)	~0:30	(~0:15)
LHCb gen-sim	5		~0:40	
Total			**~ 6:35**	**(~3:30)**

HEP Score

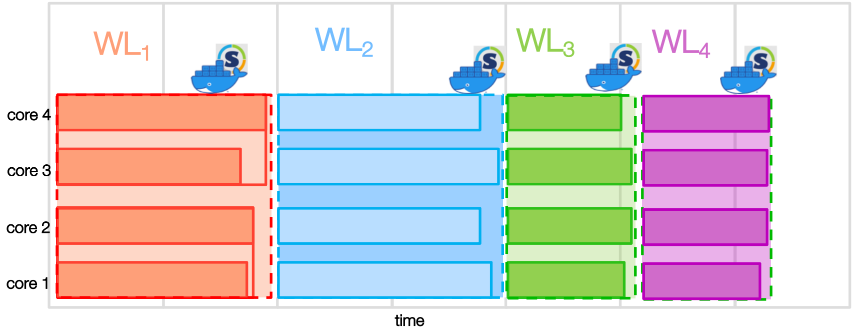
- Orchestrate the run of a series of HEP Workloads

- 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 used by other domains



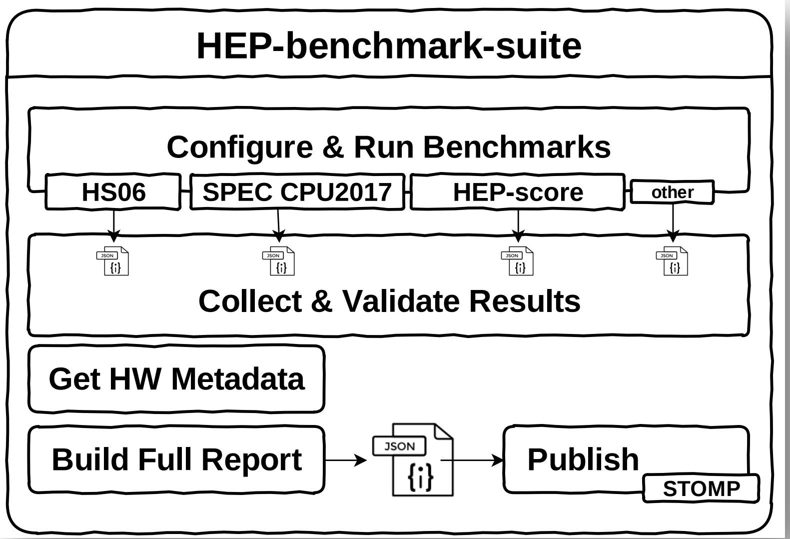
HEP Score running mode



- 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***

HEP Benchmark Suite

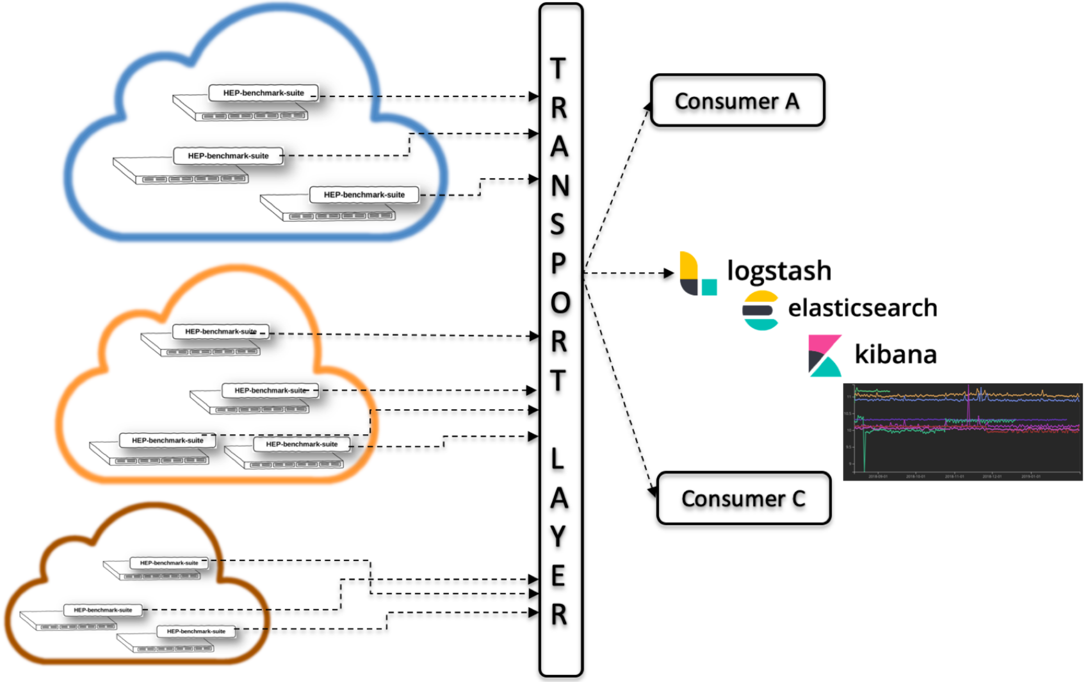
- ❑ Control the execution of several benchmarks
  - HEP-score, SPEC CPU2017, HS06, KV, DB12, …
  - NB: does NOT distribute sw under proprietary license (such as HS06), just requires that code to be pre-installed
- ❑ Simplify the sharing, tracking and comparison of results
  - Metadata track data-centre name, CPU model, host, kernel version, …
- ❑ A sort of "Push the button" & Run & Get Results



Centralise the benchmark data storage

- Global results "publishable" to a messaging system in json format
  - Ideal for monitoring and offline analysis

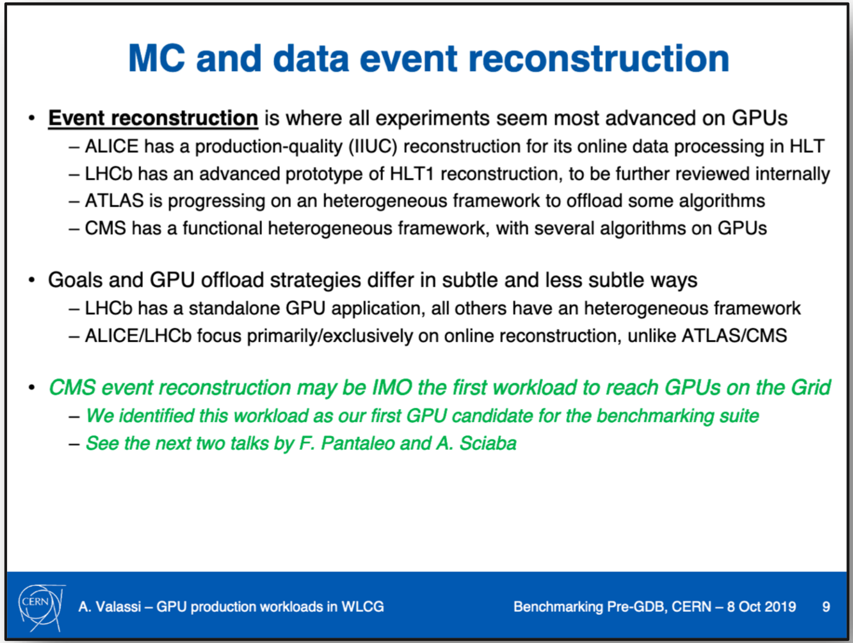
- Adoption
  - @ CERN to continuously benchmark available CPU models
  - Used in CERN commercial cloud procurements (HNSciCloud)
  - Tested by other site managers (GridKa, RAL, INFN-Padova, …)



Benchmarks for heterogeneous resources

- In the future WLCG resources will likely include HPCs with GPUs
  - "How to value pledged HPC resources"?
    - WLCG MB requested to investigate approaches

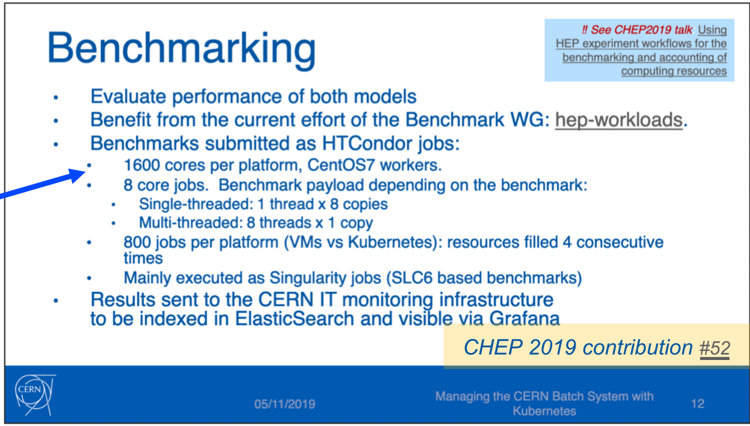
- First demonstrator of **standalone container** for **GPU** benchmarking available
  - Based on CMS reco with GPUs (Patatrack)
    - Pixel track reconstruction, Calorimeter reconstruction
  - Essential for us, to understand how to apply the approach used for **CPU HEP Benchmarks** to the **CPU+GPU** system
- Other GPU applications welcome
  - LHC simulation, ML production applications, …



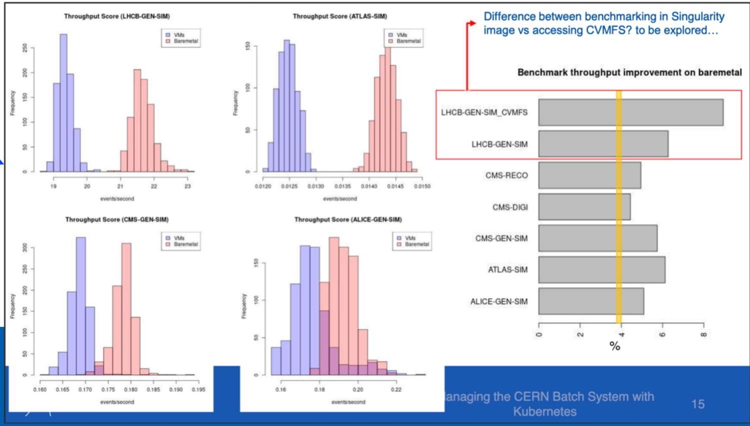
Find here a recent seminar on GPU adoption in HLT farms

Early adopters

- ❑ The HEP-Benchmarking-Suite will drive the CPU benchmark of all node in the CERN data centre
  - – Embedded into the Openstack Ironic enrollment procedures
- ❑ HEP-Workloads are being used to test/improve the CERN batch infrastructure
  - – First large scale tests



- ❑ Successful examples of runs in HPC centres and/or on new x86 CPU models
  - – eg: AMD EPYC 7702P 64-Core Processor (128 HT)
    - • x 1.15 score of a Intel E5-2640v3 (16 HT threads) when normalized to the number of threads



- ❑ Looking for many more adopters: could be YOU!

Ongoing work

- ❑ Validation of the running WLs and produced "**scores**"
- ❑ Inclusion of new CPU/GPU WLs
- ❑ Consolidation of the WLs' report
- ❑ Consolidation of the HEP Score and HEP Benchmark Suite implementations
- ❑ Studies
  - Compare Docker Vs Singularity HEP scores
  - Run at large scale on production nodes
    - Compare performance of HEP benchmarks and standard jobs
    - Compare with HS06 and SPEC CPU 2017

All those areas would greatly profit of new contributors

Conclusions

- After 10 years, HEP-SPEC06 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 exist for all workloads provided by the LHC experiments
    - GEN-SIM of all four experiments, DIGI and RECO of CMS and ATLAS

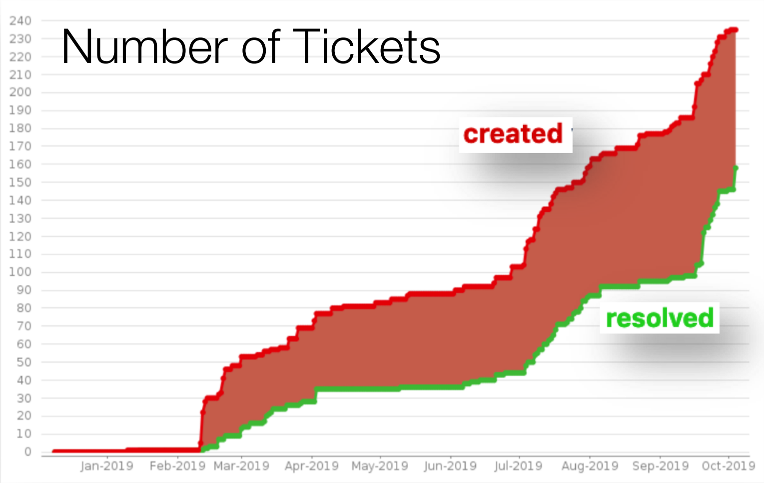
- The full **HEP Benchmarks** chain is in place
  - Already used by a number of beta testers

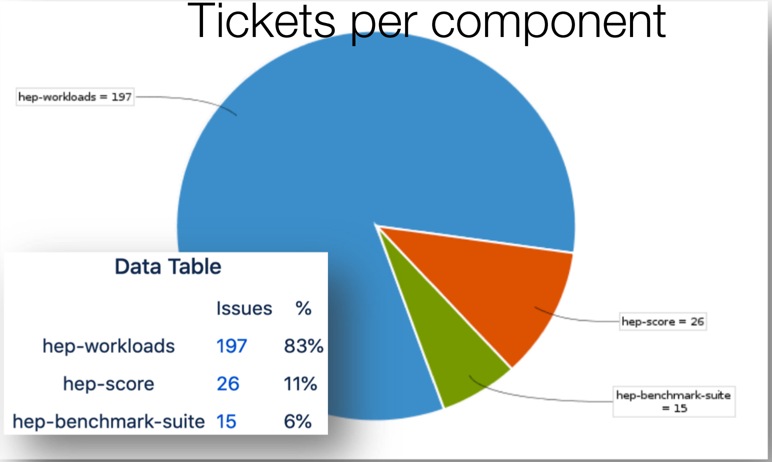
- Outlook: can extend the idea and implementation to HPCs and non-x86 resources
  - A container for a workload with optional GPU offload (CMS Patatrack) is being tested

CERN

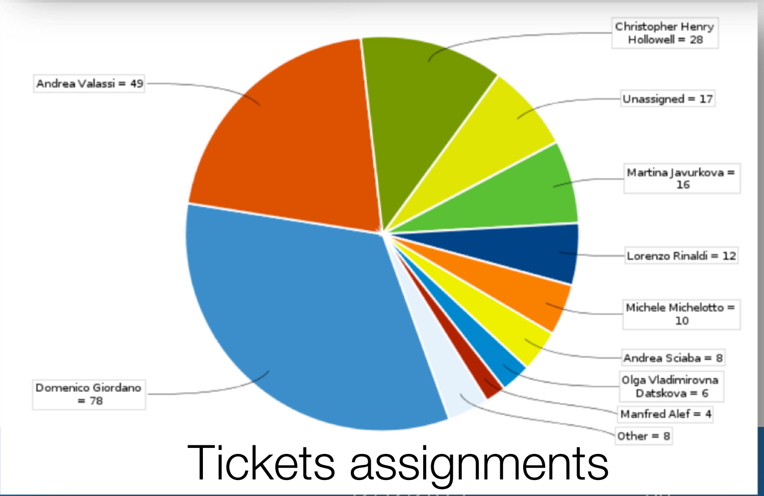
Jira Statistics

- Work progression mainly in few months of activity
- Collective work and major attention gone to HEP-Workloads
- Main activities:
  - Develop the infrastructure
  - Develop the Experiment specific wrappers (drivers)
  - Validation





	Issues	%
hep-workloads	197	83%
hep-score	26	11%
hep-benchmark-suite	15	6%



CMS Patatrack in container

Felice Pantaleo: for the Patatrack Team
for the CMS Experiment
felice@cern.ch

- A GPU-based full reconstruction of the Pixel detector from RAW data decoding to Pixel Tracks and Vertices determination, ECAL and HCAL local reconstruction has been implemented
- This reconstruction is fully integrated in the CMS Software
- Conversion to the legacy data formats and the standard validation can be run on demand
- Can achieve better physics performance, faster computational performance at a lower cost with respect to the baseline solution
- Working ongoing to develop the entire Phase-2 High Granularity Calorimeter Reconstruction directly with Heterogeneous Programming techniques
- Portable code is key for long-term maintainability, testability and support for new accelerator devices
    - Ongoing study and comparisons of solutions in Patatrack for CMS reconstruction
    - Starting from a CUDA code makes life **much** easier

14

How to create the benchmark

1. Get all the executables and binaries
2. Get the input data and prepare it
3. Have a script that runs Patatrack and extracts a score
4. Package everything in a Docker image
5. Distribute it!

Viktor Khristenko
Andrea Sciabà

Where are we?

- Simplified instructions to run the job by hand
    - https://github.com/sciaba/patatrack-tests
- CMS Open data set as input
    - In the working group EOS project space, takes 5 GB
- Scripts to build the Docker image
    - https://github.com/vkhristenko/
    - (the latest version is not yet committed)
- Image
    - Very large, about 50 GB – to become much smaller once the binaries are in CVMFS
    - All configuration parameters still hard-coded
    - Not yet publicly available
    - Still needs network connection for Frontier