SPEC Benchmarks in the Suite

D. Giordano (CERN/IT)

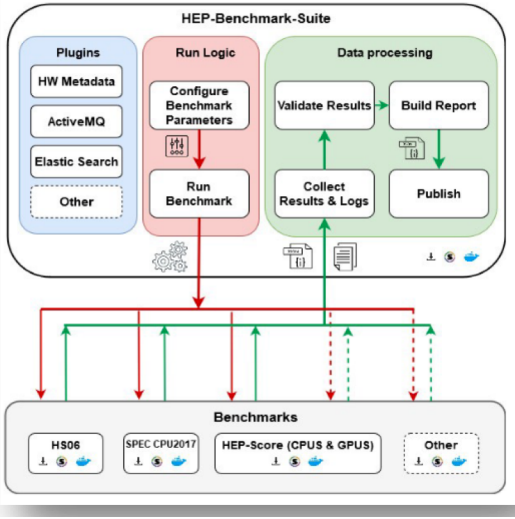
on behalf of
HEPiX CPU Benchmarking WG
hepix-cpu-benchmark@hepix.org

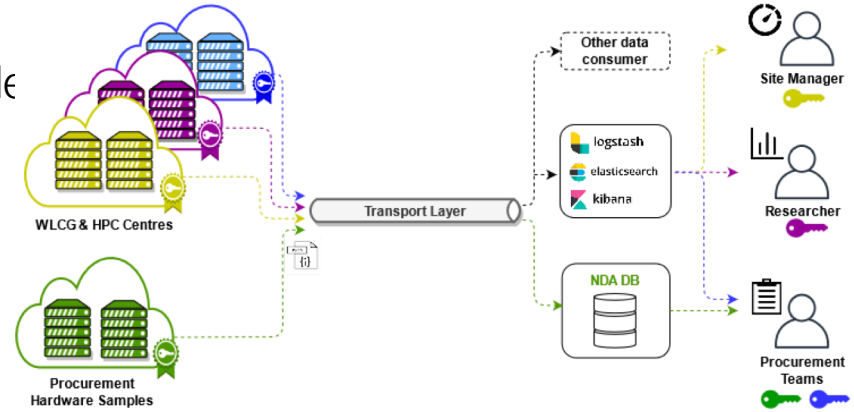
WLCG HEPscore task force
17 March 2020

- A mix of slides from the current HEPiX workshop and old presentations

HEP Benchmark Suite

- Meta-orchestrator for the execution of several benchmarks
  - HS06, HEP Score, SPEC CPU 2017, DB12 (mainly for functional tests!), …
- Features
  - Modular design, fully rewritten in python3.6+
  - Distributed via pip install, python wheels available
  - Metadata section with detailed HW information
  - Very few dependencies needed
  - Install as unprivileged user
    - HPC / Grid compatible





HS06 and SPEC CPU 2017

- Make sure that HS06 and SPEC CPU 2017 can run via the Suite
  - Orchestrator scripts and libraries available in a container image, built at https://gitlab.cern.ch/hep-benchmarks/hep-spec/
  - NB: the SPEC suites are not included for license reasons, have to be pre-installed on the host
- HS06
  - Config is linux_gcc_cern.cfg used by HS06 in the last decade, with few adaptations
- SPEC CPU 2017
  - Default “HEP” benchmark set to mimic HS06: Cpp-rate set of benchmarks
    - Benchmark set can be reconfigured. Eg. –b intrate will run SPEC INT 2017 rate
    - NB: All configuration changes are tracked in the reported results
  - Config similar to linux_gcc_cern.cfg, distinct for x86 and ARM

HS06 for ARM cpus

```
We do not appear to have working vendor-supplied binaries for your
architecture.  You will have to compile the tool binaries by
yourself.  Please read the file
   SPEC_CPU2006_v1.2/Docs/tools-build.html
for instructions on how you might be able to build them.
Please only attempt this as a last resort.
```

- Enable support for ARM
  - Multi-architecture container
    - gitlab-registry.cern.ch/hep-benchmarks/hep-spec/hepspec-cc7-multiarch:v2.0
  - SPEC CPU 2017 already supports ARM cpus. Only CPU model needs to be changed when running on ARM
  - HS06 too old to support natively ARM cpus: SPEC 2006 tooklit needed to be built to work
    - Build the toolkit following instructions https://www.spec.org/cpu2006/Docs/tools-build.html after patching some old code
      - Patch procedure available https://gitlab.cern.ch/hep-benchmarks/hep-spec/-/tree/master/patch_SPEC2006
- NB: patching the SPEC 2006 toolkit is one time operation
  - The toolkit is then included in the tool/bin area: creating an archive for SPEC CPU 2006 allows to use it in any other aarch64 machine
    - At CERN this is available as usual as tarball (clearly it cannot be shared!)
- HS06 for ARM only supported at 64 bits

Running HS06 on ARM

AWS Graviton2 bare-metal server benchmarked using the multi-architecture container (see previous slide)

```
$ echo ${SECRET_URL} | /hep-spec/scripts/hep-spec.sh -w $BMK_RUNDIR -b $BMK -m $BMK_OPTION -p ${SPEC_DIR} -u -
############################
          CERN HEPSPEC
Sat Feb 27 17:16:00 UTC 2021
############################
2021-02-27T17:16:00 [/hep-spec/scripts/hep-spec.sh] Variable values:
       HEPSPEC_SOURCEDIR=/hep-spec/scripts
       HEPSPEC_BMK=hs06
       HEPSPEC_NUMPROC=64
       HEPSPEC_PATH=/scratch/HEPSPEC/CI_hs06_ext_g2_bare_12384842
       HEPSPEC_SET=all_cpp
       HEPSPEC_MACHINE_OPTION=default
       HEPSPEC_ITERATIONS=3
       HEPSPEC_WORKDIR=/scratch/jobs/hs06_ext_g2_bare_12384842/hep-spec
       HEPSPEC_DEBUG=0
```

```
$ lscpu
Architecture:                    aarch64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
CPU(s):                          64
On-line CPU(s) list:             0-63
Thread(s) per core:              1
Core(s) per socket:              64
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       ARM
Model:                           1
Model name:                      Neoverse-N1
Stepping:                        r3p1
BogoMIPS:                        243.75
L1d cache:                       4 MiB
L1i cache:                       4 MiB
L2 cache:                        64 MiB
L3 cache:                        32 MiB
NUMA node0 CPU(s):               0-63
```


```
{"hs06":{"start":"Sat Feb 27 17:19:26 UTC 2021", "end":"Sat Feb 27 20:10:46 UTC 2021", "copies":64,
            "runcpu_args":"1 runspec: runspec --define machine_option:64 --config=linux_gcc_cern.cfg --action=build all_cpp;64 runs
pec: runspec --define machine_option:64 --config=linux_gcc_cern.cfg --nobuild --noreportable --iterations=3 all_cpp;", "bset":"all
_cpp", "LINK":"  6  g++ -O2 -fPIC -pthread  -DSPEC_CPU_LP64    <objects>     -o options;  1  g++ -O2 -fPIC -pthread  -DSPEC_CPU_LP
64 -DSPEC_CPU_LINUX   <objects>     -o options;", "hash":"7b84bb375cee11731a958a26d6fc155d",
            "score":1170.998, "avg_core_score" : 18.296, "num_bmks":7 ,"bmks":{ "444.namd":[  23.5, 23.5, 23.5, 23.5, 23.5, 23.5, 2
```

Some consideration from the past

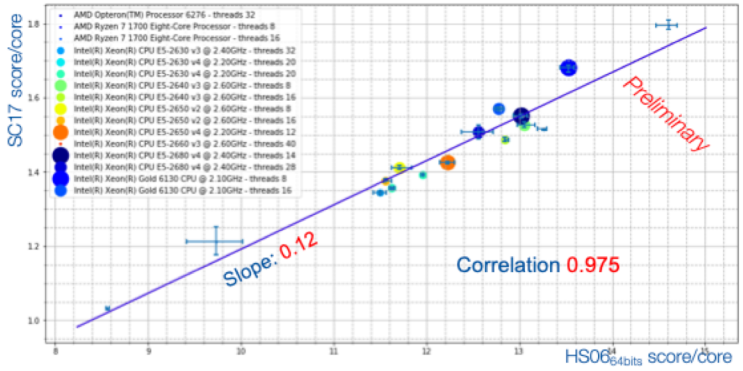
At the time of the study, no advantage seen in adopting SPEC CPU 2017

SPEC CPU2017

Waited since years, considered the successor of HS06, and possibly the solution for all our issues
- Is that true?

Essential to dissect and understand SPEC CPU2017
- Understand the configuration: benchmark mix, compiler flags, **rate** Vs speed
- Study relationship w.r.t. **HS06** and **WLCG job mix**

Current understanding:
- The CPU2017 suite of C++ benchmarks is
  - longer to run
  - **highly correlated** with HS06 score
- The WLCG job mix won't be better represented by SPEC CPU2017 respect to HS06
  - **No reason to move away from HS06 now!**
  - Conclusions can change when the new HEP workloads are delivered (I doubt!)



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)
- Since 2009 HS06 has been used for
  - Performance studies
  - Procurement procedures
  - Pledges & Accounting

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

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

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

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

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

same application area as in HS06

Are the individual benchmarks independent?

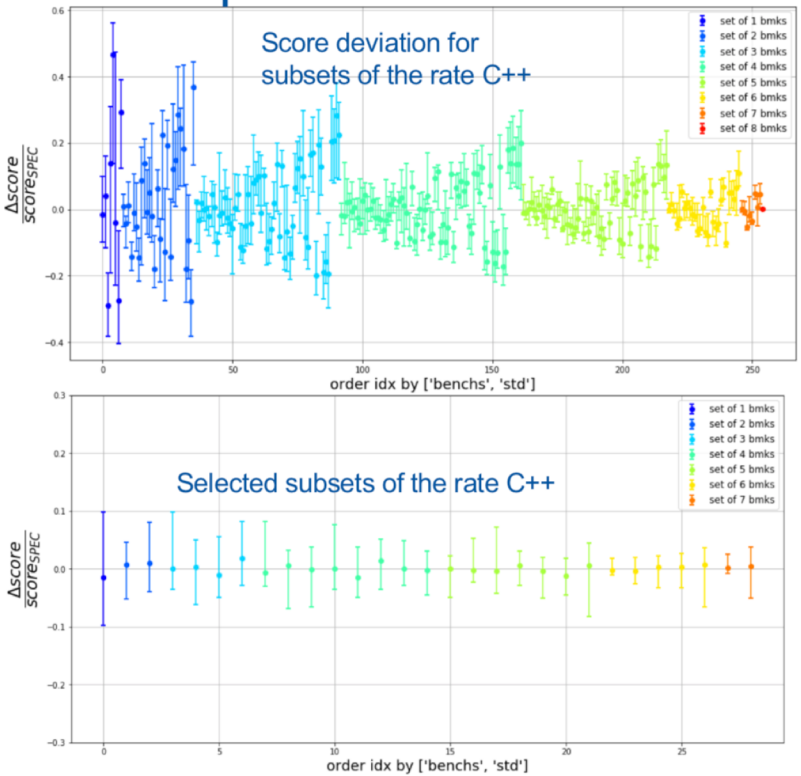
Are all SPEC CPU® 2017 benchmarks needed?

- Less benchmarks => Shorter runtime
  - Currently a run of the 8 **C++** benchmarks takes <2.5hour/iteration> in the 7 tested CPU models
- Better control of benchmark score Vs HEP job mix

Found subsets of the **rate C++** still well representative of the full performance score

- Example:
  - 508.namd_r , 520.omnetpp_r, 526.blender_r Discrepancy
    - » max= 0.06
    - » mean= -0.003 ± 0.004

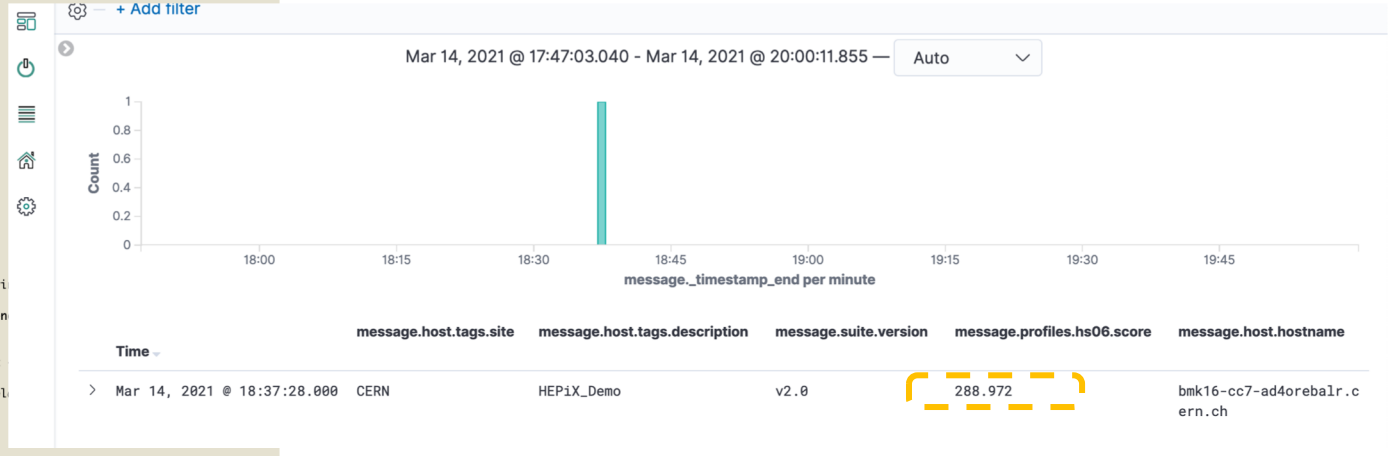
Limitation of the study: focusing on x86 arch, mainly Intel CPUs



Back to the future: Run HS06 by the Suite

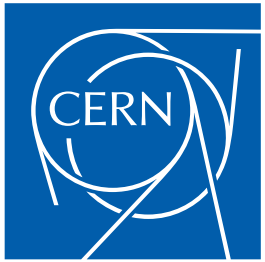
```
cat > $WORKDIR/bmkrun_config.yml <<EOF2
activemq:
  server: dashb-mb.cern.ch
  topic: /topic/vm.spec
  port: 61123  # Port used for certificate
  ## include the certificate full path (see documentation)
  key: '/root/userkey.pem'
  cert: '/root/usercert.pem'

global:
  benchmarks:
  - hs06
  mode: singularity
  publish: true
  rundir: ${WORKDIR}/suite_results
  show: true
  tags:
    site: CERN
    description: "HEPiX_Demo"

hepspec06:
  # Use the docker registry
  image: "docker://gitlab-registry.cern.ch/hep-benchmarks/hep-spec/hepspec-cc7-multiarch:v2.0"
  # URL to fetch the hepspec06. It will only be used if the software
  # is  not found under hepspec_volume.
  url_tarball: [redacted]

  # Define the location on where hepspec06 should be found
  # If hepspec06 is not present, the directory should be writeable
  # to allow the installation via the url_tarball
  hepspec_volume: "/tmp/SPEC"

  ## Number of iterations to run the benchmark
  iterations: 3
  ## Specifies if benchmark is run on 32 or 64 bit mode
  ## Default is 64-bit
  mode: 32
EOF2

cd $WORKDIR
export MYENV="env_bmk"        # Define the name of the environment.
python3 -m venv $MYENV        # Create a directory with the virtual envi
source $MYENV/bin/activate    # Activate the environment.
python3 -m pip install git+https://gitlab.cern.ch/hep-benchmarks/hep-ben
cat bmkrun_config.yml

if [ `cat bmkrun_config.yml | grep "this_is_dummy_replace_me" | grep -c
then
  echo -e "\nERROR. You are using the url_tarball parameter. Please repl
  exit 1
fi
bmkrun -c bmkrun_config.yml
```

- Similarly to run HEP Score
  - Download the example
  - Modify few parameters & run
- SPEC CPU 2017 is similar (read the doc)



Conclusions

- The HEP Benchmark Suite orchestrates not only HEP Score
- HS06 and SPEC CPU 2017 are also included
  - ARM CPUs are also supported
- SPEC CPU 2017 can run any config
  - a HS06 “all_cpp” like, Int Rate, …
- Other benchmarks can be integrated, as dedicated plugins


CERN