HEPscore Application – Design, Status and Plans

Chris Hollowell <hollowec@bnl.gov>
On behalf of the HEPiX Benchmarking Working Group

Benchmarking Pre-GDB – 10/8/2019

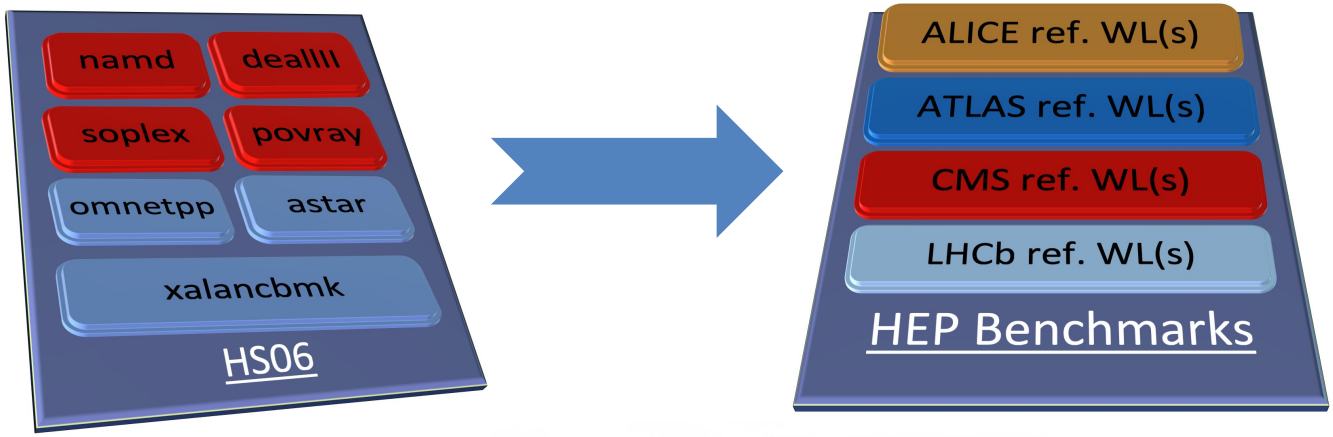
BROOKHAVEN NATIONAL LABORATORY | Scientific Data and Computing Center

U.S. DEPARTMENT OF ENERGY

HEPscore Application – Overview and Design

The HEPiX Benchmarking Working Group has started the process of creating an alternative to HEPSPEC06

Based on containers from the HEP Workloads project

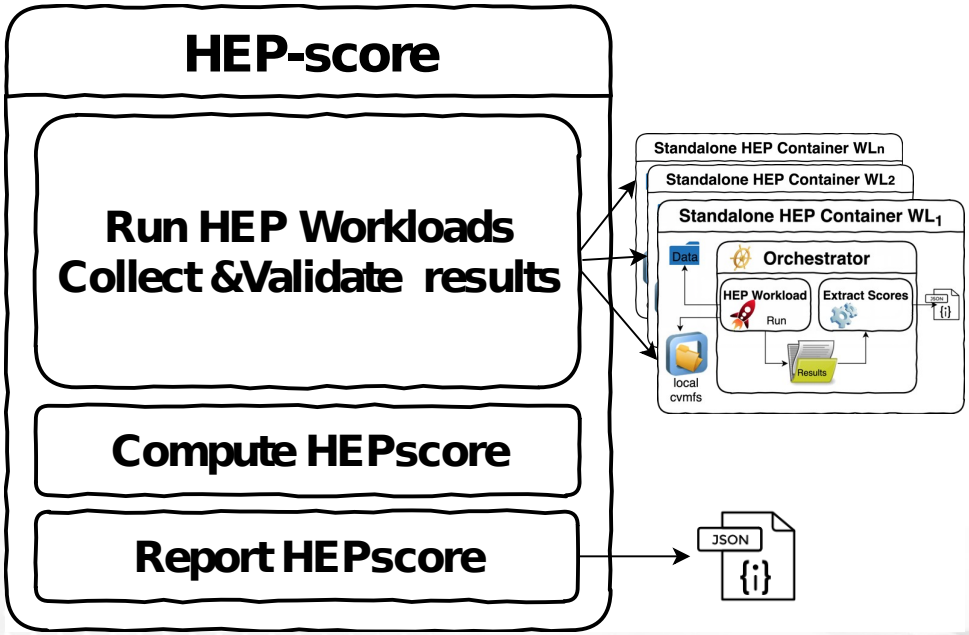


HEPscore Application – Overview and Design (Cont.)

In order to do this, it was necessary to develop an application which:

1. Orchestrates the sequential execution of the benchmark containers
2. Collects the results
3. Computes a final overall score

Functionality implemented in the “HEPscore” utility



HEPscore Application – Overview and Design (Cont.)

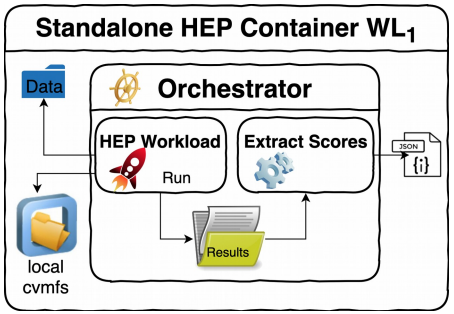
Desired features/requirements

- Decouple the workload applications from the orchestration and overall score computation
    - Allow users to create their own container benchmark suites using custom YAML configuration files
    - Same application could be used to create future versions of the recommended HEPiX/WLCG benchmark
- Develop in Python
- Attempt to minimize 3rd party software requirements
    - Ideally, allow users to run without installing any non-stock software on CentOS7/SL7 OS systems (other than Docker or Singularity)
- Function with both Singularity and Docker
- Traceability of the configuration and workload code versions

HEPscore Application – Overview and Design (Cont.)

- Produce JSON/YAML output

  - Include individual benchmark container JSON output data in overall HEPscore JSON output, so that overall score can be validated




```
"report": {
  "wl-scores": {
    "gen-sim": 0.4438
  },
  "wl-stats": {
    "CPU_score": {
      "max": 0.0226,
      "score": 0.1123,
      "median": 0.0225,
      "avg": 0.0225,
      "min": 0.0222
    },
    "throughput_score": {
      "max": 0.0892,
      "score": 0.4438,
      "median": 0.089,
      "avg": 0.0888,
      "min": 0.088
    }
  },
  "log": "ok",
  "app": {
    "bmkdata_checksum": "e57b3ad19144b7e9574b97056fb35d11",
    "cvmfs_checksum": "b2ab0e3bd4ba1333ebfc7dc49a024536",
    "bmk_checksum": "fc73ae9f18c4ef90791f097cd31b45dc",
    "version": "v1.0",
    "description": "CMS GEN-SIM of ttbar events, based on CMSSW_10_2_9"
  },
  "threads_per_copy": 4,
  "copies": 5,
  "events_per_thread": 100
},
```

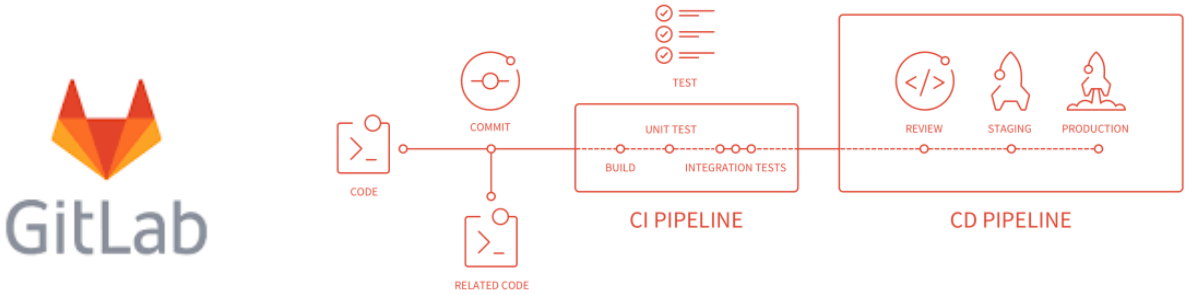
HEPscore Application – Status

Functioning beta application available for testing

- The application itself is currently a single Python script – simplifies downloading/installation
    - No need to install an RPM, or run installation scripts to execute
- If a configuration file is not passed to the benchmark, the built-in configuration is used
    - Currently, the built-in configuration is for an early prototype “HEPscore19” benchmark

Fully integrated into the HEP Benchmark Suite project (thanks to Domenico Giordano)

HEPscore Application – Status (Cont.)



Various Gitlab continuous integration tests in place (also thanks to Domenico)

hep-score project in CERN's Gitlab:

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

- After cloning, possible to run "pip install ." to install as "hep-score", along with all needed dependencies

To download the "hepscore.py" script directly:

https://gitlab.cern.ch/hep-benchmarks/hep-score/raw/master/hepscore/hepscore.py?inline=false

HEPscore Application – YAML Configuration

Example HEPscore benchmark YAML configuration:

```
hepscore_benchmark:
  name: HEPscoreTest
  version: 0.35
  repetitions: 3  # number of repetitions of the same benchmark
  reference_machine: 'Intel Core i5-4590 @ 3.30GHz - 1 Logical Core'
  method: geometric_mean # how to calculate overall score
  registry: gitlab-registry.cern.ch/hep-benchmarks/hep-workloads
  scaling: 10.0
  benchmarks:
    atlas-sim-bmk:
      version: v0.18
      scorekey: wl-scores
      ref_scores:
        sim: 0.0052
    cms-reco-bmk:
      version: v0.11
      scorekey: wl-scores
      refs_scores:
        reco: 0.1625
      events: 2
      threads: 4
      copies: 1
      debug: true
```

HEPscore Application – YAML Configuration (Cont.)

Individual benchmarks specified under the “benchmarks” parameter

- Various flags for the benchmark containers implemented in the config: debug, events, copies and threads

```
benchmarks:
  cms-reco-bmk:
    version: v0.11
    scorekey: wl-scores
    ref_scores:
      reco: 0.162
    events: 2
    threads: 4
    copies: 1
    debug: true
```

The docker registry used for the benchmark containers is also changeable in the configuration

```
registry: example.com/reg
```

Allows for multiple executions of the same benchmark

```
repetitions: 3
```

- Median is taken in this case
- Allows for the same behavior as implemented in SPEC CPU2006
  - 3 runs of the same sub-benchmark with median reported
- Helps eliminate anomalous run data

HEPscore Application – YAML Configuration (Cont.)

"geometric_mean" is supported to compute the overall score for all of the benchmark containers – most suitable function to utilize in this context

```
method: geometric_mean
```

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

- Standard statistical method to summarize unrelated data
  - Also utilized in HEPSPEC06
- Considering adding a weighted geometric mean function in the future

Implementation of benchmark workload reference scores in the configuration

```
reference_machine: E5-2660
benchmarks:
  atlas-sim-bmk:
    ref_scores:
      sim: 0.162
```

- The score of each workload on the defined reference machine becomes "1"
- Specified via the "reference_machine" and "ref_scores" parameters
- Also similar to SPEC CPU2006

Support for scaling the final overall score

```
scaling: 10.0
```

HEPscore Application – Execution

```
HEPscore Benchmark Execution - Version 0.64
hepscore.py {-s|-d} [-v] [-V] [-y] [-o OUTFILE] [-f CONF] OUTDIR
hepscore.py -h
hepscore.py -p [-f CONF]
Option overview:
-h           Print help information and exit
-v           Display verbose output, including all component benchmark scores
-d           Run benchmark containers in Docker
-s           Run benchmark containers in Singularity
-f           Use specified YAML configuration file (instead of built-in)
-o           Specify an alternate summary output file location
-y           Specify output file should be YAML instead of JSON
-p           Print configuration and exit
-V           Enable debugging output: implies -v
```

The default built-in configuration can be displayed with “-p”

HEP Workloads containers attempt to utilize all logical cores by default

Both Docker (“-d”) and Singularity (“-s”) execution of the containers is supported

Requires the PyYAML module be installed (available in stock CentOS/SL)

OUTDIR must be specified and must exist

- Subdirectory named HEPscore_DATETIME is created underneath and shared with the containers as the “/results” area

- Container stdout/stederr redirected to BMKSUITENAME.log in subdir

HEPscore Application – Execution Example

```
$ cat ./test.yaml
hepscore_benchmark:
  benchmarks:
    atlas-kv-bmk:
      scorekey: wl-scores
      version: ci1.1
      ref_scores:
        sim: 1.0
  method: geometric_mean
  name: TestBMK
  reference_machine: Intel Core i5-4590 @ 3.30GHz - 1 Logical Core
  registry: gitlab-registry.cern.ch/hep-benchmarks/hep-workloads
  repetitions: 3
  version: 0.1
$ ./hepscore.py -svyf ./test.yaml /tmp/hepscore
Using custom configuration: ./test.yaml
TestBMK Benchmark
Version: 0.1
System: Linux fedora.localdomain 5.2.11-100.fc29.x86_64 #1 SMP Thu Aug 29 12:52:22 UTC 2019 x86_64
Container Execution: singularity
Registry: gitlab-registry.cern.ch/hep-benchmarks/hep-workloads
Output: /tmp/hepscore/HEPscore_30Sep2019_115348
Date: Mon Sep 30 11:53:48 2019

Executing 3 runs of atlas-kv-bmk
Running  ['singularity', 'run', '-B', '/tmp/hepscore/HEPscore_30Sep2019_115348:/results',
'docker://gitlab-registry.cern.ch/hep-benchmarks/hep-workloads/atlas-kv-bmk:ci1.1'] ...
 1.6129
 1.9531
 1.996
 Median: 1.9531
Final result: 1.9531

$ ls /tmp/hepscore/HEPscore_30Sep2019_115348
atlas-kv-c1-e100-1569858866_7638  atlas-kv-c1-e100-1569859231_8193  TestBMK.yaml
atlas-kv-c1-e100-1569859047_4810  TestBMK.log
```

HEPscore Application – Example YAML Output

```
$ cat /tmp/hepscore/HEPscore_30Sep2019_115348/TestBMK.yaml
hepscore_benchmark:
  benchmarks:
    atlas-kv-bmk:
      ref_scores:
        sim: 1.0
      run0:
        duration: 187.0
        end_at: Mon Sep 30 11:56:55 2019
        report:
          app:
            bmk_checksum: 03fc2656453a4033702712d41f07a3f7
            bmkdata_checksum: 199a9d5cb218c303eafd74076bb7a3c0
            cvmfs_checksum: 0de27e9c8ccb8d9bf5a191943a19ca2f
            description: 'ATLAS KV: GEANT4 simulation of 100 single muon events, based
              on Athena version v17.8.0.9'
            version: ci1.1
          copies: 1
          events_per_thread: 100
          threads_per_copy: 1
          wl-scores:
            sim: 1.6129
          wl-stats:
            avg: 1.6129
            max: 1.6129
            median: 1.6129
            min: 1.6129
            score: 1.6129
        start_at: Mon Sep 30 11:53:48 2019
      run1:
        ...
      run2:
        ...
      scorekey: wl-scores
      version: ci1.1
  environment:
    container_exec: singularity
    date: Mon Sep 30 11:53:48 2019
    system: 'Linux fedora.localdomain 5.2.11-100.fc29.x86_64 #1 SMP Thu Aug 29 12:52:22
      UTC 2019 x86_64'
  method: geometric_mean
  name: TestBMK
  reference_machine: Intel Core i5-4590 @ 3.30GHz - 1 Logical Core
  registry: gitlab-registry.cern.ch/hep-benchmarks/hep-workloads
  repetitions: 3
  score: 1.9531
  version: 0.1
  wl-scores:
    atlas-kv-bmk:
      sim: 1.9531
```

HEPscore Application – Example YAML Output (Cont.)

JSON output by default

- Commandline option available to alternatively generate more readable YAML output (“-y”)

Output contains the full HEPscore configuration data, and the JSON output of each individual benchmark container execution run

- Individual container run output:

  benchmarks → BENCHMARK_NAME → runX → report

Start/end time, and wall clock duration (seconds) for each run included

Some system information, such as container execution tool (docker or singularity) and kernel version included

Overall/final score reported under “score” key

HEPscore Application – Plans

Long-term Plans

- Enhance the modularity of hepscore.py
  - Allows for a cleaner integration of hepscore into the HEP Benchmark Suite and other tools
  - Simplifies development by others
  - Facilitates more advanced CI functionality

Ensure Python 3 compatibility

Consider switching from getopt() to argparse() for command line option processing

  - Could be used to eliminate the help() function

Potentially use Docker and Singularity Python APIs to instantiate the benchmark containers instead of Popen()?

  - Singularity Python API is a 3rd party package, and not available in EPEL

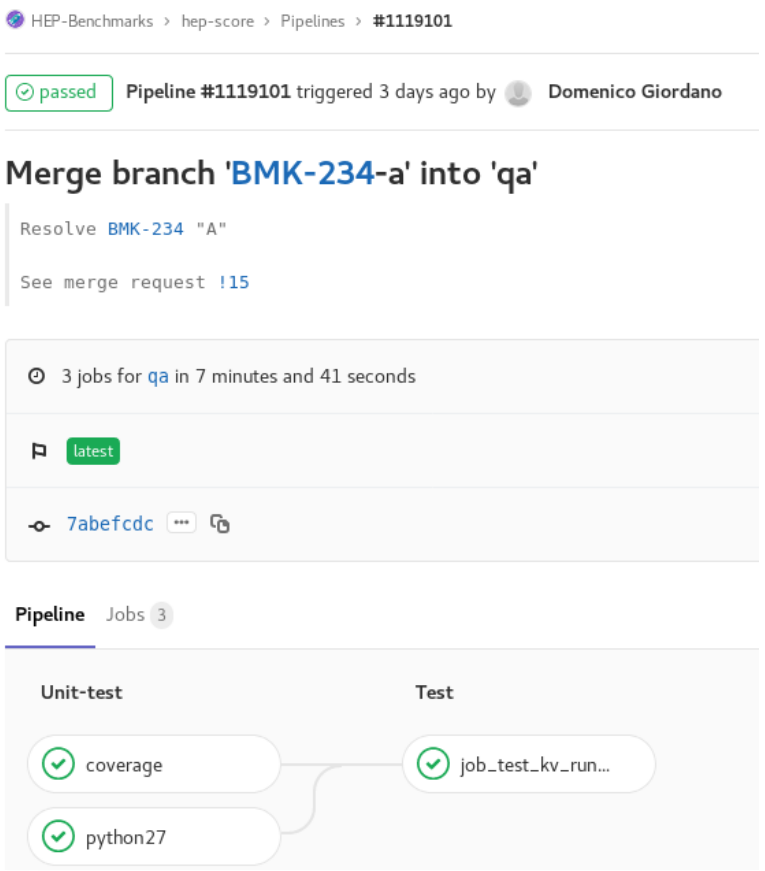
HEPscore Application – Continuous Integration

Utilizing Gitlab Continuous Integration

- Various tests implemented:

- Best practices
  - pep8 lint
  - bandit security checks

- Execution
  - Validation of statistical functions (implemented in the hepscore code to avoid dependency on “statistics” package)
  - Functionality of various command line options
  - Configuration dumping
  - Parsing and score generation from mock results
  - Execution of atlas-kv-bmk via hepscore.py succeeds



Example CI Pipeline

HEPscore19 Benchmark

A prototype “HEPscore19” benchmark is included as the default configuration in the HEPscore utility

- Presently it runs the following benchmark containers 3 times each (taking the median result) and reports a final score based on the geometric mean:

  atlas-gen-bmk v1.1
  atlas-sim-bmk v1.0
  cms-gen-sim-bmk v1.0
  cms-digi-bmk v1.0
  cms-reco-bmk v1.0
  lhcb-gen-sim-bmk v0.12

- The above benchmarks were chosen, as from previous studies, they have been shown to be robust and reproducible

- Reference scores in the configuration based on the performance of a single socket Intel Xeon E5-2640 v3 @ 2.60 GHz CPU

HEPscore19 Benchmark (Cont.)

Presently takes ~16+ hours to complete on modern hardware

- May be necessary to reduce number of events processed per benchmark, or reduce the number of benchmarks run

In my opinion, ideally we would include at least one benchmark for each LHC experiment

- Unfortunately, the alice-gen-sim-bmk container currently occasionally crashes, so it could not be included

Additional discussion, work and testing necessary to determine the final set of benchmarks to include in HEPscore19

Conclusions

A beta version of the HEPscore benchmark utility is available for testing
https://gitlab.cern.ch/hep-benchmarks/hep-score

- Most of the desired functionality has been implemented

The application includes a built-in configuration for an early prototype "HEPscore19" benchmark

- Being evaluated/tuned as potential alternative to HEPSPEC06

- Final set of benchmark containers to include under discussion

Some additional features/improvements will be made in the coming weeks