Programmatically run SPEC CPU2017

D. Giordano (CERN)

pre-GDB Benchmarking WG
12 June 2018
SPEC CPU 2017 and HEPiX Benchmarking WG

SPEC CPU2017 (*SC17*) is available since ~1 year

Plan to study:
- Features of the SC17 suite: benchmark mix, compiler flags, rate Vs speed
- Relationship w.r.t. HEP job mix

But so far only scattered results produced by our community
- Main results presented today

IMHO the main reason is the difficulty in:
- Defining the appropriate configuration(s)
- Performing reproducible studies: collecting, sharing and tracking results

In the next slides a possible solution to those obstacles is proposed
“Push the button” & run SC17

Ability to trigger SC17 runs and extract results using a single wrapper script call

- Pass arguments via command line
  - cern-benchmark --benchmarks=spec2017 --spec2017_path=<SC17 installation dir> --spec2017_iter=<num iterations> --spec2017_bmk=<some bset. Eg 511.povray_r>

- Install the software from tarball archive [optional]
  - cern-benchmark --benchmarks=spec2017 […] --spec2017_url=<url to SC17 tarball to install>

- Run in sequence with other benchmarks
  - cern-benchmark --benchmarks="hs06_32;whetstone;DB12;kv;hs06_64;spec2017" […]

- Collect results of each running benchmark in a json document
  - Simplify portability and usage in other applications, e.g.: monitoring and analytics
A wrapper script similar to HS06

The wrapper to trigger SC17 is very similar to the HS06 script (runspec.sh)

- Evaluate the score as the geometric mean of the independent runs
  - Same formula as HS06
- NB: standalone script, can be used independently of the full suite

Optional arguments include:

- Config. file
  - Adapted from the SPEC CPU template Example-gcc-linux-x86.cfg
  - NB: Currently sets OPTIMIZE = -g -O3 will be changed to -O3 -fPIC -pthread (see M. Alef talk)

- Set of benchmarks
  - Defined a new set (pure_rate_cpp.bset) including only pure C++/C benchmarks as in HS06

https://www.spec.org/cpu2017/Docs/index.html#benchmarks
How to install the benchmarking suite

On SLC6
- Install from benchmarking suite gitlab repository

On CC7
- Follow the SLC6 approach, OR
- Use **docker container** based on SLC6 image
  - Published in gitlab registry
    - `docker run --net=host -v /tmp:/tmp -v /var/HEPSPEC:/var/HEPSPEC
gitlab-registry.cern.ch/cloud-infrastructure/cloud-benchmark-suite/cloud-benchmark-suite-standalone:1.5 cern-benchmark [...]`

Upcoming
- New rpm, to be made available in the CERN SLC6/7 repos
- New documentation, describing all those features
At CERN this json doc is transferred to ElasticSearch via a message broker.
The last 3 months of data

SC17 score/core by physical node

Example of start time desynchronization between VMs on the same physical node (cleaned in the following analysis)
Analysis’ results

- Description of the benchmarking setup @ CERN

- Focus on two questions
  1. How much SC17 is different from HS06?
  2. Are all the individual benchmarks in the SC17 independent?
     i.e. Are all benchmarks needed to build a representative benchmark mix?

Amount of data collected

<table>
<thead>
<tr>
<th>cpuname</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
<td>1097.0</td>
</tr>
<tr>
<td>Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz</td>
<td>278.0</td>
</tr>
<tr>
<td>Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz</td>
<td>486.0</td>
</tr>
<tr>
<td>Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz</td>
<td>476.0</td>
</tr>
<tr>
<td>Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz</td>
<td>572.0</td>
</tr>
<tr>
<td>Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz</td>
<td>1005.0</td>
</tr>
<tr>
<td>Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz</td>
<td>995.0</td>
</tr>
</tbody>
</table>
Benchmarking approach

- Profile full physical node, SMT enabled
- Split node resources in VMs. No overcommit, NUMA aware flavors
- Benchmark suite run synchronously in all VMs
- Benchmarks sequence
  --benchmarks="hs06_32;whetstone;DB12;kv;hs06_64;spec2017"
- Run `a la HS06`
  - use N x SPECrate to mimic the HS06 multiple speed
- NB: running on VMs gives an overhead of few % respect to bare metal
  - Presented results are ratios, mainly cancel this effect

https://w3.hepix.org/benchmarking/how_to_run_hs06.html
Are SC17 and HS06\textsubscript{64bit} correlated?

Implications

- No correlation => weird situation
- High correlation => questioning the change

Reminder: using just \texttt{pure\_rate\_cpp}
Are SC17 and HS06\textsubscript{64bit} correlated?

- Error bars represent the percentiles [5%, 95%]
- Marker size proportional to the number of entries
- Correlation coefficient of the mean values: 0.950
- Linear fit
  \[ SC17 = m \times HS06 + q \]
  - \( m = 0.141 \), \( q = -0.280 \)
  - Residuals ratios: (extr./meas.-1) < 4%
Validation: HS06 32bits Vs 64bits

Study on HS06 reported in the past
- Discrepancy between HS06 @ 32bits and 64bits is ~10%

- Correlation coefficient of the mean values: 0.97
- Linear fit
  \[ \text{HS06}_{64} = m \times \text{HS06}_{32} + q \]
  - \( m = 1.18, q = -0.156 \)
  - Residuals ratios: \( \frac{\text{extr.}}{\text{meas.}} - 1 \) < 3%
Are the individual SC17 benchmarks independent?

Are all SC17 benchmarks (*pure_rate_cpp*) needed?

- Less benchmarks => Shorter runtime
  - Currently a run of *pure_rate_cpp* (8 benchmarks) takes <2.5 hour/iteration> in the 7 tested CPU models

- Better control of SC17 score Vs HEP job mix

- Studied if a subset of the *pure_rate_cpp* is as well representative of the performance score
Approach

- Build all $r$ combinations of the 8 benchmarks
  - With $r$ in $[1, 8]$, number of combination = $C(8, r)$

- For each combination re-evaluate the SC17 score
  - NB: The benchmarking suite stores the individual results of each benchmark and thread

- Aggregate the scores by CPU model, compute stats (weighted avg and std, min, max)

- Measure the residuals ratio respect to the original SC17 ($C(8, 8)$)
Computed scores

- New scores discrepancy ranges from <1\% to >50\%

- NB: Error bars represent the [min,max]

- Several subset with only 2 or 3 benchmarks are well in agreement with the original SC17 score

- Example: compatible with the original SC17 within 1 sigma
  - 508.namd_r, 510.parest_r, 531.deepsjeng_r
  - Discrepancy
    - max= 0.017, min= -0.035
    - mean= -0.001 ± 0.005

Subsets compatible with original SC17 within 3 sigmas
What about HS06?

- Similar behavior, but larger incompatibilities
- NB: Error bars represent the [min,max]

- Examples of reduced set compatible with the original HS06 within 1 sigma
  - 444.namd, 447.dealII, 471.omnetpp
    - Discrepancy
      - max= 0.056, min= -0.015
      - mean= -0.003 ± 0.007
  - 444.namd, 447.dealII, 473.astar
    - Discrepancy
      - max= 0.066, min= -0.02
      - mean= -0.004 ± 0.009
Summary

Convergence toward a common tool (and configuration) to run SPEC CPU2017 (SC17)
  – Now possible using a wrapper script in the style of HS06 runspec.sh
  – Additional features offered by the benchmarking suite
    • Documentation will be available in the next weeks

Correlation between SC17 and HS06
  – Very high correlation, measured on 7 different Intel CPU models
    • Covering >90% of the CERN Openstack cloud
  – Need to extend the study to other CPU models and architectures
  – If the very high correlation remains, why changing from HS06 to SC17?

The individual SC17 benchmarks (pure_rate_cpp) are not fully independent
  – It is possible to find subsets of benchmarks giving the same score ratio
  – How those set will scale with the HEP job mix? To be studied