Updates from the HEPiX Benchmarking Working Group

M. Alef (KIT), D. Giordano (CERN), M. Michelotto (INFN)

hepix-cpu-benchmark@hepix.org
https://twiki.cern.ch/twiki/bin/view/HEPIX/CpuBenchmark

HEPiX Spring 2017 Workshop
26 April 2017
Mandate of the Working Group

- Investigate **scaling issues** between HS06 and CPU intensive HEP workloads (i.e. EvtGen, Simulation)

- Study the **next generation of long-running benchmark** (successor of HS06)

- Evaluate **fast benchmarks**; identify their properties; provide recommendations to the community
  - To estimate the performance of a job slot (in traditional batch farms) or VM instances (in cloud environments)
Reference Twiki Page

- Track topics under study
  - Collect results
  - Document procedures to access dedicated resources and to run benchmarks
  - Includes link to previous HS06 Twiki page

Meeting topics under study:

- Collect results
- Document procedures to access dedicated resources and to run benchmarks
- Includes link to previous HS06 Twiki page

https://twiki.cern.ch/twiki/bin/view/HEPIX/CpuBenchmark
Checkpoints

- Two main checkpoints until now
  - Feb. pre-GDB and Apr. GDB

- Debates around the scenarios for fast benchmarks adoption
  😊 Forecasting job slot duration or checking performance of VM in cloud environments (large consensus)
  😞 Replacement of HS06 for site pledge and procurement
    - Large divergence of opinions
    - Limited instruction mix ➔ exposed to microarchitecture changes/optimization
    - Risk of missing all implications of that choice on the medium-long term

- Major effort to study in detail the fast benchmarks
Fast Benchmarks

Started with 5 candidates

- ATLAS KV (KitValidation)
  - Mainly GEANT4. Default workload: 100 single muon event simulation
- DIRAC Benchmark 2012 (DB12), a.k.a. FastBmk, LHCbMarks
  - Python script: random.normalvariate() [see next slides]
- ROOT Stress test
- Legacy benchmarks: Whetstone, Dhrystone

- Systematic studies converge towards DB12 and Atlas KV
  - In particular DB12 seems to reproduce the job performance in Intel Haswell CPUs (a.k.a. “magic boost”)
KV performance Vs ATLAS Sim job

Running benchmarks and jobs in Commercial Clouds (VM)
ATLAS Sim jobs Vs KV and DB12 Vs KV:
  – At first order good linearity proven across different VM (and CPU) models
DB12 Vs jobs (LHCb and ALICE)

- **LHCb**
  - HS06 not a good predictor for MC
    - In particular for Intel Haswell CPUs
    - Not as bad for LHCb reconstruction jobs though
  - DB12 is much better

- **ALICE**
  - Very good correlation of DB12 Vs MC
  - Large discrepancy respect to HS06

- **NB:** the average load of the server (i.e. number of running job slots) plays a major role in those evaluations (see next slides)
CMS Tests

Preliminary comparison of DB12, KV, HS06 Vs CMS jobs

- Several dedicated nodes targeted for benchmarking
  - Sandy Bridge, Ivy Bridge, Haswell, AMD, ARM
  - ttbar simulation

- Ongoing activity
  - Pilots instrumented to run fast benchmarks and collect results Vs any kind of job
  - CPU model reported in the job dashboards
    - This will enable also the passive benchmark approach (see next slides)
HammerCloud reference jobs (single and multi core) running on benchmarked resources at GridKa

- Further investigation is needed
  - In particular for multi core jobs
Scaling factors across VM configurations

• Benchmarking of Haswell servers (E5-2630 v3 CPUs) in virtual environment

• Partition the available compute resources in a number of VMs of same size
  – => Fully load the servers as done for HS06

• Study the ratio of performance among different configurations
  – VM sizes and SMT enabled/disabled

• The +20% gain in performance seen in HS06 with SMT=ON is not reproduced by DB12 and KV
  – Bigger discrepancy from DB12
DB12 studies

Seen the multiple and somehow diverging results on DB12 the working group decided to invest effort in additional studies

– Application profiling
– Reproducibility under different Python versions
– Effect of different implementations: C++ or using Numpy
DB12 (python) Features

Main components of the python script
- Multi – Processes spawn by `multiprocessing` module
- Per process:
  - Loop around random number generation
    - `random.normalvariate(10,1)` from `random` module
  - Eval CPU time (cput)
  - Get DB12 score as calib/cput
- Compute average, median, sum of DB12 individual scores

Assumption: DB12 well represents MC jobs because mainly dominated by random number generation, as are MC jobs
- Is this assumption correct? (see next slides)

```
# Set up all the subprocesses
for i in range( copies ):
    results.append( multiprocessing.Value('d', 0.0 )
processes.append( multiprocessing.Process( target = singleDiracBen
# Start them all off at the same time
for p in processes:
P.p.start()

# Possibly do one extra iteration to avoid tail effects when copies run in batches
for i in range( iterations + 1 + (1 if extraIteration else 0)):
    if i == 1:
        start = os.times()
    # Now the iterations
    for j in xrange( n ):
        t = random.normalvariate( 10, 1 )
        m += t
        m2 += t * t
        p += t
        p2 += t * t
    if i == iterations:
        end = os.times()
    cput = sum( end[:4] ) - sum( start[:4] )

# Return DIRAC-compatible values
return ( 'CPU' : cput, 'WALL' : wall, 'NORM' : calib * iterations / cput, 'UNIT' : 'DB12' )
```
The Branch Prediction due to CPython Module

```
• perf studies from M. Guerri have shown that the main contributor to DB12 is `PyEval_EvalFrameEx`
• Starting from Haswell models, this module benefits from a better branch prediction that boosts the DB12 performance
```

[see L. Atzori’s talk]
Other Implementations of DB12

- DB12 Optimised Python version with Numpy (V. Innocente): x10 faster than standard DB12
- DB12 C++ implementation (D. Giordano): x10 faster than standard DB12
  - Multi-processing using fork, pipe, same normalvariate algorithm

- Comparison of the perf profile for the three implementations shows that, contrary to the standard DB12 code, the Numpy and C++ versions are dominated by calls to math and rand libs

![Table showing performance comparison]

- Documented @ http://cern.ch/go/Pq9T
DB12 Vs OS (and python) Versions

- DB12 scores are affected by changing python version
  - Variation of 10%-18%
- Ratio $DB12(32 \text{ cores}) / DB12 (16 \text{ cores}) = \sim 1$
  $\Rightarrow$ no gain in SMT=ON

- As reference:
  - the scores of the C++ version are less affected by the different OS (~5%)
  - Ratio $DB12(32 \text{ cores}) / DB12 (16 \text{ cores}) = \sim 1.5$
  $\Rightarrow$ 50% gain with SMT=ON

- NB: This is not a suggestion to migrate DB12 to a C++ version, but just an example to highlight potential issues and discrepancies among implementations

- Documented @ [http://cern.ch/go/Pq9T](http://cern.ch/go/Pq9T)
Benchmark Comparison on Grid Nodes

Comparison on different HW models (M. Alef)

- Better scaling of DB12-cpp and DB12-np with HS06 than initial DB12 Python script

- The +45% boost appears only when running DB12 on 

# of slots <= # physical cores

and goes down when SMT is enabled

- Another effect of the lack of gain of DB12 with SMT enabled

- NB: In SB also DB12 benefits of the 20% gain with SMT=ON
Dissecting Benchmarks (I)

Open Question

DB12 does not profit from SMT on Haswell/Broadwell architecture. KV apparently does the same. What is the reason behind this slowdown? Is it the same for both benchmarks? e.g. (hypothesis, not validated yet):

- DB12 is heavy on branches/jump. On Haswell, there are two execution ports that can execute branch instruction (compared to only one on IvyBridge. These are not ports exclusively for branch instructions). Validation of speculative fetch happen very fast keeping the pipeline always very busy.
- KV heavily profits from iTLB. When running two hardware threads, iTLB entries are thread specific. High number of conflicts.

ATLAS KV - Functions profile

![Functions profile chart]

1. Functions profile on core 0 (single event)

2. Functions profile on core 8 (single event)

Unexpectedly large number of cycles spent at the beginning of each function (e.g. mov of literal value to register taking up 8%)

Hint of instruction cache misses?

NB: old Athena version 17.8.0.9 used in KV benchmark
Conclusions 1

- Simulation (Geant4) (and python?) shows a high degree of code non-locality
  - Any different behavior in instruction pre-fetching will affect it much more than other benchmarks
- HS06 is memory greedy (at least compared to cms simulation& reconstruction)
  - Multiple instances running against common resources will scale worse than CMS Sim&Reco
  - It surely depends on the details of the cache hierarchy (arm, atom, knl, skylake, amd)
Passive Benchmarks of ATLAS Tier-0 CPUs

- Use real jobs to measure the relative speeds of different CPU models
  - Description of the analysis in the pre-GDB talk
- The analysis was applied to jobs run at the ATLAS Tier-0
  - Mainly reco jobs
  - Scaling generally good, two exceptions
    - Opteron way off
    - Haswell tends to perform better than what HS06 predicts: +10% with both SMT ON and OFF

[see A. Sciabà’s talk, tomorrow]
Plan of Studies

• Start drafting requirements for the validation of the successor of HS06
  – Whatever it will be: HS17, HEP suite based on experiments’ workloads, etc.
  – Aim: Scale, within a given accuracy, with a representative WLCG job mix

• Need to disentangle several effects
  – Bare-metal server Vs VMs, HT ON/OFF, different OS, -m32 Vs -m64, job variability, load on neighbor slots,…

• Identify testbed with representative set of hardware models
  • Include Non-default hardware models (e.g. ARM) of interest in HEP

• Perform reproducible studies
  – Document procedures
  – Experiments: identify and share representative job types (e.g. via CVMFS, containers, etc..)
  – Have access to monitoring data of production jobs
Summary of Intermediate Results

- DB12 (python version) has been (and still is) deeply studied
  - Scale well with ALICE and LHCb MC applications
  - BUT:
    - Runtime is dominated by libpython calls. Nothing to do with random number generation!
    - DB12 shows dependency from Python version, and it doesn’t benefit from SMT enabled
    - +40% boost (respect to HS06) from Intel Sandy Bridge to Haswell only for ½ loaded servers (SMT enabled)
      - Discrepancy with HS06 goes down when processes running are x2 physical cores
      - The initial boost is due to a better branch prediction in the Haswell CPU frontend
    - ATLAS and CMS are instrumenting pilots to run, collect and study DB12 scores Vs production jobs (including multi-thread jobs)

- HS06
  - Preliminary study still shows good agreement among HS06 and CMS MC ttbar, when server fully loaded
  - Passive benchmarking
    - Discrepancies among HS06 and ATLAS reco jobs are within 10% 

- Work in progress to setup a testbed for the HS06 successor