Karlsruhe Institute of Technology

GridKa

Towards HS17 (Iteration #1)

Manfred Alef

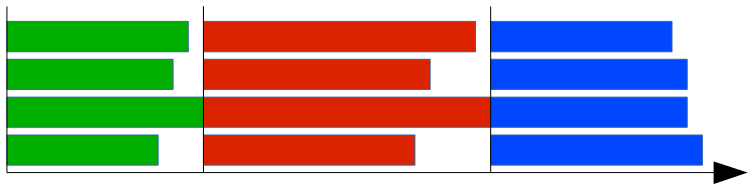
STEINBUCH CENTRE FOR COMPUTING (SCC)

Preamble

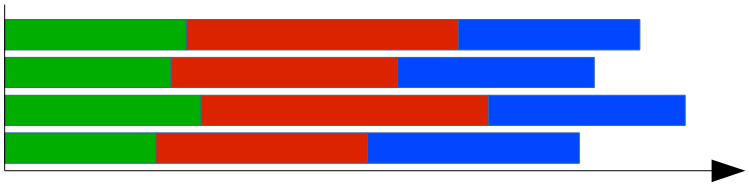
- This is "iteration #1" and not presenting final results
- Benchmark scores, as well as job efficiencies reported by users, have been measured until January 2018 – before the endless series of Meltdown/Spectre security patchings

Running SPEC CPU2017

- Benchmark metrics:
  - ➔ CPU2017 suite comes with 2 metrics (like CPU2006):
    - SPEED (single benchmark run, OpenMP supported)
    - RATE (multiple benchmark copies running in parallel)
  - ➔ HS06: running multiple SPEED benchmarks in parallel (better simulation of batch system than pure RATE)



4 synchronized benchmark copies when running RATE benchmark



4 independent benchmark copies of HS06 (like LRMS!)

Running SPEC CPU2017

- Benchmark metrics:
  - ➔ CPU2017 SPEED runs require too much memory per run (up to 16 GB because of bigger workloads), can‘t run multiple copies
  - ➔ CPU2017 RATE run happy with 2 GB per copy, can run multiple copies in parallel
    - Memory default at WLCG sites: 2 GB per job slot
  - ➔ Number of RATE benchmark copies = number of job slots
    - On Broadwell (2x10-core) systems: 20, 32, or 40 job slots (1.0, 1.6, or 2.0 slots per physical core)
  - ➔ Not mimicking the parallel run model of HS06 so far

Running SPEC CPU2017

- Building the benchmark:
  - Operating system: SL 6 (GridKa batch farm)
  - Compiler: gcc-4.8.5 (SL7 / CentOS 7 default)
    - One of the included benchmarks fails with gcc-4.4.x (SL6 default compiler)
  - Optimization: `-O0 -fPIC -pthread`
    - (No final decisions so far!)
  - Hardware model: `-m64`
  - Iterations: 5

Running SPEC CPU2017

- Running the benchmark:
  - ➔ Running all 10+13 RATE benchmarks of SPEC CPU2016
    - Configuration item:
      `runlist = intrate fprate`
    - Desired bset scores can be calculated from the individual benchmark results of that bset (geometric mean)
      - Benchmark sets used in this first assessments:
        - cpp = fprate_mixed_cpp.bset + intrate_any_cpp.bset (7 benchmarks: 507, 511, 520, 523, 526, 531, and 541)
        - int = intrate.bset (10 benchmarks)
        - fp = fprate.bset (13 benchmarks)
        - 520 = 520.omnetpp_r (example of single benchmark)

Test Setup

- Systems under test (only dual-socket servers):
  - Intel Xeon:
    - E5-2665 (8-core, Sandy Bridge)
      - 16 job slots (1 slot per physical core)
      - 3 GB memory per job slot
    - E5-2630v4 (10-core, Broadwell)
      - 20, 32, or 40 job slots (1, 1.6, or 2 slots per physical core)
      - 3.2 GB, 3 GB, or 2.4 GB memory per job slot
  - AMD Opteron:
    - 6168 (12-core)
      - 24 job slots
      - 3 GB memory per slot

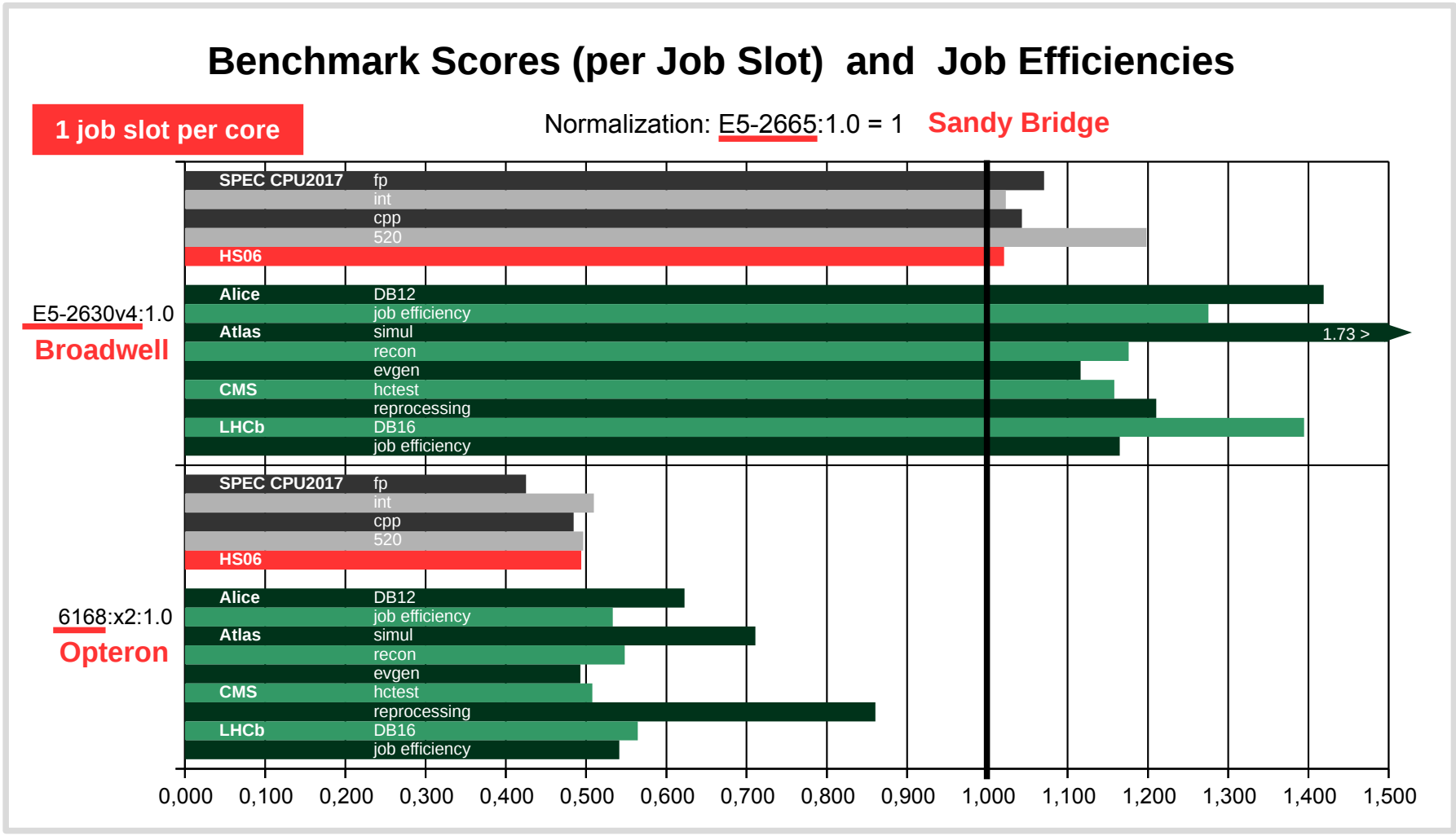
Test Setup

- Notations:
  - ➔ System under test:
    hardware_model:slots_per_core
    - Example 'E5-2665:1.5':
      Hardware model: Intel Xeon E5-2665 (Sandy Bridge 8-core)
      Job slots: 24 (1.5 per physical core)
- Almost homogeneous job distribution to WNs, except few nodes with less than 3 GB memory per slot

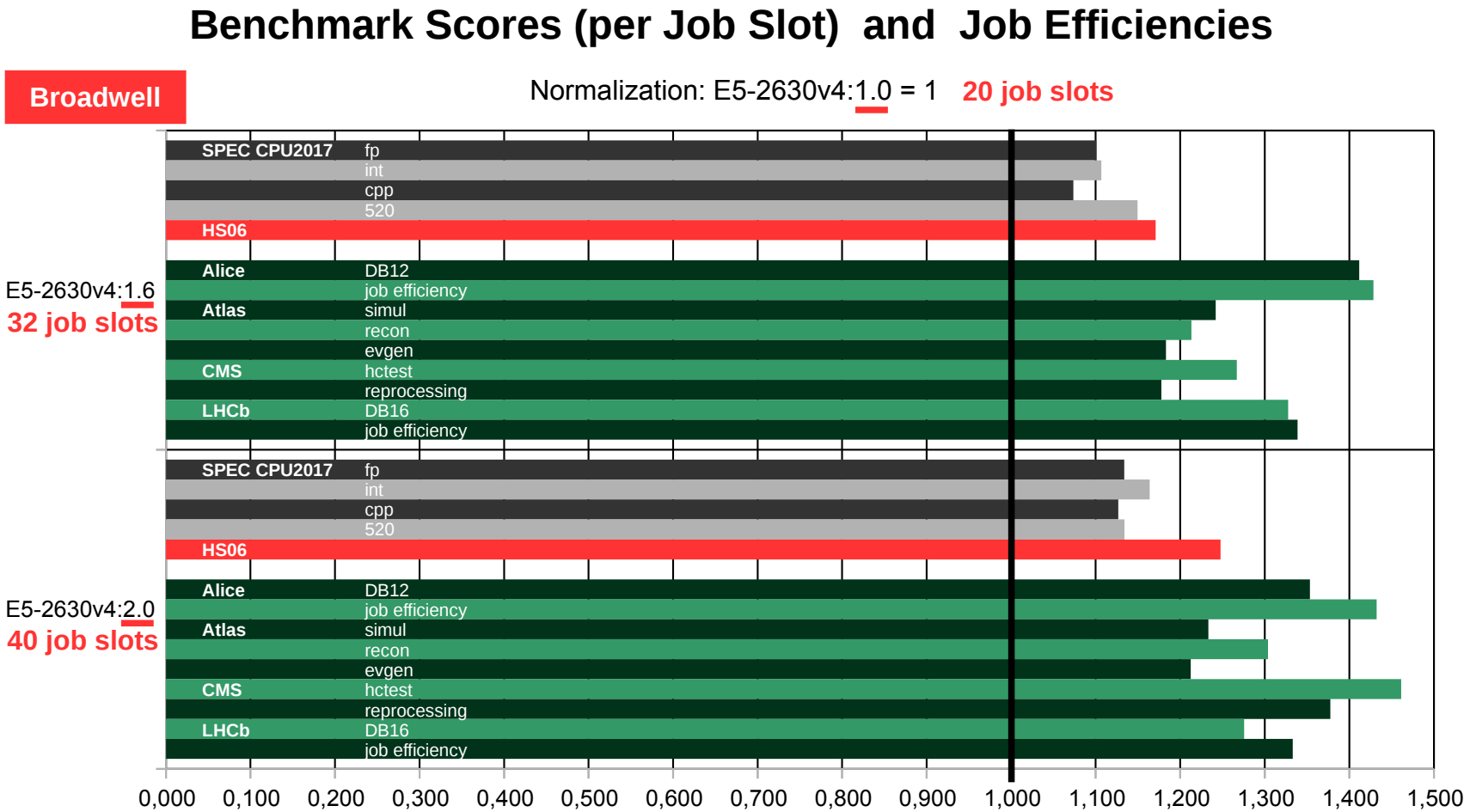
Scaling with HEP Applications

- Comparing benchmark results with representative performance scores (in units of events/s) of grid jobs run at GridKa
  - → Many thanks to Philippe Charpentier (LHCb), Costin Grigoras (Alice), Manuel Giffels and Valentin Kuznetsow (CMS) for providing measured values
  - → Atlas: performance statistics from Bigpanda, TaskIDs:
    - 10944000 (simul), 11323845 (recon), 11330855 (evgen)

Scaling with HEP Applications



Scaling with HEP Applications



Summary and Outlook

- First assessments started, no final results so far
  - ➔ Comparing benchmark results and job efficiencies across relevant hardware models, and different batch configurations of WNs
  - ➔ HEP reference workloads to be included
- Imbalanced scaling of benchmark scores as well as HEP applications
  - ➔ Final benchmark must correlate with average job mix running at WLCG sites (pledges!), but can never scale perfectly with each and every application
- Challenge to find a new benchmark (or set of benchmarks) scaling with typical mix of applications within a reasonable bandwith

Summary and Outlook

- Stable system performance is essential for benchmark development – be aware of artefacts caused by Meltdown/Spectre/Spectre-NG patch levels!