Karlsruhe Institute of Technology

Towards HS17 (Iteration #0)

Manfred Alef (KIT)

STEINBUCH CENTRE FOR COMPUTING (SCC)

Preamble

- This is "iteration #0" and not a final talk
- Benchmark results, as well as job efficiencies reported by users, have been measured in 2017 – before the Meltdown/Spectre security patchings

Running SPEC CPU2017

- Building the benchmark:
  - Operating system: SL 6 (GridKa batch farm)
  - Compiler: gcc-4.8.5 (SL7 / CentOS 7 default)
  - Optimization: `-O0 -fPIC -pthread`
  - Hardware model: `-m64`
  - Iterations: 5

Running SPEC CPU2017

- Running the benchmark:
  - ➔ Only RATE metrics, no SPEED benchmarks
    - Memory restrictions at WLCG sites: 2 GB per job slot
  - ➔ Not mimicking the parallel run model of HS06

Running SPEC CPU2017

- Running the benchmark:
  - ➔ All 10+13 RATE benchmarks
    - `runlist = intrate fprate`
    - Desired bset scores can be calculated from the included individual benchmark results (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 (single benchmark)

Running SPEC CPU2017

- Systems under Test (dual-socket servers):
  - ➔ Intel Xeon:
    - E5-2665 (8-core, Sandy Bridge)
      - 16, 24, or 32 job slots
    - E5-2630v4 (10-core, Broadwell)
      - 20, 32, or 40 job slots
  - ➔ AMD Opteron:
    - 6168 (12-core)
      - 24 job slots

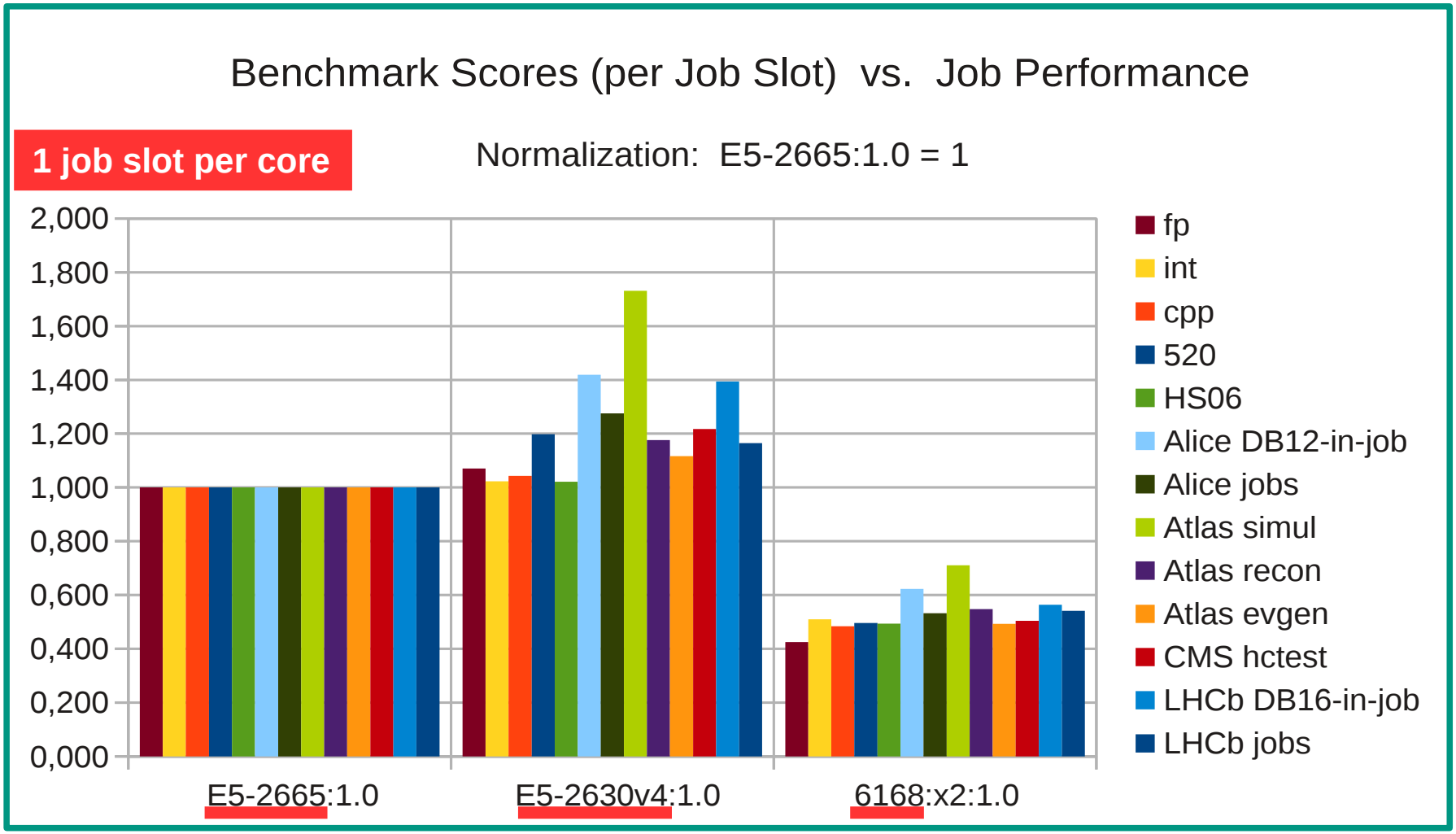
Running SPEC CPU2017

- Running the benchmark:
  - ➔ 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)

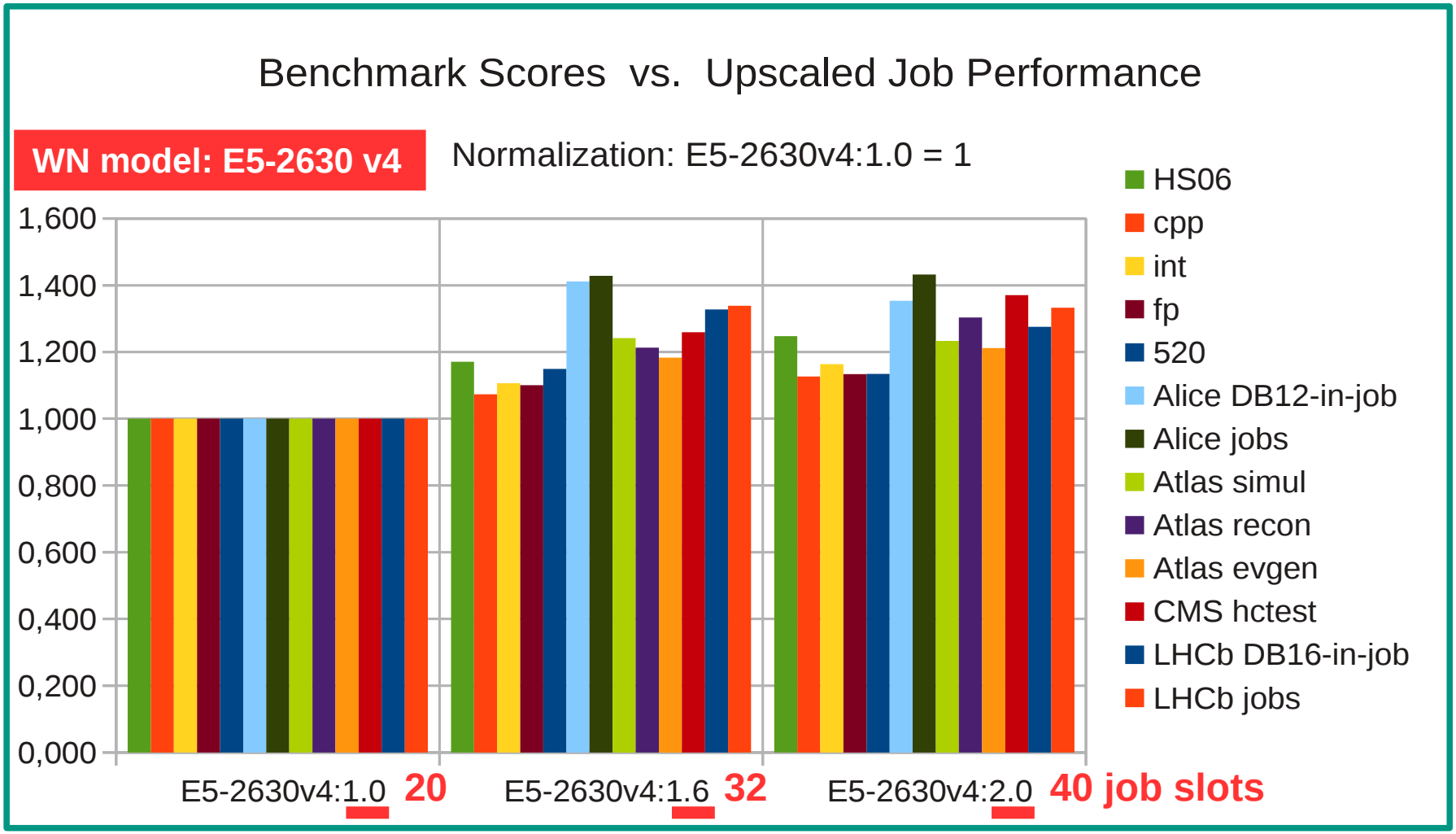
Scaling with HEP Applications

- Comparing benchmark results with performance scores (in units of events/s) of HEP application reported by users
  - → Many thanks to Philippe Charpentier (LHCb), Costin Grigoras (Alice), 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

- These are still very preliminary results, not a final solution
- Cross-checks by other sites will be welcome
- Results of reference workloads?

Appendix – How to Run the Analysis

- Example: CMS
  - ➔ I have got a CSV file (thanks to Valentin Kuznetsov/CMS) of jobs run at GridKa (i.e. WNHostName contains domain '.gridka.de')
    - Relevant rows:
      TaskMonitorId, WNHostName, Type, NEvProc, WrapWC
  - ➔ Grep the most relevant TaskMonitorId(s)
    - → 'hctest' jobs (around 2/3 of all included jobs)
    - Other tasks (analysis) with too few jobs each to plot lines
      - No outliers found, though
  - ➔ For each WN model calculate
    Average_Efficiency = sum(NEvProc)/sum(WrapWC)

Appendix – How to Run the Analysis

- Example: CMS
  - ➔ Results for Intel Xeon E5-2630v4 (Broadwell):

WN Model	Slots	Average_Efficiency	Upscaled	Normalized
E5-2630v4:1.0	20	3,96	79,2	**1,00**
E5-2630v4:1.6	32	3,12	99,8	**1,26**
E5-2630v4:2.0	40	2,72	108,8	**1,37**

    - Upscaled values: Average_Efficiency * Slots
    - Normalizing: dividing by the Upscaled value of E5-2630v4:1.0

  - ➔ Remark: HS06 scores of all WN hosts at GridKa are available from MJF (see `$JOBFEATURES/hs06_job`), so experiments can consider to run their own analysis