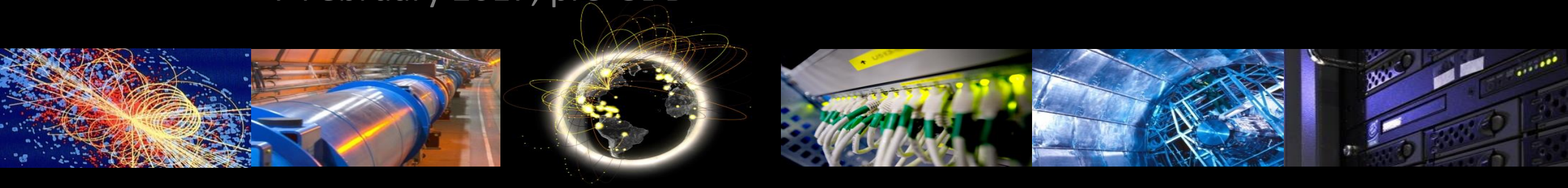
CPU benchmarking with production jobs

Andrea Sciabà
Andrea Valassi

7 February 2017, pre-GDB

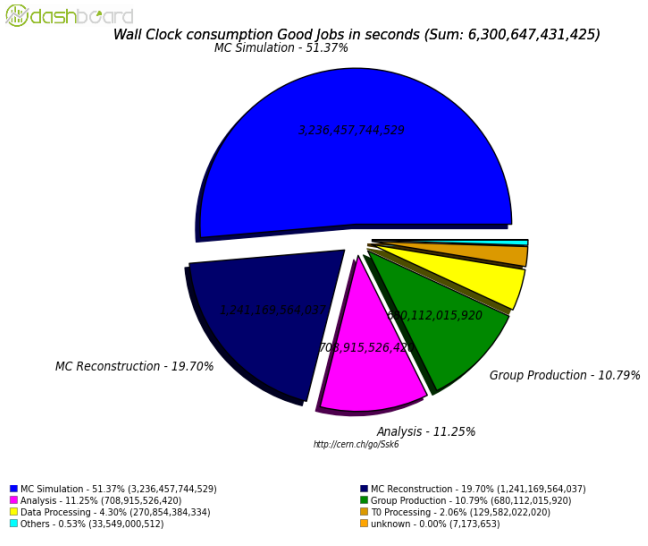


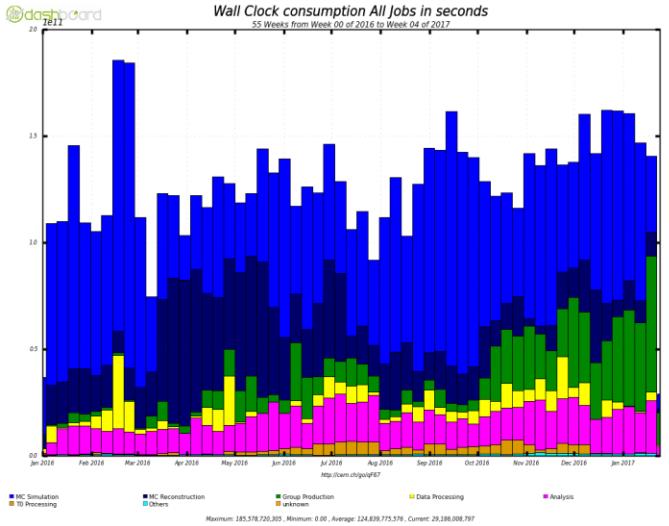
Initial motivation

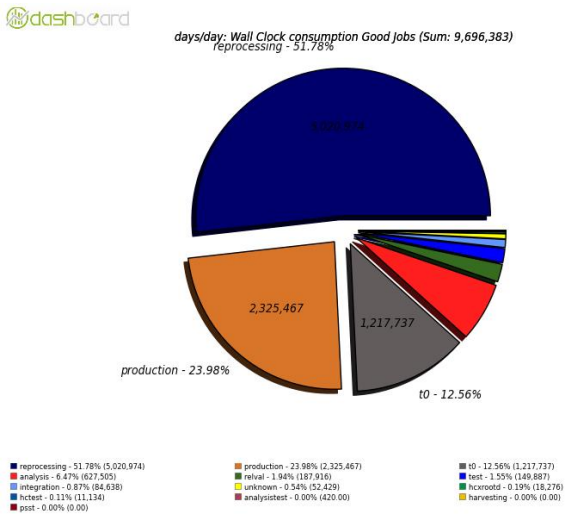
- Understand how experiments use their CPU resources
  - What types of jobs are (primarily) run?
  - How many resources do they require?
  - Are they “efficient” (i.e. do they waste wall-clock time)?
- Understand the behavior of the infrastructure
  - Can we measure the “speed” of CPUs, or sites, by looking at different types of jobs? Are the results compatible? Can we validate commonly used benchmarks using “real” jobs?

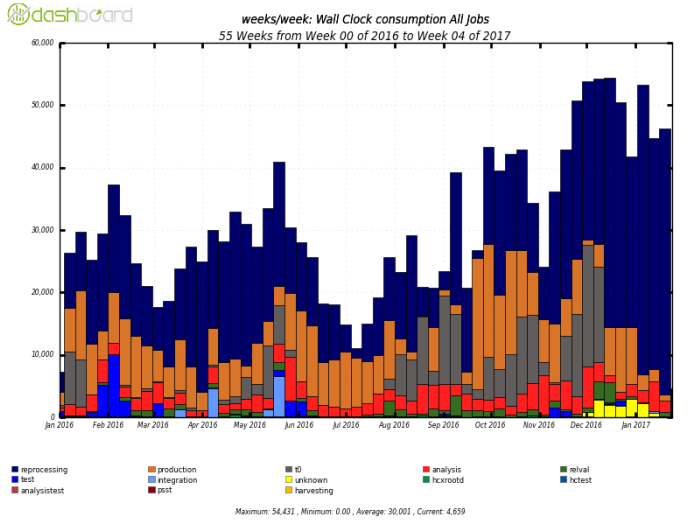
ATLAS and CMS wallclock time consumption

- ATLAS
  - MC simulation: 51%
    - Event gen. + simul.
  - MC reconstruction: 20%
  - Analysis: 11%
  - Group production: 11%
  - Reprocessing: 4%
- CMS
  - Reprocessing: 52%
    - DIGI-RECO of MC, re-RECO of RAW
  - Production: 24%
    - Event gen. + simul.
  - T0: 13%
  - Analysis: 6%



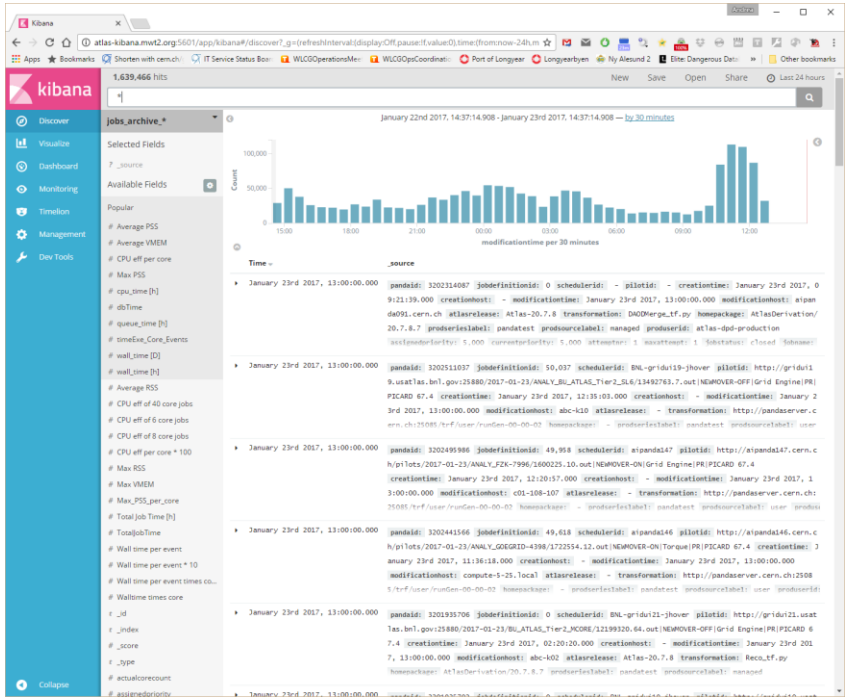


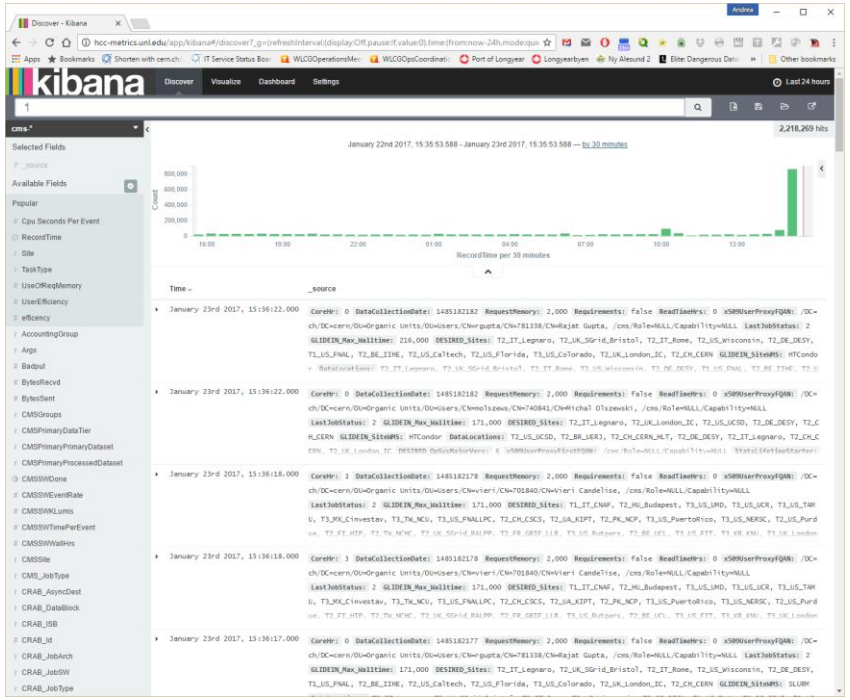




Data sources

- ATLAS and CMS independently developed machinery to store data about jobs in ElasticSearch databases
  - Kibana available for data exploration and fast prototyping of analyses
  - Very easy to make complex queries and data aggregations





Fitting CPU speeds

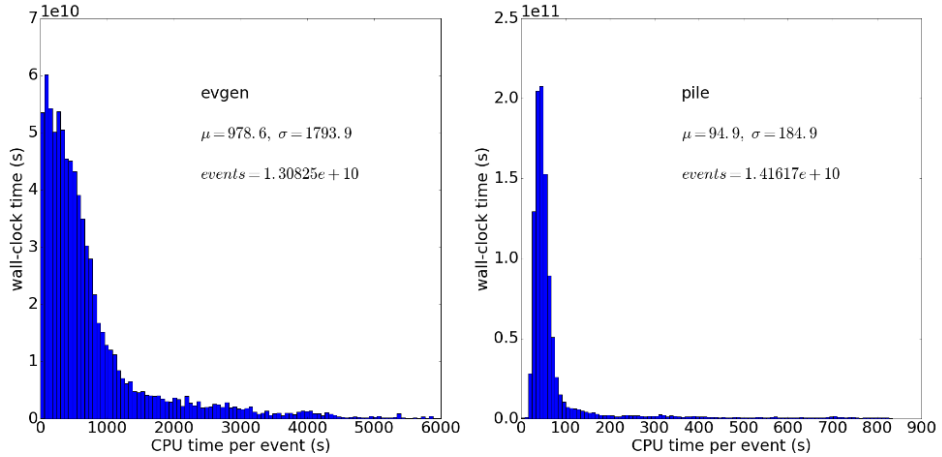
- Goal: explore the possibility to measure the “speed” of CPUs (or the average “speed” of a site) using CPU time consumption of real production jobs
- Basic assumptions
  - The CPU “speed” and the average CPU time/event of a given job are inversely proportional, i.e. running the same job on a 2x faster CPU will take ½ of the CPU time
  - All jobs in the same task are comparable, i.e. even if they run on different events they will need approximately the same amount of CPU time/event on the same node
- Limitations
  - Limited information about the worker node: the CPU model is known, but the SMT status is not, as the nature (virtual vs. physical) of the machine or any overprovisioning, etc.
    - WNs at different sites having the same CPU model will be treated as identical, so a systematic effect is unavoidable
  - There is no “absolute scale” for the CPU speed, unless a very specific application is used as a benchmark and defined as reference
    - Still, ratios of speed of different CPU models can be measured

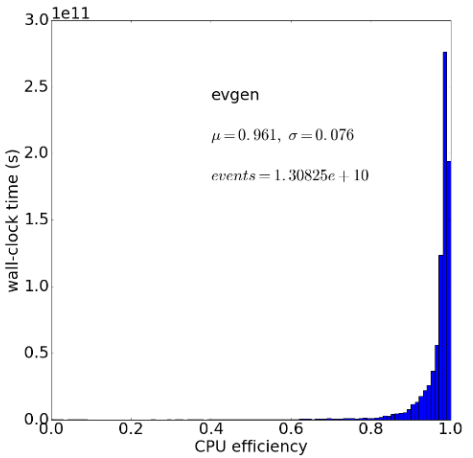
Data aggregation (ATLAS example)

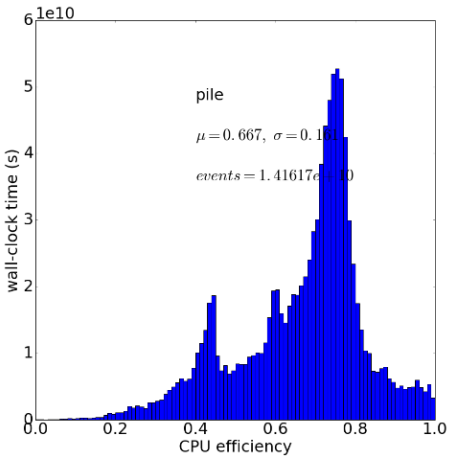
- ElasticSearch contains a record per job
  - ~ 0.5 Gjobs / year, way too much for an “online” analysis
- Data is aggregated and the results dumped to CSV files for “offline” analysis
- Aggregation variables:
  - JEDI task ID (all jobs in the same task are “similar”)
  - Site
  - CPU model
  - Job type (evgen, simul, etc.) & “transformation” (a wrapper around Athena specific to the type of processing)
- Aggregated metrics:
  - CPU and wall-clock time: total, average per job, average per event, etc.
  - Used cores
  - Total number of jobs and events

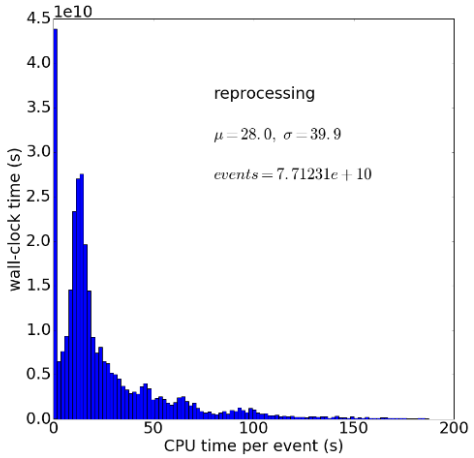
CPU per event and CPU efficiency (ATLAS)

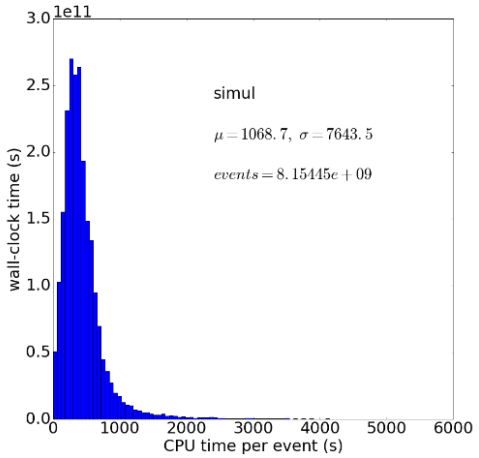
- Different job types have very different CPU efficiency ratios
  - CPU time is expected to be an equally good estimator in all cases

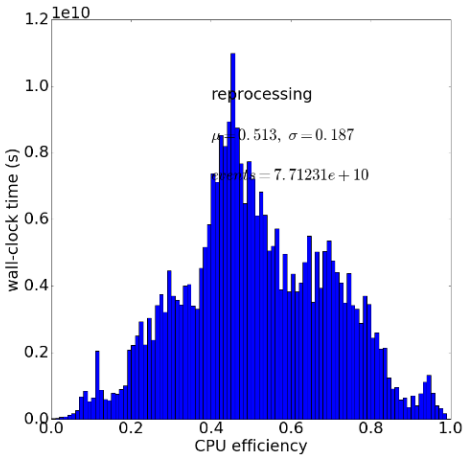


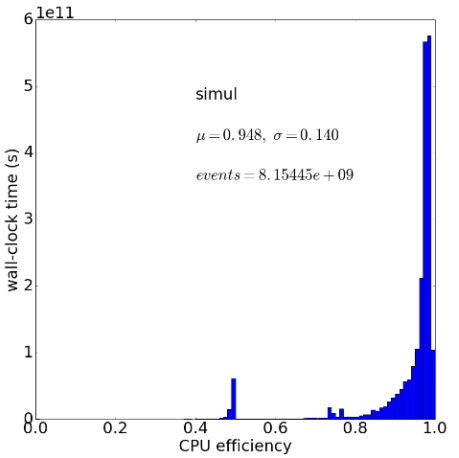












Analysis

- Population average of CPU time/event for task α on CPU *i*: $\mu_{i\alpha}$
  - ($a_{i\alpha}$: measured value, normal-distributed around $\mu_{i\alpha}$ with s.d. $\sigma_{i\alpha}$)
- Assumptions:
  - $\mu_{i\alpha} k_i = A_\alpha$ (k_i: speed factor for CPU model *i*)
  - $\sigma_{i\alpha} = S_\alpha \mu_{i\alpha}$
    - Used also as error on $a_{i\alpha}$ ("ignoring" a $1/\sqrt{N_{i\alpha}}$ factor)
  - $S_\alpha = S$
    - Independent on task *α*
- We will fit all A_α and k_i by minimising a χ^2

$$\chi^2 = \sum_{i\alpha} \left(\frac{a_{i\alpha} - \mu_{i\alpha}}{\sigma_{i\alpha}} \right)^2 = \cdots = \frac{1}{S^2} \sum_\alpha \frac{1}{A_\alpha^2} \sum_i (a_{i\alpha} k_i - A_\alpha)^2$$

- Gradient and Hessian can be calculated analytically
  - Crucial for the minimization to be feasible in a reasonable amount of time!

Fitting procedure

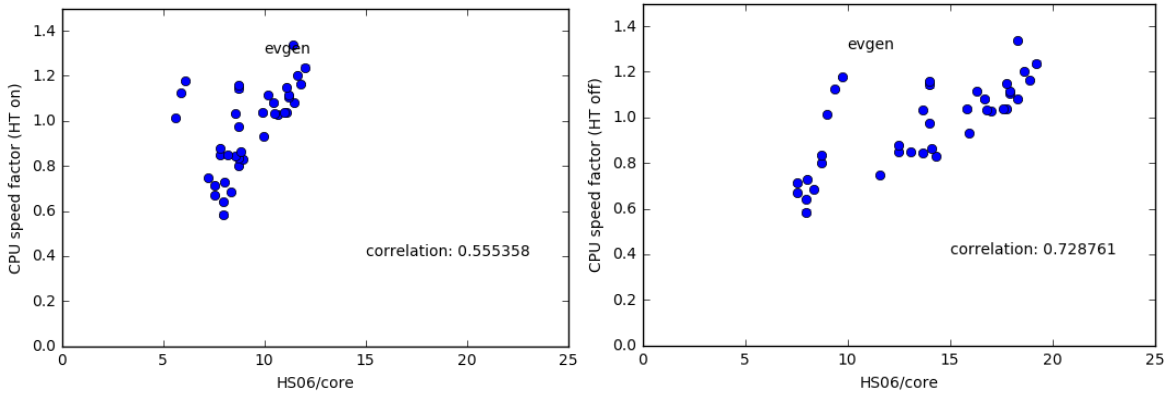
- Data sample divided (randomly) in 10 parts, fit performed separately on each
  - Initial values: $k_i = 1, A_\alpha = (\sum_i a_{i\alpha})/N_\alpha$
  - For each part, O(100) k_i and O(1000) A_α
- The fitted parameters are not fully constrained
  - Multiplying all k_i and A_α by the same number does not change the χ^2
- The "right" method of combining the fits is not obvious
  - Rescale k_i so that the vector has l_1 norm equal to the number of factors
  - Rescale k_i so that $k_0 = 1$
  - Rescale k_i so that the distance to (1, …, 1) is minimal
  - …
  - All methods produce similar results and strongly reduce the spread of each k_i measurements across the 10 samples
  - After rescaling, the 10 scale factor vectors are averaged

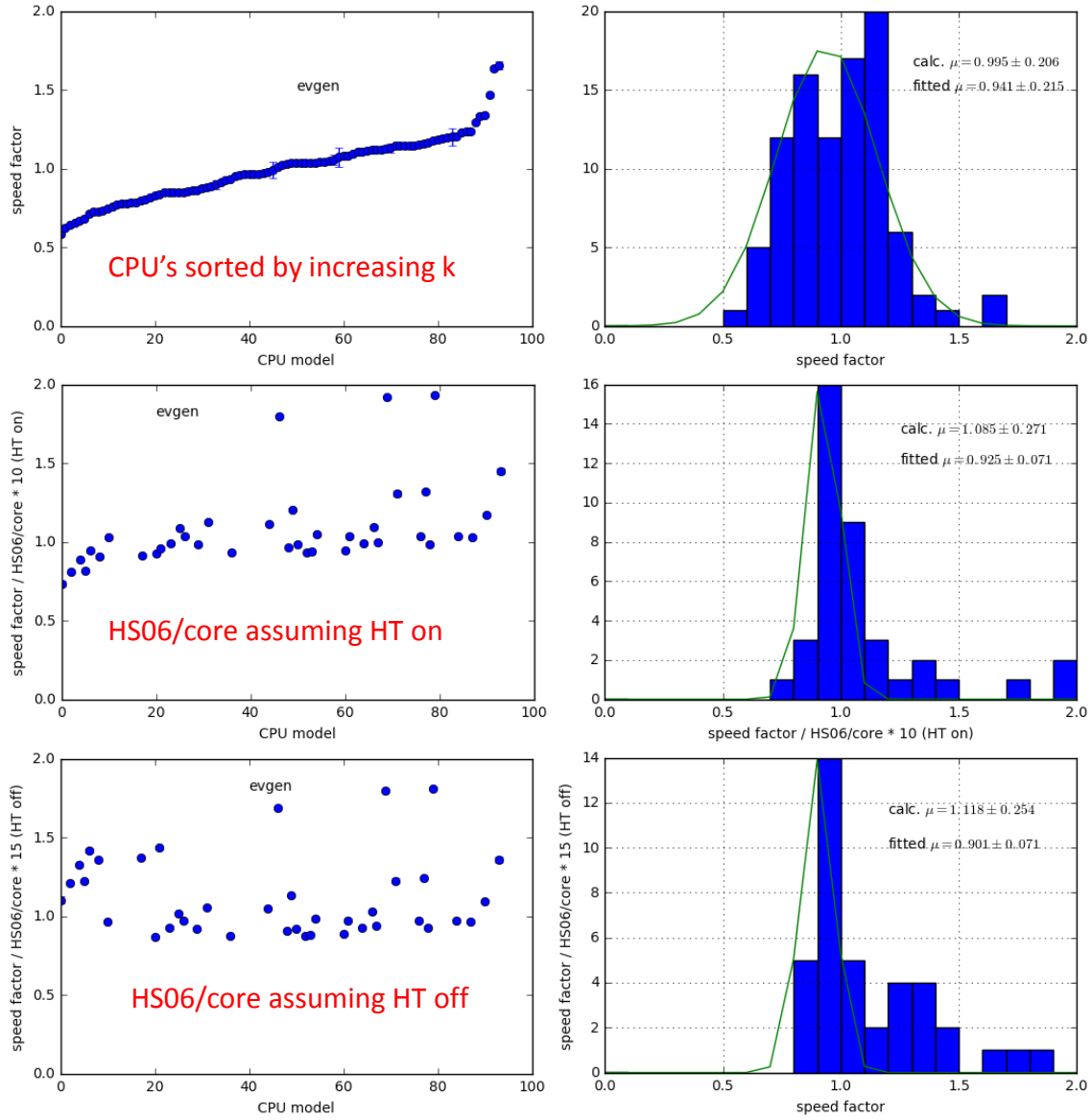
Comparing to HS06

- Used values published on HEPiX benchmarking WG page
  - http://w3.hepix.org/benchmarks/doku.php?id=bench:results_sl6_x86_64_gcc_445
  - Only 32 CPU models of the ~90 found in production jobs have a score
- HS06 score is obtained by running on all available cores
  - Usually from two physical CPUs
- Hyperthreading is sometimes enabled, sometimes not
  - HT on increases total score by ~25%
- HS06/core (core=hardware thread) heavily depends on HT
  - HT off: HS06/core ~1.6 larger than for HT on
    - Half threads, ~1.25 times HS06: 2 / 1.25 = 1.6
  - Impossible to know from job data if the WN had HT on or off
    - Large systematic uncertainty

Effect of HS06 rescaling

- Rescaling by HS06/core reduces the spread of the speed factor distribution
  - But the shape is not Gaussian due to outliers
  - Scenarios assuming HT off or on
  - Several systematic uncertainties
- Correlation value favors the scenario with HT off

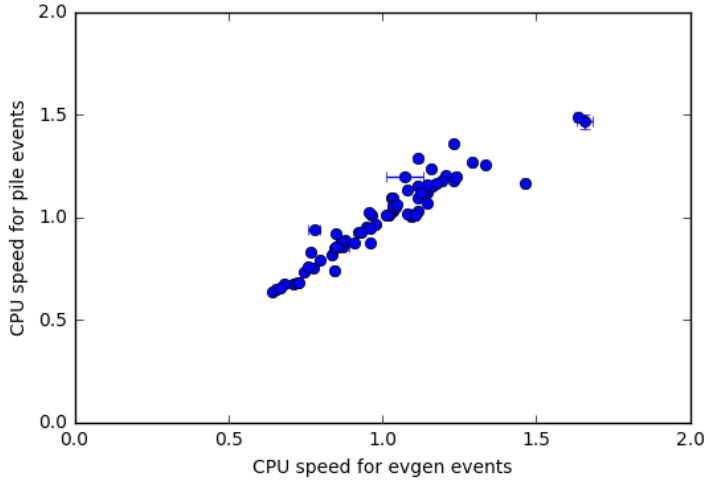


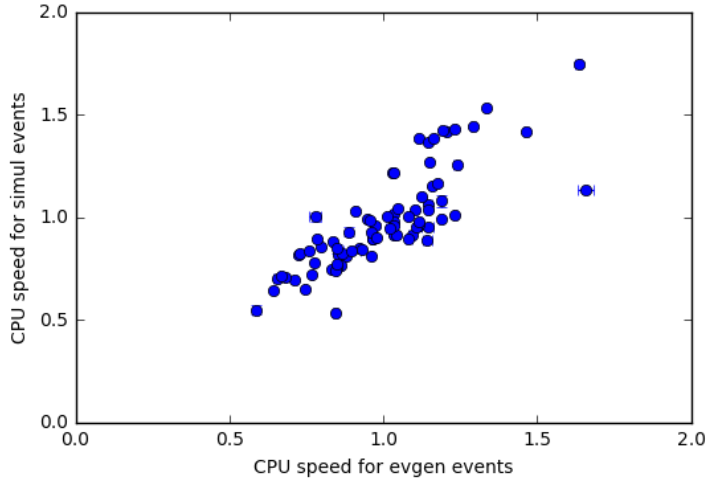


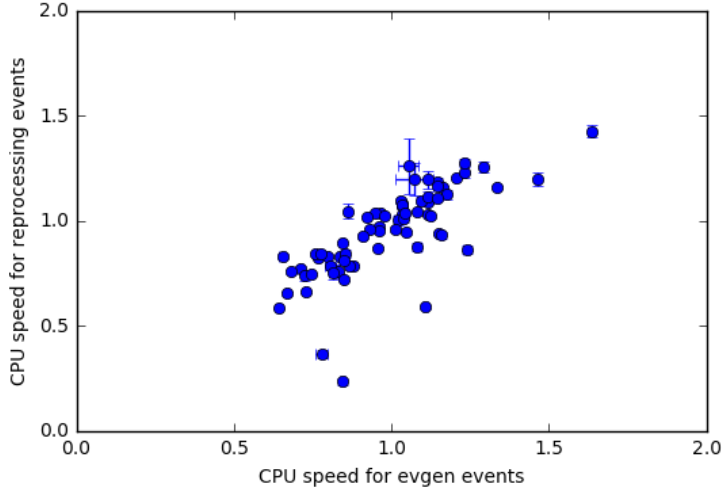
Fitted speed factors for ATLAS evgen jobs

Correlation between CPU speed factors from different job types (ATLAS)

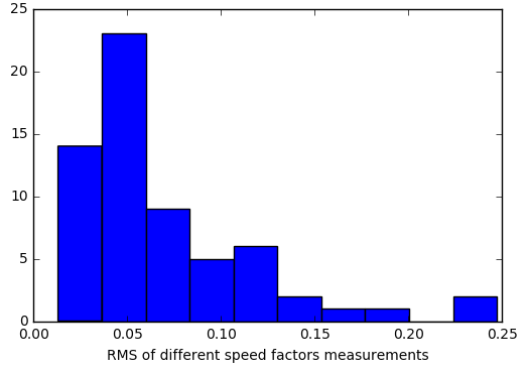
- Speed factors calculated on different types of jobs can be compared
  - The expectation is that results should be similar, as CPU time / event is not sensitive to I/O







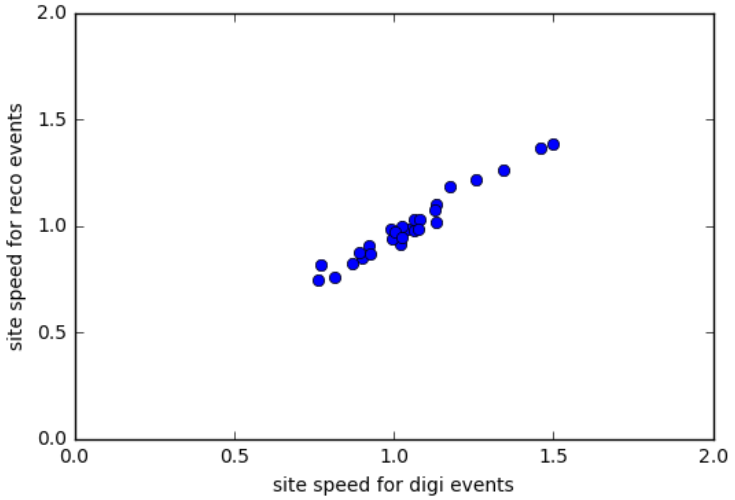
  - Different measurements agree within 5.5% in average

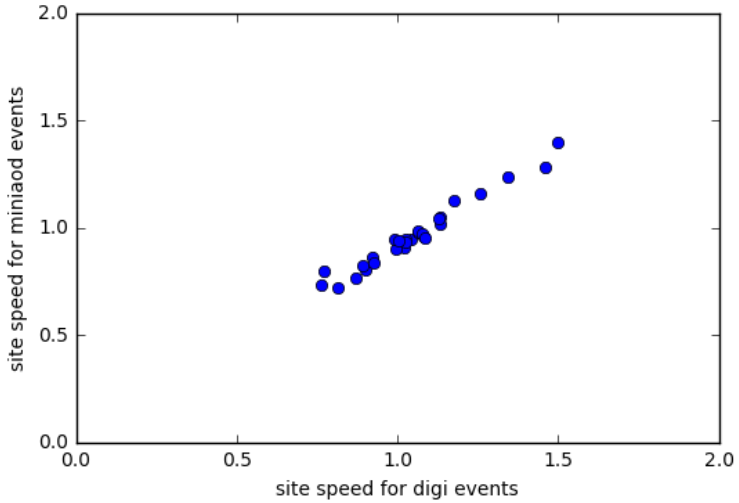


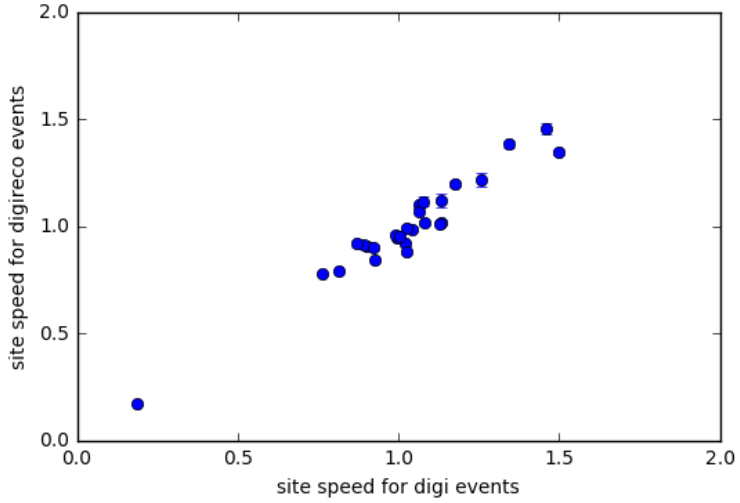
Fitting site speed

- The same method can be used to fit the “site speed”
  - Should correspond to a weighted average of the speed of the CPUs at the site, over a specific interval of time
    - Site speed may change with time, e.g. due to upgrades
  - Measurement could be used to check the estimated average HS06/core declared by the site

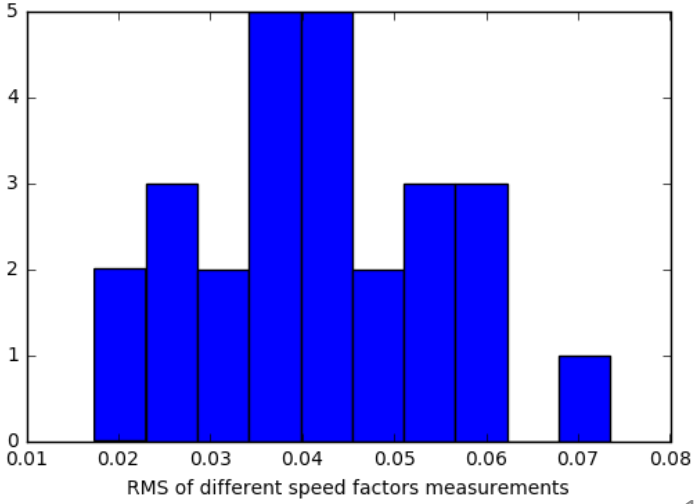
Correlation between site speed factors from different job types (CMS)







- Also when fitting site speed, results are consistent across different job types
  - Different measurements agree within 4.3% in average

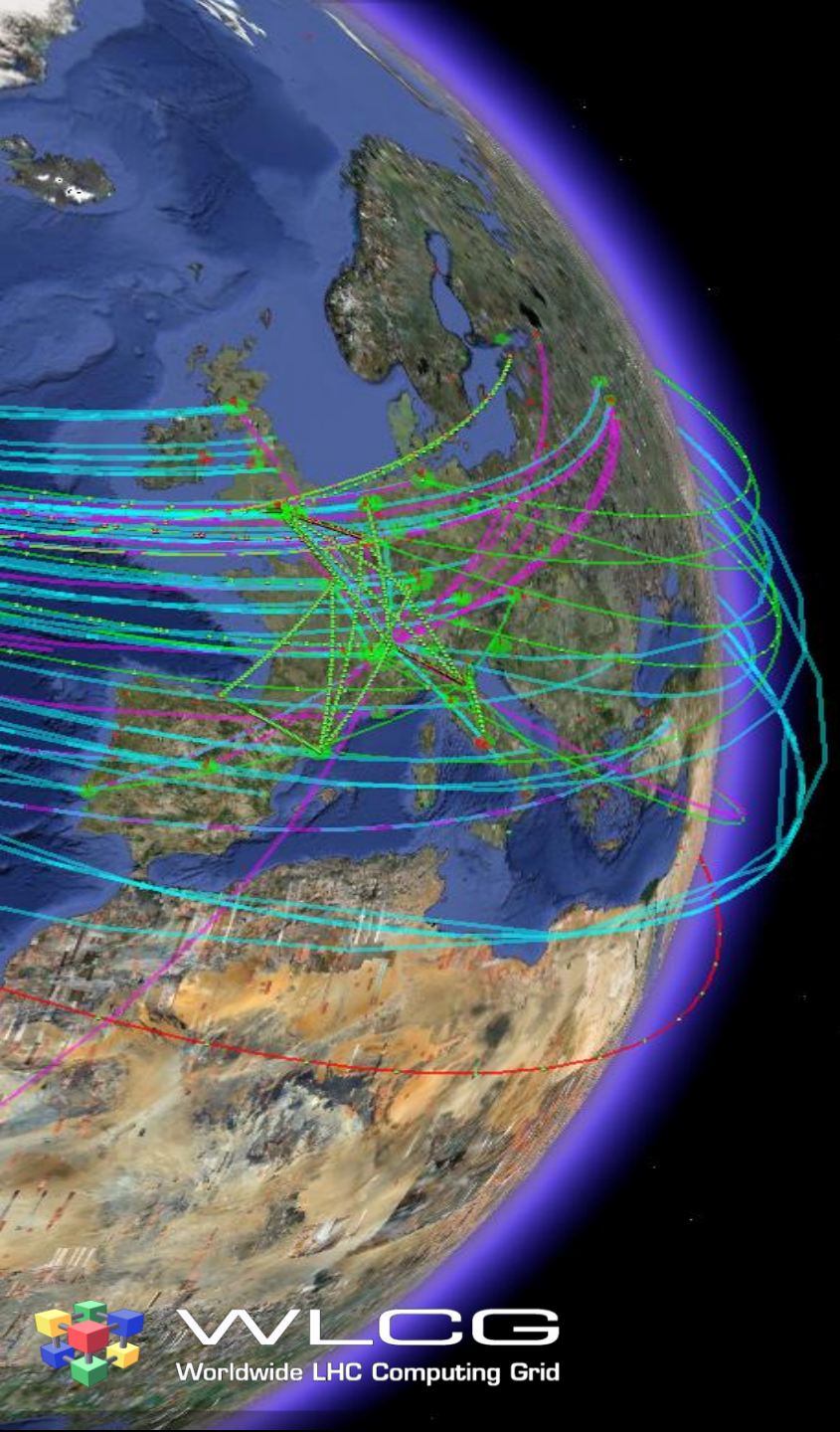


Conclusions

- Real experiment jobs can be used to measure average WN performance with a few percent precision independently on the job type
- Comparison with published HS06/core values shows a clear correlation (albeit with large systematic effects)
  - Correlation is higher when HS06/core values for “HT off” are taken
- Applications
  - CPUs: provide a benchmark based on real jobs
    - Provided that a very specific type of job is assumed as reference; otherwise, only performance ratios are measured
  - Sites: provide an alternate measurement of the average site WN performance
    - Can be used to spot unrealistic values of site HS06 power

Acknowledgements

- Thanks to all members of the ATLAS computing workflow performance group
  - A. Di Girolamo, J. Elmsheuer, A. Filipcic, S. Kama, A. Limosani, F. Legger, etc.
- Thanks to I. Vukotic, B. Bockelman and P. Saiz for ElasticSearch
- Thanks to IT data analytics experts
  - D. Giordano, D. Duellmann, C. Nieke, G. Rzehorz
- Thanks to our colleagues from the UP team



Backup slides

WLCG
Worldwide LHC Computing Grid

Used values of HS06

	cpu	hs06	cores	ht	ht_on	ht_off
2	AMD Opteron(TM) Processor 6234	402.0	48.0	na	8.375000	8.375000
5	AMD Opteron(TM) Processor 6274	484.0	64.0	na	7.562500	7.562500
6	AMD Opteron(TM) Processor 6276	510.0	64.0	na	7.968750	7.968750
9	AMD Opteron(tm) Processor 6168	193.0	24.0	na	8.041667	8.041667
12	AMD Opteron(tm) Processor 6378	558.0	64.0	na	8.718750	8.718750
22	Intel(R) Xeon(R) CPU E5420 @ 2.50GHz	72.0	8.0	off	5.625000	9.000000
24	Intel(R) Xeon(R) CPU E5440 @ 2.83GHz	75.0	8.0	off	5.859375	9.375000
25	Intel(R) Xeon(R) CPU E5450 @ 3.00GHz	78.0	8.0	off	6.093750	9.750000
27	Intel(R) Xeon(R) CPU E5520 @ 2.27GHz	98.0	8.0	off	7.656250	12.250000
28	Intel(R) Xeon(R) CPU E5520 @ 2.27GHz	125.0	16.0	on	7.812500	12.500000
29	Intel(R) Xeon(R) CPU E5540 @ 2.53GHz	137.0	16.0	on	8.562500	13.700000
31	Intel(R) Xeon(R) CPU E5630 @ 2.53GHz	112.0	8.0	off	8.750000	14.000000
32	Intel(R) Xeon(R) CPU E5630 @ 2.53GHz	141.0	16.0	on	8.812500	14.100000
33	Intel(R) Xeon(R) CPU E5645 @ 2.40GHz	152.0	12.0	off	7.916667	12.666667
34	Intel(R) Xeon(R) CPU E5645 @ 2.40GHz	187.0	24.0	on	7.791667	12.466667
39	Intel(R) Xeon(R) CPU L5640 @ 2.27GHz	174.0	24.0	on	7.250000	11.600000
41	Intel(R) Xeon(R) CPU X5550 @ 2.67GHz	112.0	8.0	off	8.750000	14.000000
44	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz	169.0	12.0	off	8.802083	14.083333
45	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz	210.0	24.0	on	8.750000	14.000000
51	Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz	215.0	24.0	on	8.958333	14.333333
55	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	278.0	16.0	off	10.859375	17.375000
56	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	355.0	32.0	on	11.093750	17.750000
59	Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz	190.0	12.0	off	9.895833	15.833333
61	Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz	372.0	32.0	on	11.625000	18.600000
62	Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz	218.0	16.0	off	8.515625	13.625000
63	Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz	262.0	32.0	on	8.187500	13.100000
64	Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz	358.0	32.0	on	11.187500	17.900000
65	Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz	444.0	40.0	on	11.100000	17.760000
67	Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz	399.0	40.0	on	9.975000	15.960000
68	Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz	374.0	20.0	off	11.687500	18.700000
69	Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz	472.0	40.0	on	11.800000	18.880000
70	Intel(R) Xeon(R) CPU E5-2665 0 @ 2.40GHz	261.0	16.0	off	10.195312	16.312500
71	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz	282.0	16.0	off	11.015625	17.625000
72	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz	352.0	32.0	on	11.000000	17.600000
73	Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz	342.0	20.0	off	10.687500	17.100000
74	Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz	420.0	40.0	on	10.500000	16.800000
75	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz	511.0	48.0	on	10.645833	17.033333
76	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz	279.0	16.0	off	10.898438	17.437500
77	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz	334.0	32.0	on	10.437500	16.700000
78	Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz	458.0	40.0	on	11.450000	18.320000
79	Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz	434.0	24.0	off	11.302083	18.083333
80	Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz	549.0	48.0	on	11.437500	18.300000
82	Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz	672.0	56.0	on	12.000000	19.200000