Benchmarking and Publishing in a Diverse Grid Site

S Jones, R Fay, J Bland
University of Liverpool

August 2016

Introduction

- This talk is about HEPSPEC06 and the BDII (readers of Bleak House will be familiar with the similar case of Jarndyce and Jarndyce.)
- Every few months, the GDB discusses benchmarking. Various opinions of the current arranges (HS06) are expressed; shortcoming are condemned; alternatives are put forward. Thus the matter moves on towards some eventual recommendation. It's impossible to say what it will be, or when it will be made.
- A similar cycle exists for the BDII. From my brief review of the GDB minutes, I'd say the plans for slimming down the BDII are getting more mature. It looks set to disappear, but we yet don't know what will be used instead, nor when it will come in.
-

Introduction

- So, since it's impossible to know what the future has in store, it's probably wise to discuss Benchmarking and Publishing in the general terms that can be applied whatever technology is chosen.
- We can be sure of this, though - at present, we have a lot of diversity out there in Gridland, and the measures we use to get the figures right have to apply right across the full spread of that.
- Thus, rather than spending this twenty minute slot discussing possibilities and probabilities, let's talk about a concrete example of a system that generalises the problem.
- The outputs of the system can be applied to any technology, when it comes up.

Introduction

- Benchmarking: estimate the power of a system.
- Accounting: estimate the effort consumed by jobs.
- Publishing: transmit benchmark values to others.
- We'll discuss how we manage these aspects at Liverpool.
- Since UK sites are operating, then this information is obviously reasonably well understood already by many in GridPP. The intent here is :
  - To review the key aspects for the benefit of all.
  - To try to create a shared view on how it is done.
  - To systematise the process, yielding requirements for any replacement.

Introduction

- The Liverpool grid site has become quite diverse; our site is heterogeneous at several levels. We have a range of worker-node hardware, running over over three different clustering technologies:
  - CREAM/Torque
  - ARC/Condor
  - VAC
- This talk will cover benchmarking, publishing (and accounting) across the spread.
- We don't use the glite-CLUSTER node type, which aggregates the publishing information from several resources to model a heterogeneous cluster.

Introduction

- We will discuss HW benchmarking, and the requirements for publishing accounting information and cluster power.
- We'll also show a database tool to manage the site's cluster layout and do the arithmetic for YAIM configuration for CREAM/Torque, and for configuration without YAIM for Condor and VAC.
- These techniques comprise an integrated approach which may be used directly elsewhere, or adapted to suit different scenarios.
- This work is based on the GridPP Publishing Tutorial, which describes how to do the job manually:
  - https://www.gridpp.ac.uk/wiki/Publishing_tutorial

What are we measuring?

- We are measuring the power at our site, and how much work we do with that power, using a benchmark.
- A benchmark is a standard reference against which things may be compared.
- The WLCG lays down the requirements for benchmarking, and they have communicated to GridPP that the current benchmark is HEPSPEC06 (HS06), in 32 bit mode running on an SL6 system. Links:
  - Hepix Benchmark Site: http://w3.hepix.org/benchmarks/doku.php
  - GridPP HEPSPEC06 Table: https://www.gridpp.ac.uk/wiki/HEPSPEC06
- Support for publishing the required benchmark data is provided in the GLUE schemas.

Units

- Accounting and power figures are now generally in HS06.
- However, values are sometimes transmitted to the outside world in SpecInt2K (SI2K). For compatibility, it was decided that SI2K would be redefined to be equal to 1/250th of a HS06, and the values would then be transmitted in the new SI2K units (which might be called "bogoSpecInt2k" if I were being rigorous.) HS06 values are converted to this SI2K by timesing them by 250 prior to transmission.
- Accounting, in this talk, uses HS06 hours (HS06.h) similar to the convention for electricity accounting (i.e. kW.h). These values may be prefixed by a k, or m or whatever, or postfixed by .h, .s etc. depending on the magnitude or time period in question.

The task at hand; terminology

- Each worker node system is composed of 1..N CPUs.
- A CPU contains 1..N cores.
- Each core can run 1..2 hyperthreads.
- Jobs are allotted to “slots” on worker nodes. A slot (aka logical cpu ...) is a “space” for a single core job. Hence it's “full” if it's running a job, or “empty”. Total slots is the number of slots on a node.
- We wish to set total slots to get maximum node throughput; it will be <= hyperthreads.
- Hence we vary total slots and then benchmark the node at each step to find the best value.
- When the node is running a full complement of CPU-bound jobs, then the node will be working at maximum expected load and giving maximum applied computing power.

Hardware benchmarking

- Sites have put benchmarks in a table for others:
  - https://www.gridpp.ac.uk/wiki/HEPSPEC06
- However, results depend on slots used and hardware variations. It might be best to run your own benchmarks.
- The HS06 benchmark is built on top of another benchmark.
  - SPEC06: https://www.spec.org/cpu2006/
- That part (product filename is spec2006-1.2.tar.gz) requires a licence.
- The HEP part is free to download, and has instructions:
  - HS06: http://w3.hepix.org/benchmarks/doku.php?id=bench:howto
- It is actually the config files (product filename is spec2k6-2.23.tar.gz).

Hardware benchmarking

Whatever the clustering technology, the HW benchmarking process is the same.

1. License the SPEC2006 benchmark suite, to obtain
2. spec2006-1.2.tar.gz
3. Download the free spec2k6-2.23.tar.gz
4. Drain a system of the type you wish to benchmark.
5. Make a working directory and put the products in it.
6. Unpack spec2006-1.2.tar.gz then spec2k6-2.23.tar.gz as per vendor instructions.
7. Run HS06 on your node, in 32 bit mode.
8. View the results with the runspec-result-sum.sh script.
9. Publish the values, to be discussed.

BUT...

The best number of slots

- Making slots == hyperthreads does not necessarily maximise throughput.
- Experiments show that it is sometimes necessary to choose a number of slots that is higher than the number of cores but slightly lower than hyperthreads. Contention?
- Anyway, there is often negligible if any increase in overall throughput when approaching slots == hyperthreads.
- So you get practically the same throughput with slots < hyperthreads AND ...
- You won't need as much memory in each node, because it will run fewer jobs but each job will run more efficiently.

The best number of slots

- The standard script (runspec.sh) that comes with the products assumes you want to use slots == hyperthreads, i.e. it runs as many instances of the benchmark as hyperthreads.
- So we changed the benchmark script to allow slots to be set by the user as a parameter.
- It does not change the results, only the number of benchmark instances.
- The new file is called runspec32-N.sh, and we give instructions how to use it:
  - Modified script: http://hep.ph.liv.ac.uk/~sjones/gpp37/hepspeccing

The best number of slots

- With this modification, the procedure for choosing total slots is :
- For each type of node at your site, run an instance of the benchmark for every number between cores and hyperthreads, to cover the whole area.
- Compare all the results and select the number of benchmark instances (slots) that gives the maximum applied computing power overall from these scenarios, i.e. chose the sweet spot for the slots used in this node type.
- Where the sweet spot is flat (e.g. practically the same overall throughput is obtained with 14, 15 or 16 slots on a 16 hyperthread node), choose the lowest because this combines the highest throughput with the most memory available per job.
- And, in any-case, always chose a number that at least provides adequate memory per job.

The best number of slots

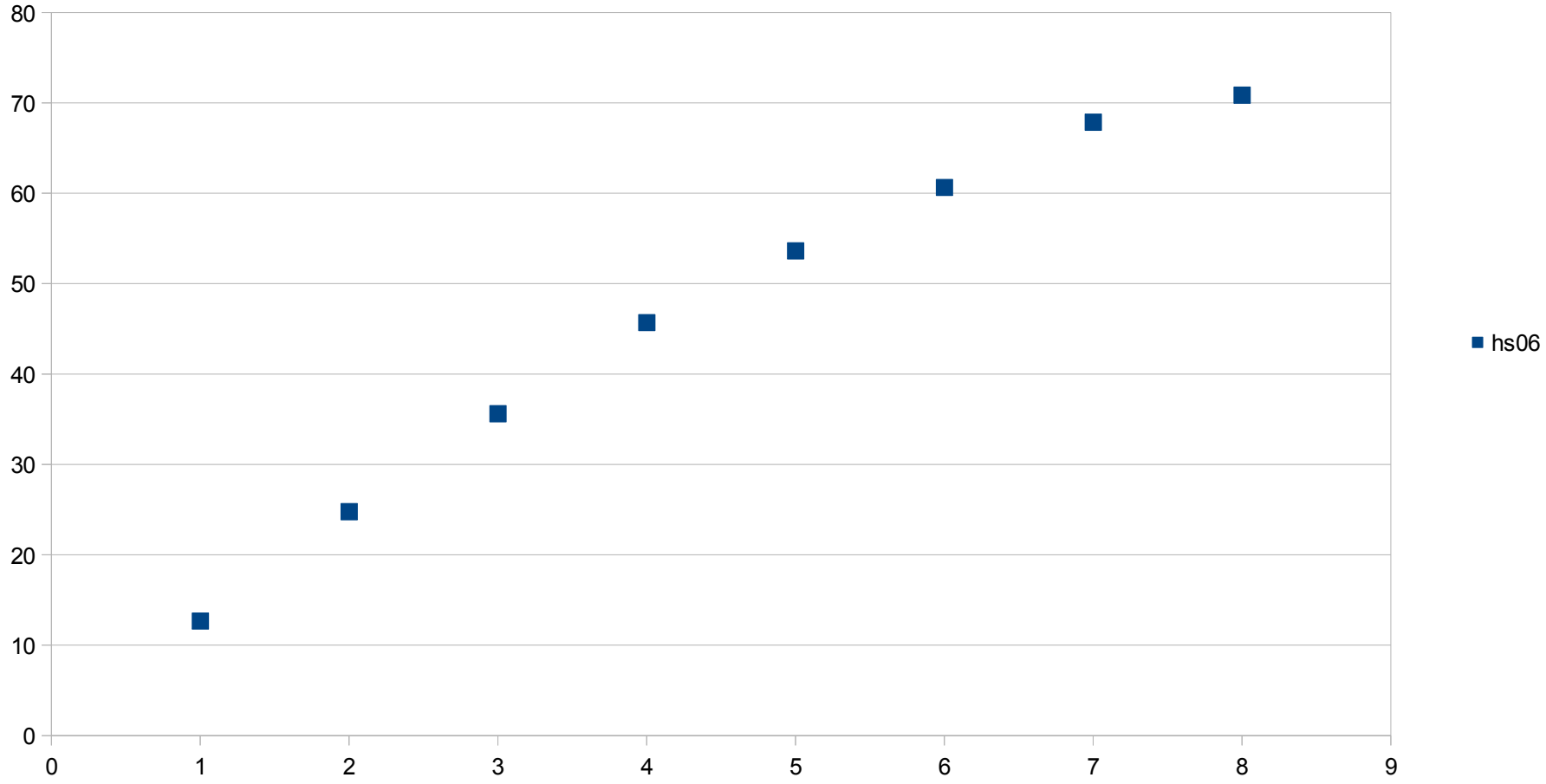
- Other concerns that are outside the scope this is talk are:
- We don't consider multicore slots. Sites may wish to use a multiple of 8 for a node that runs multicore jobs; this would allow the node to be maximally utilised even if there are no single-core jobs.
- Constraints like bandwidth or data rates may also be factors.
- The best number of slots for a certain type of node used with one technology, e.g. CREAM/Torque, may not be not the best number of slots for the same type of node when used with another technology, e.g. VAC.
- E.g. the context of VAC VMs takes some RAM. We found that machines went into swap with a E5620 using 12 slots, which does not happen on CREAM/Torque. Thus we put 10 slots on those VAC nodes. There is little practical difference to throughput because we are close to the top of the curve anyway, see Figure 2 below.

The best number of slots

- The following plots, which are not definitive at our site, serve to verify these observations.
- Initially, there is only a slight plateau near max slots; on more advanced hardware with more cores, the plateau gets gradually more apparent, and sometimes the slope of the curve goes negative.
- However, each run does not yield exactly the same results - spreads of variation of between 1% to 3% are quite typical, indicating the noise limits of this benchmark.

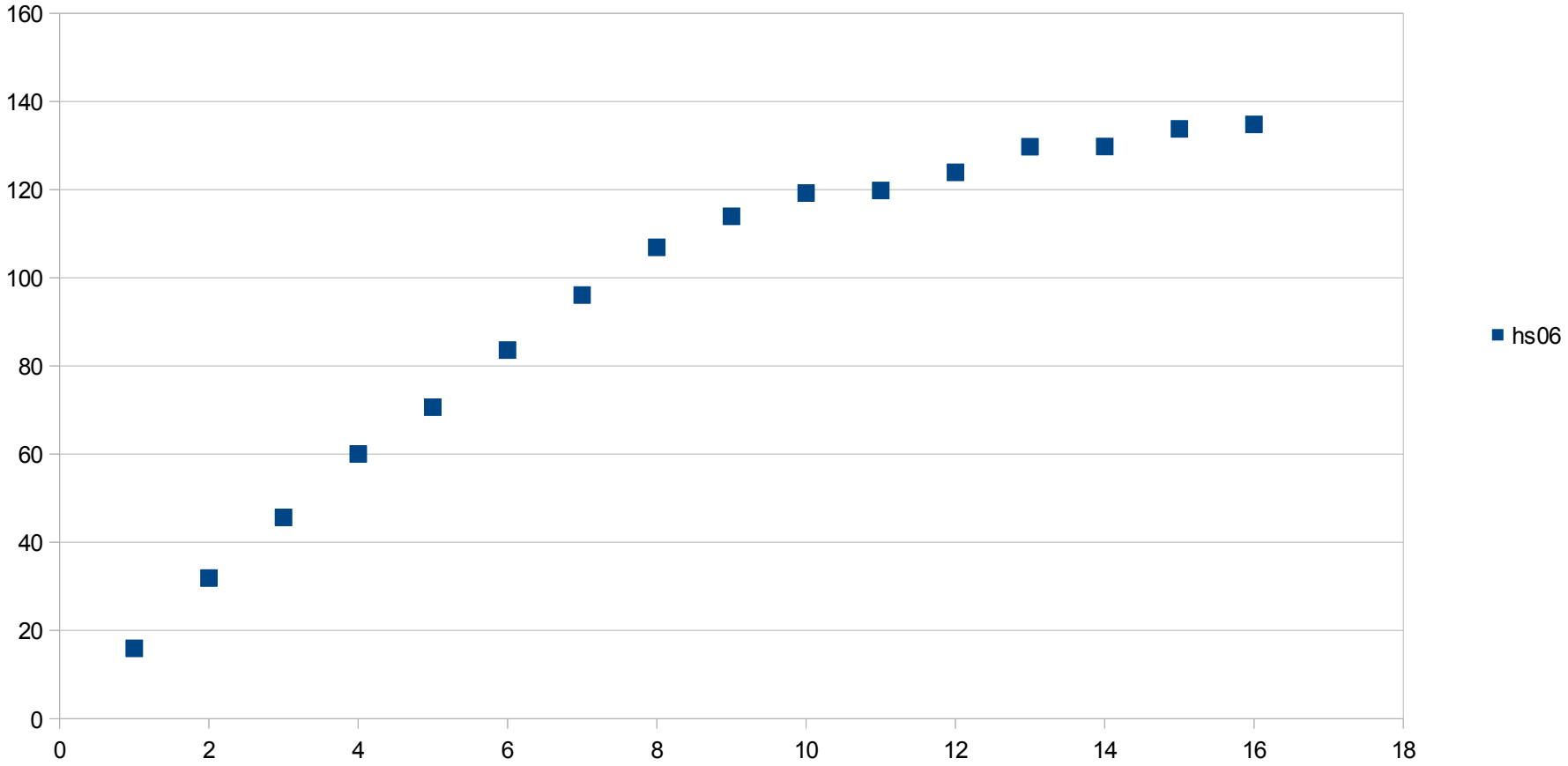
The best number of slots

Figure 1: L5420 - 8 hyperthreads; no plateau.



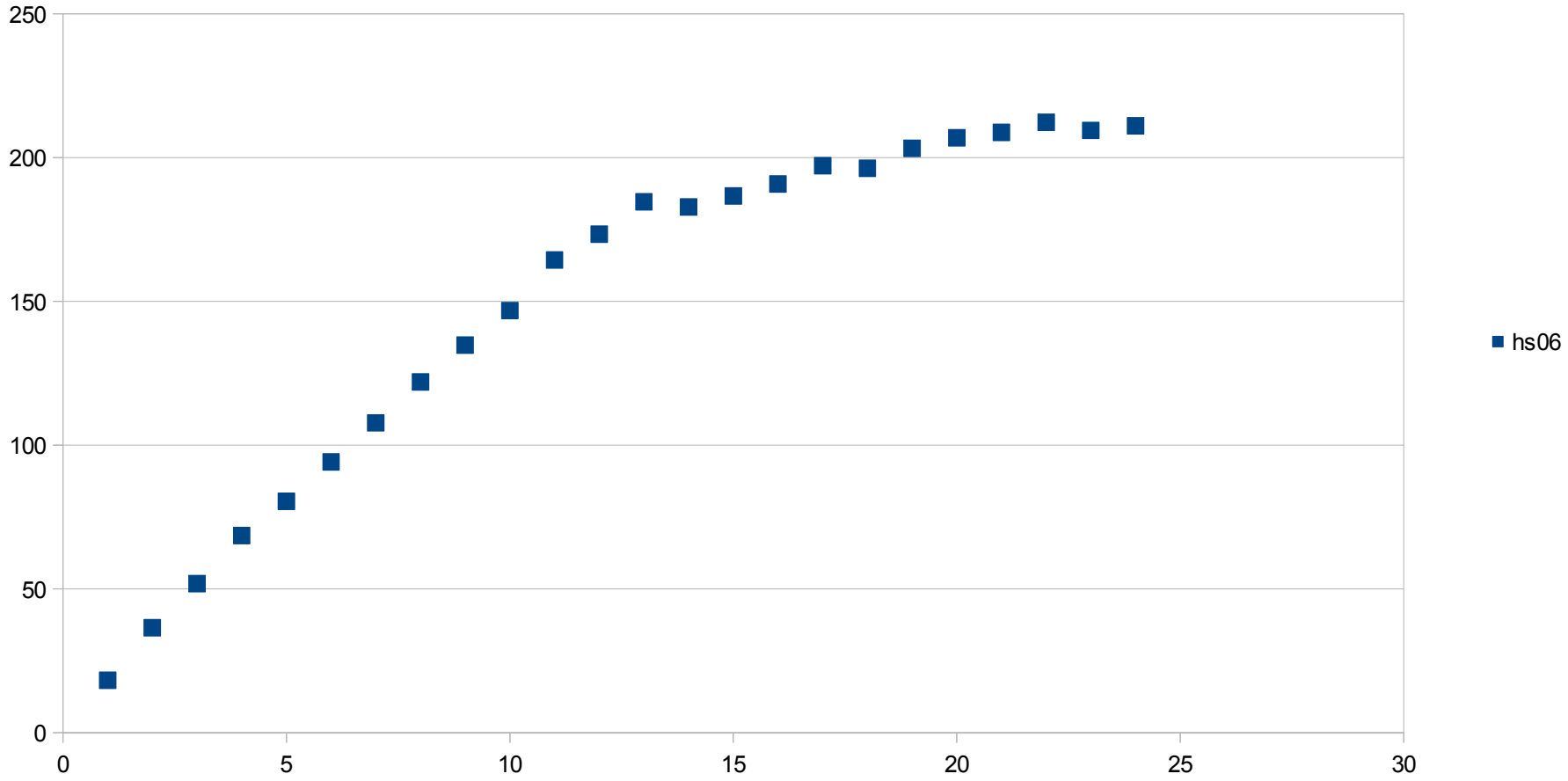
The best number of slots

Figure 2: E5620 - 16 hyperthreads; plateau emerging.



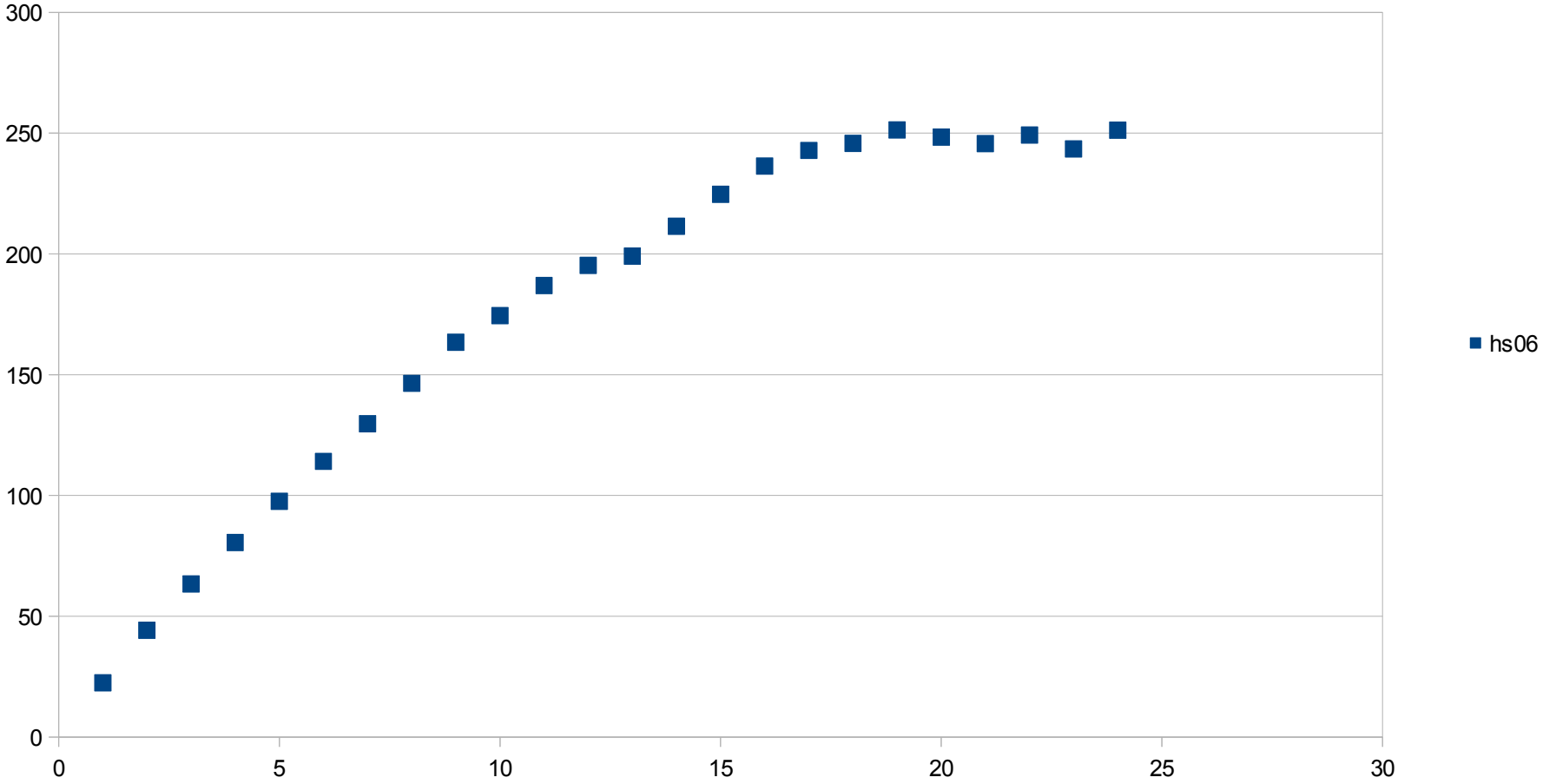
The best number of slots

Figure 3: X5650 - 24 hyperthreads; plateau and drop off.



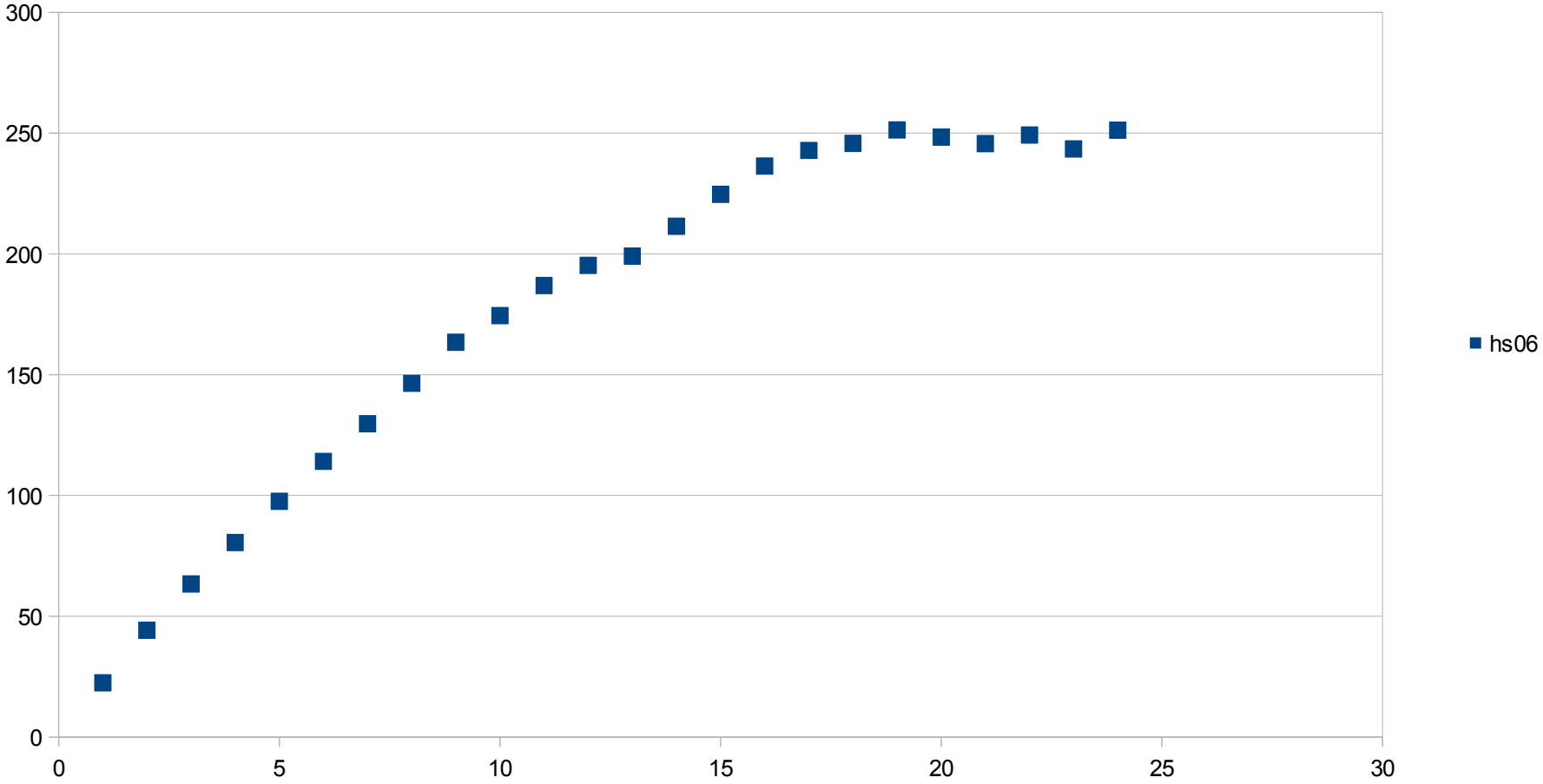
The best number of slots

Figure 4: E52630 v2 - 24 hyperthreads; plateau, drop off and recovery.



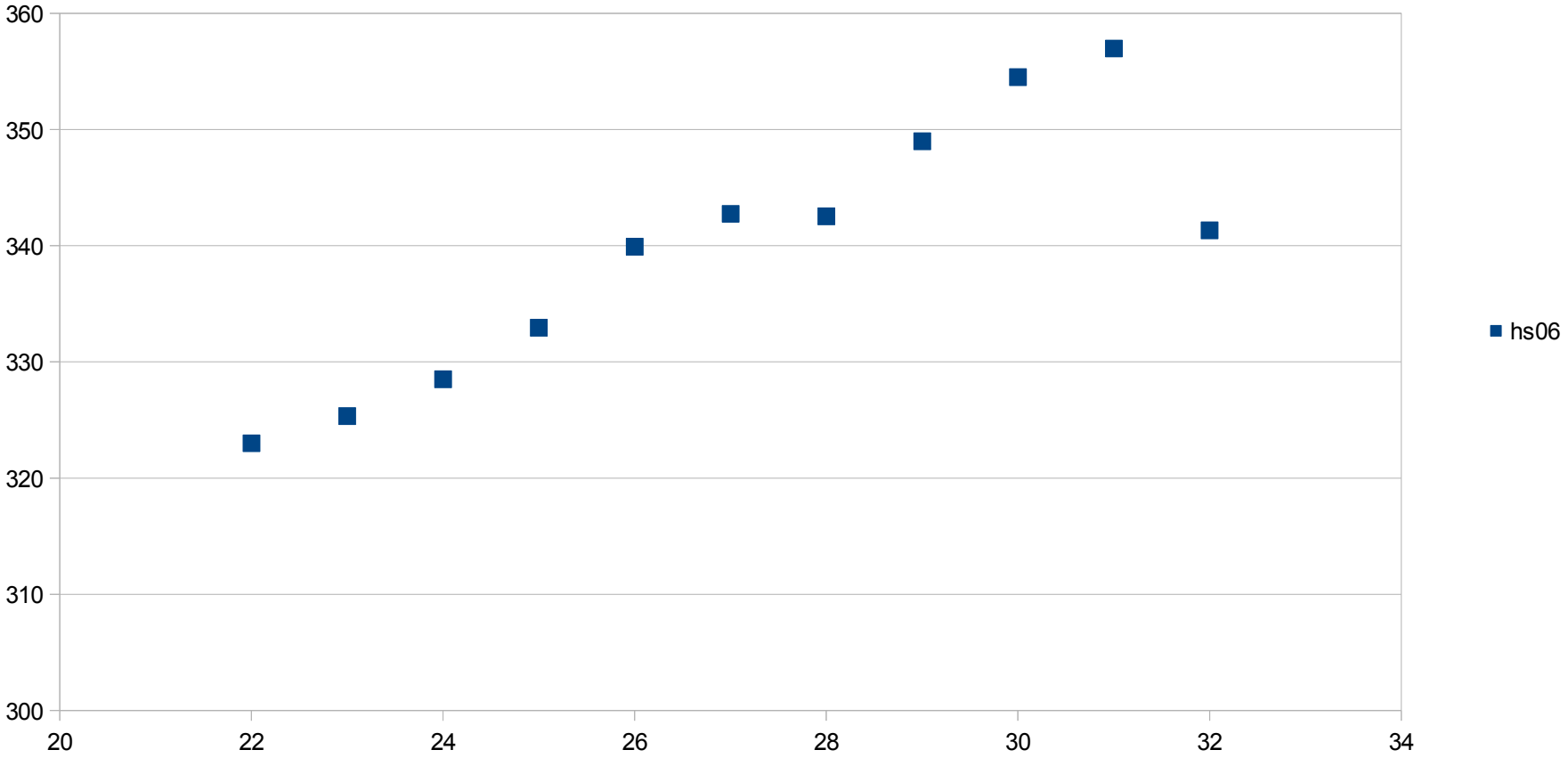
The best number of slots

Figure 4: E52630 v2 - 24 hyperthreads; plateau, drop off and recovery.



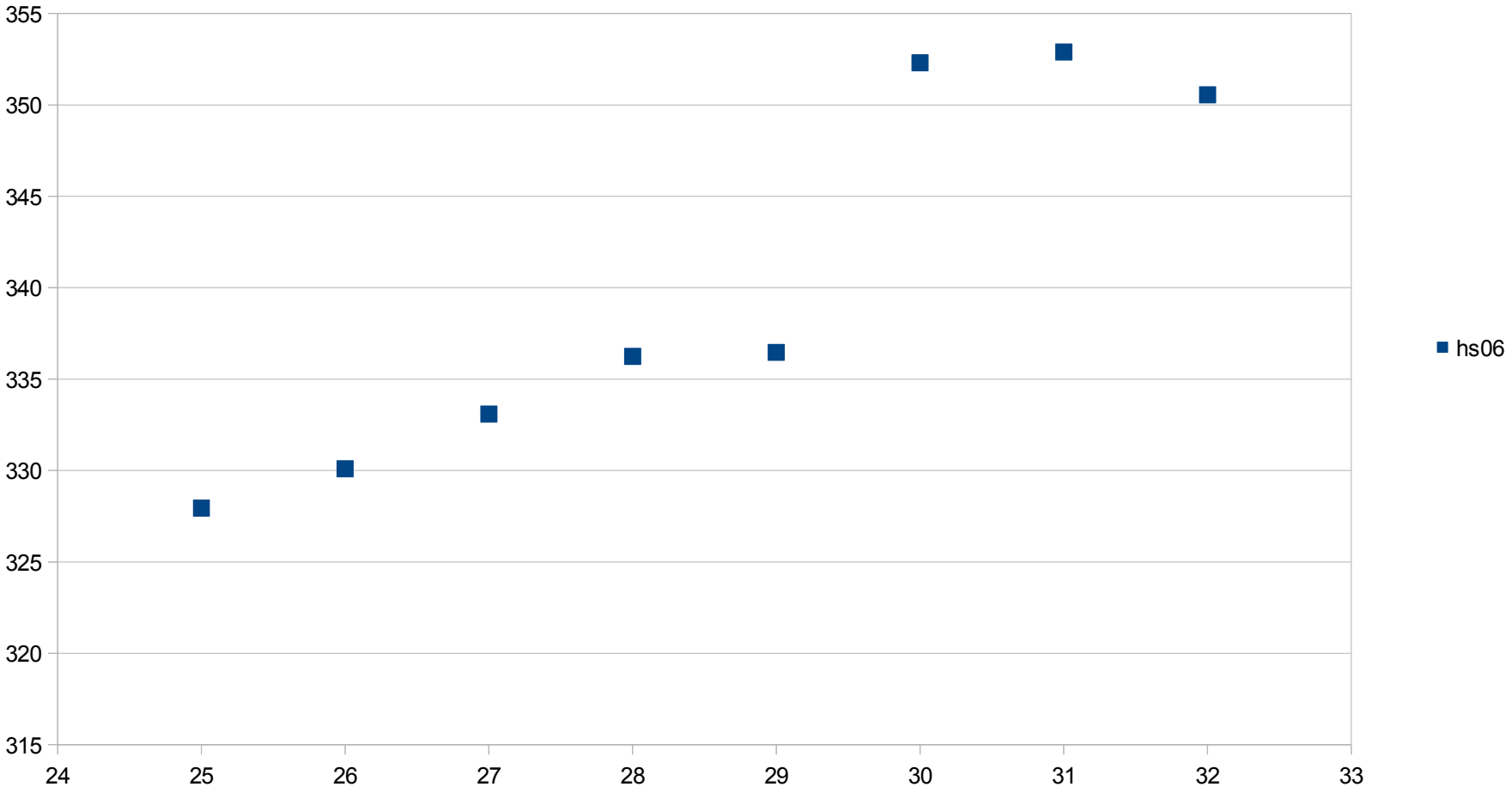
The best number of slots

Figure 5 - E52630-v3 - 32 hyperthreads; last point is outlier



The best number of slots

Figure 5 - E52630-v3 - Retry the peak. Weird results. 5% noise...

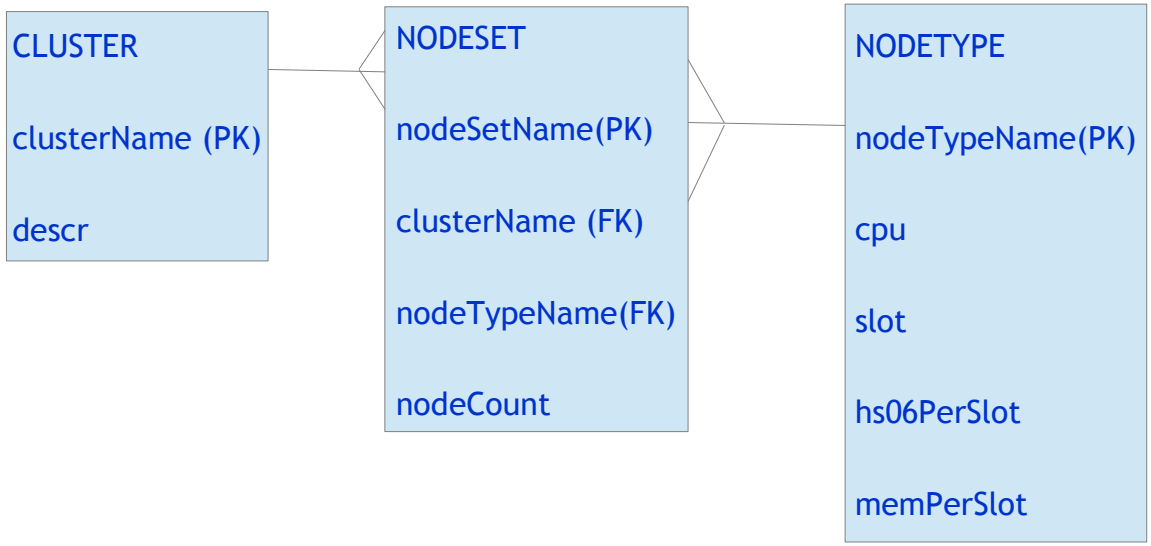


Site Layout

- To recap: Following the procedures, we run the benchmarks over the whole area for each node type at our site and we decided the total slots for each type.
- Next I'll discuss how we manage the site layout, whereby the worker-nodes are allocated to various types of cluster technology, namely CREAM/Torque, ARC/Condor or VAC.
- The eventual plan could be to expand this so that workernodes can be seamlessly reallocated to different technology on an as-needed basis, making workernodes technologically agnostic.
- I expect sites have their own ways of tracking this information. So I'll show how we do the configuration using the Liverpool tools, and how to do it manually if you have your own arrangements for handling the layout.

Site Layout

- We use a model of the site that I'll refer to as the Site Layout Database. The model is the simplest that can do the job, and it uses three tables arranged in this relationship diagram.



Site Layout

- We maintain the layout in this DB. CLUSTER gives a name and description to each cluster; CREAM/Torque, ARC/Condor and VAC.
- The other tables define sets of nodes and types of nodes respectively, and these are assigned to clusters. A particular cluster can have many different types of node, and a particular type of node can occur in many clusters. NODESET maps what nodes of what type belong to any specific cluster.
- We don't model node ranges or physical locations. Instead, the “nodeCount” attribute lets us model sets of nodes.
- By convention, a clue to the physical location is given in the nodeSetName attribute, which is indicative of the rack location where sets of nodes reside (e.g. 23p1 represents nodes 1 to 10 in rack 23, while 23p2 represents nodes 11 to 20 etc.)
- With this model, we can have heterogeneous racks.

Site Layout

- The primary output from the database is a report which is produced by a tool called clusterReport.pl.
  - Example layout: http://hep.ph.liv.ac.uk/~sjones/gpp37/sitelayoutdb/report.txt
- Amongst other things, it gives us the benchmark figures for entry into our publishing system, which will be discussed in the next sections. The database is maintained using an SQL user interface.
- Note: we used spreadsheets for this before we put the database together. They contained redundancy (which sounds like a good thing, but it's not) and editing errors were made now and again.
  - Site Spreadsheet: http://hep.ph.liv.ac.uk/~sjones/gpp37/spreadsheet/

Benchmark Publishing Config

- I'll show how we map the values from the site layout database into the configurations of each cluster type. I'll use the report from our site. I'll also show briefly how to do this by hand.
- The first section in the report is general, and just shows the details for each node type used in the entire site.

```
Node types:
     Name,    CPUs, Slots, HS06 (Slot), RAM (Slot), Scale factor
 BASELINE,       0,     0,       10.00,       0.00,    0.0000
    L5530,       2,     7,       12.34,       3.50,    1.2340
    L5420,       2,     8,        8.90,       2.00,    0.8900
    E5620,       2,    12,       10.63,       2.00,    1.0633
    X5650,       2,    24,        8.66,       2.08,    0.8660
  E5-2630,       2,    23,       11.28,       2.17,    1.1280
 E5-2630V3,      2,    32,       11.07,       4.12,    1.1070
 E5620-VAC,      2,    10,       12.05,       2.40,    1.2050
```

Benchmark Publishing Config

- By convention we use the name of the CPU as the unique node type name; one could have different node types with the same CPU if (for example) one type had (say) fewer slots, e.g. E5620-VAC. The other fields are:

```
CPUs: Number of CPUs in each node of that node type.
SLOTS: Number of slots chosen for each node of that node type.
HS06 (Slot): HS06 per slot
RAM (Slot): RAM per slot
Scale Factor: Used to scale run times in heterogeneous clusters.
```

- Some words are needed on BASELINE nodetype and Scale factor attribute. The APEL accounting system accepts job run time data from sites, i.e. how long each job ran for. It also accepts a figure representing the power of the node the job ran on. This allows the computing power spent by the job to be computed by multiplying time by power, giving HS06.s. BUT ....

Benchmark Publishing Config

- This would be fine for a site that has one node type. But sites have multiple node types, with different powers.
- Thus the site layout database must contain one NODETYPE called BASELINE. This is an abstract NODETYPE - it is not necessary for real nodes like this to exist. The job run time data from jobs that ran on real nodes is scaled (made longer or shorter) commensurately to how powerful the real node is compared to the abstract BASELINE node.
- By way of example: At our site, the BASELINE node type has a power of 10 HS06. We tell the accounting system what BASELINE is, and the accounting system will therefore assume that each job ran on a node with 10 HS06s of power. An L5530 actually gives 12.34 HS06 per slot. Hence the scale factor is 12.34/10 = 1.234, so the run time for a job that ran on a L5530 is multiplied by 1.234 by the system, to make it correct for then sending into APEL.

CREAM/Torque

CREAM/Torque configurations typically make use of YAIM. We'll use the right section from the report:

- http://hep.ph.liv.ac.uk/~sjones/gpp37/sitelayoutdb/report.txt

```
Cluster:  TORQUE_BATCH_HAMMER
Set label,  Nodetype, Number, Slots, Slot HS06,  HS06
     25p2,   E5-2630,     10,    23,     11.28,  2594.40
Cluster properties:
HS06   :      2594
Physical CPUs       :        20
Logical CPUs (slots):       230
Cores:    11.500
Benchmark:    11.280
CE_SI00:   2820
CPUScalingReferenceSI00:  2500.000
```

CREAM/Torque

- Note 1: This small cluster only contains one type of node, the E5-2630. See the section in the report for CONDOR_BATCH_HAMMER to see an example of a cluster with multiple types of node.
- Note 2: You should obviously feed the Slots attribute into your batch system configuration (e.g. via puppet), via whatever mechanism that involves for Torque, Condor, VAC, whatever...
- Getting back to Publishing ... there's a straight forward mapping from the report attributes to the YAIM variables. Same names, same values.

CREAM/Torque

- Here is the same information, in YAIM format, that is put in the site-info.def YAIM config for the CE (with a caveat, TBD below.)

```
CE_PHYSCPU=20
CE_LOGCPU=230
CE_CAPABILITY="CPUScalingReferenceSI00=2500 Share=atlas:XX Share=lhcb:YY glexec"
CE_SI00=2820
CE_OTHERDESCR=Cores=11.5,Benchmark=11.28-HEP-SPEC06
```

- Run YAIM and your _single_ CE will publish the right values.
- BUT...

CREAM/Torque

- The YAIM mapping above would be good for a site with one CE, so the only trick left is to describe how we deal with the case where we have multiple, e.g. two, CEs talking to the same Torque server.
- In this case, it would be arithmetically correct to set the logical and physical cpu counts in one CE, and set them to zero in the other, else double counting would occur.
- Unfortunately this raises divide by zero errors elsewhere. To workaround that, we set set the logical and physical cpu counts to 1 in one CE, and set them to the “count – 1” in the other CE. This kludge gives the correct arithmetic while avoiding the zero division, so everyone is happy enough.
- You can adjust this technique for any number of CEs.

CREAM/Torque

- If you want to do it by hand, the Publishing Tutorial lays out the detail: https://www.gridpp.ac.uk/wiki/Publishing_tutorial
- To summarise, count all slots in all nodes and put it in CE_LOGCPU. Count all cpus in all nodes and put it in CE_PHYSCPU.
- The CPUScalingReferenceSI00 component of CE_CAPABILITY is just the BASELINE HS06 times 250 to turn it into bogoSI2k.
- The Cores component of CE_OTHERDESCR is just

  CE_LOGCPU / CE_PHYSCPU.
- The Benchmark component is the average HS06 benchmark strength of a single slot (i.e. total HS06/ logical cpus), tagged with -HEP-SPEC06.
- The CE_SI00 attribute is Benchmark expressed in SI2k as an integer (by multiplying it by 250 and rounding it.)

ARC/Condor

- In ARC/Condor, the publishing settings are put directly in the /etc/arc.conf config file (no YAIM), so the changes can be rolled out in (say) Puppet to control your configuration.
- The process for publishing the HEPSPEC in an ARC/Condor set-up is similar to that used for CREAM/Torque, described above.
- However, the Publishing tutorial describes a situation where YAIM is used to convert and transfer the information into a CREAM/Torque BDII. In the case of ARC/Condor, the same data has to be transposed into the /etc/arc.conf configuration file so that the ARC BDII can access and publish the values.
- The following table shows how to map the YAIM values referenced in the tutorial to the relevant configuration settings in the ARC system.

ARC/Condor

Description	YAIM variable	ARC Conf Section	Example ARC Variable	Notes
Total physical cpus in cluster	CE_PHYSCPU=114	N/A	N/A	No equivalent in ARC
Total slots /cores/logical-cpus/unislots/threads ... in cluster	CE_LOGCPU=652	[cluster] and [queue/grid]	totalcpus=652	Only 1 queue; same in both sections
Accounting scaling	CE_CAPABILITY=" CPUScalingReferenceSI00= 2500 ...	[grid-manager]	jobreport_options="... benchmark_value: 2500.00"	Provides the reference for accounting
Power of 1 logical cpu, in HEPSPEC06 * 250 (bogoSI00)	CE_SI00	[infosys/ glue12]	cpu_scaling_ reference_si00= "2970"	See YAIM docs; equivalent to benchmark * 250
Cores: the average slots in a physical cpu	CE_OTHERDESCR= Cores=n.n, ...	[infosys/ glue12]	processor_other_ description= "Cores=5.72 ..."	YAIM var shared with benchmark (below)
Benchmark: The scaled power of a single core/logical-cpu/unislot/thread ...	CE_OTHERDESCR=...., Benchmark=11.88-HEP-SPEC06	[infosys/glue12]	processor_other_ description="..., Benchmark= 11.88-HEP-SPEC06"	YAIM var was shared with Cores (above)

VAC

- The VAC cluster is a headless collection of nodes. We treat it as just another cluster for these purposes. As per ARC/Condor, no YAIM support is available - the required setting is just put in the config file, directly from the report. Since a VAC cluster has no BDII information system, the publishing requirements are simple. The following is written up in the VAC documentation, which must be read to form a complete picture of VAC functionality.
- VAC nodes independently contact the central APEL system to send their accounting data over. The interesting parameter, in the context of benchmarking, is the hs06_per_cpu parameter in vac.conf, which is set to the HS06 value of the node type when running the expected maximum load. VM overhead is considered to be almost negligible. There are several other VAC parameters needed to connect to APEL, but these are outside my scope (see https://www.gridpp.ac.uk/vac/).

Testing

- We have provided a script that can be adapted to test the published power of your site by querying your BDII.
  - http://hep.ph.liv.ac.uk/~sjones/gpp37/powertestscript
- Since VAC needs no BDII, it is not supported in this script.
- We hope to do more work on ways to confirm that the benchmarking, publishing and ultimately the accounting is good.
- This may need some feedback or heuristics to estimate the amount of work done, which is then compared to the actual accounting figures to make sure everything was transmitted properly.
- I see this as a priority - at present it is not so easy to be sure that all your accounting is being properly sent, received, processed and tallied up.

Review

- We've considered what benchmarking is, how to do it, and some of its limitations. The graphs give an idea of how CPU performance tails off as slots approaches hyper-threads, and how this can be used to maximise throughput and optimise the memory per job at the same time.
- We've discussed how to use the output of the benchmarking process in a site layout database, which we use to maintain the allocation of nodes to our varied cluster technologies on an on-going basis. The database also does the arithmetic for producing the published data, but we've shown how to do the job by hand, as well.
- Finally, we discussed how the information feeds into the publishing systems of various clustering technologies, and briefly mentioned a test that can show that some of this data is correct.

References

Description	Link
Hepix Benchmark Site	http://w3.hepix.org/benchmarks/doku.php
GridPP HEPSPEC06	https://www.gridpp.ac.uk/wiki/HEPSPEC06
John Gordon's notes	http://indico.cern.ch/event/63028/session/2/attachments/1006732/1432188/Gordon-benchmarking.pptx
SPEC06	https://www.spec.org/cpu2006/
HS06 instructions	http://w3.hepix.org/benchmarks/doku.php?id=bench:howto
Modified benchmark script:	http://hep.ph.liv.ac.uk/~sjones/gpp37/hepspeccing
L'pool site layout report	http://hep.ph.liv.ac.uk/~sjones/gpp37/sitelayoutdb/report.txt
Publishing Tutorial	https://www.gridpp.ac.uk/wiki/Publishing_tutorial
ARC/Condor Build	https://www.gridpp.ac.uk/wiki/Example_Build_of_an_ARC/Condor_Cluster
Test power publishing	http://hep.ph.liv.ac.uk/~sjones/gpp37/powertestscript/power.pl
Equivalent explanation	http://northgrid-tech.blogspot.co.uk/2010/04/scaling-capacity-publishing-and.html
VAC	https://www.gridpp.ac.uk/vac/
Old Site DB Spreadsheet	http://hep.ph.liv.ac.uk/~sjones/gpp37/spreadsheet/

The End

I hope this is useful for other site admins, and that it helps us to build a common view on how this is done.

Many thanks for your attention.

Questions or suggestions?

The End