
CERN

CPU Performance Optimizations of Virtualized Compute Resources in the CERN Cloud Service

Arne Wiebalck
Tim Bell
Sean Crosby (Univ. of Melbourne)
Ulrich Schwickerath

HEPiX Autumn Meeting
BNL, Upton, N.Y., U.S.
Oct 14, 2015

The “20% overhead” problem

- On our batch full node VMs we noticed that the HS06 rating was ~20% lower than on the underlying host
  - Full node VMs are needed to the limit of the total number of hosts in LSF
- Smaller VMs behaved much better: ~8%
  - The sum of simultaneous HS06 runs on 4x 8-core VMs on a 32-core host
  - Better, but still pretty high
- IN2P3 reported significant performance penalties for ATLAS MC jobs when EPT* was switched on
  - 26% vs. 6% in wall clock time for EPT on vs. EPT off compared to bare metal
  - Surprising as EPT is supposed to make things faster

***EPT:** Extended Page Tables is Intel’s implementation of a hardware-assisted virtualization technology for page table management (secondary address translation or nested pages). AMD’s implementation is called RVI (Rapid Virtualization Indexing).

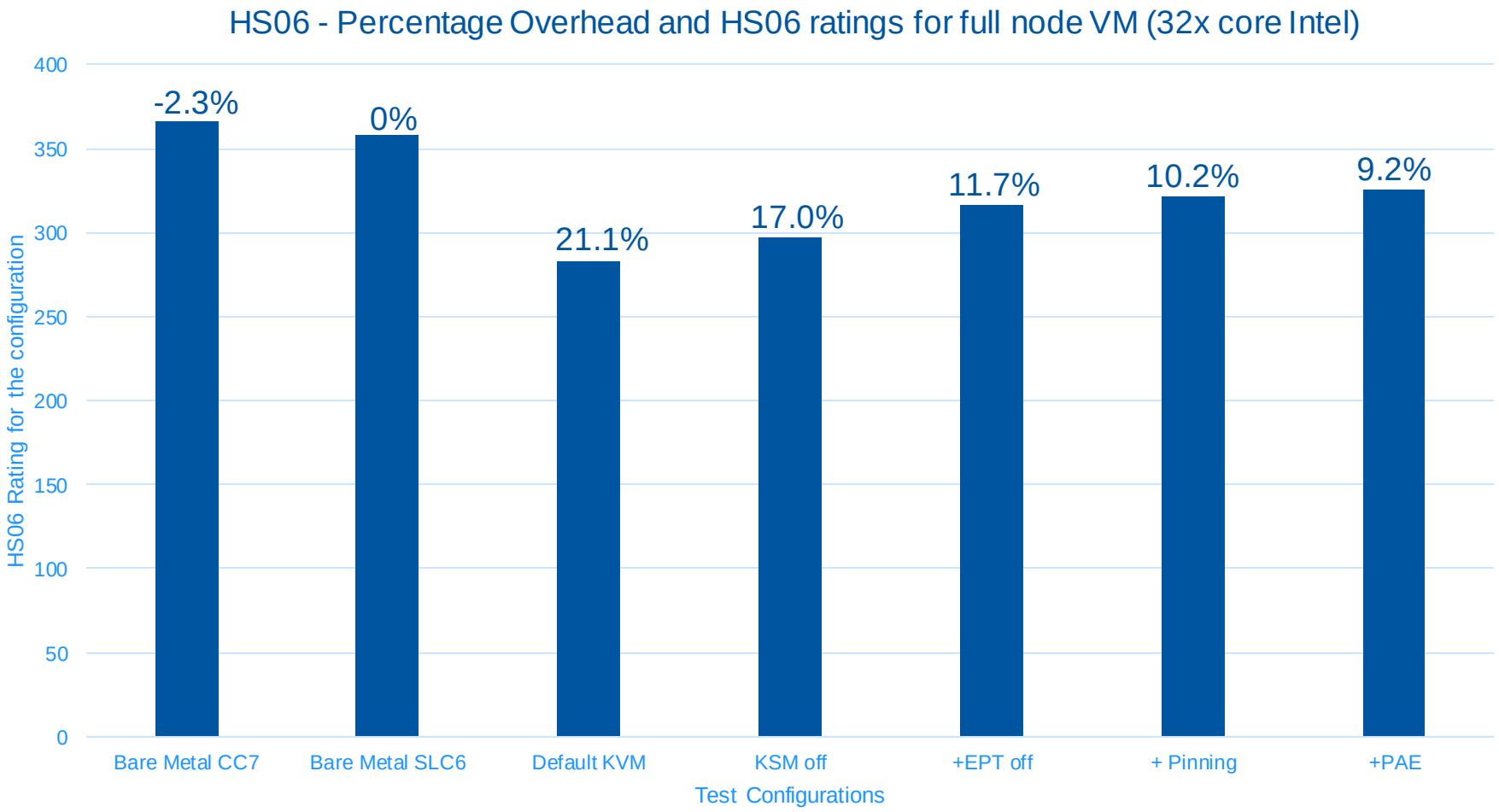
HS06 on virtual batch workers

Type 1: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz

HWDB HS06	VM Size (cores)	Per VM HS06	Total HS06	Overhead
357±16	4x 8	82.3±11	329	7.8%
	2x 16	150±5	300	16%
	1x 32	284±11	284	**20.4%**

- One hardware type
  - 30x 32-core Intel boxes
- Default settings, no tuning
  - Except for host pass-through

VM Tuning Potential



First Optimisations

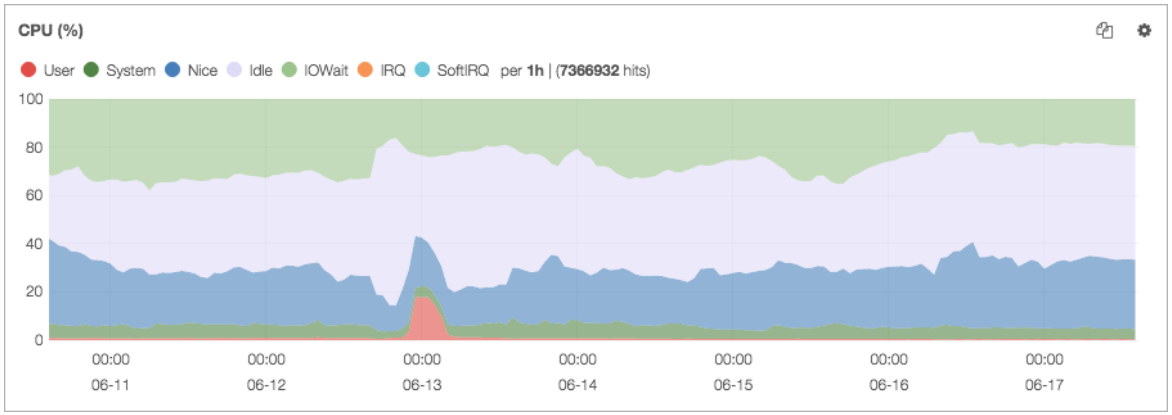
Type 1, before:

HWDB HS06	VM Size (cores)	Per VM HS06	Total HS06	Overhead
357±16	4x 8	82.3±11	329	7.8%
	2x 16	150±5	300	16%
	1x 32	284±11	284	20.4%

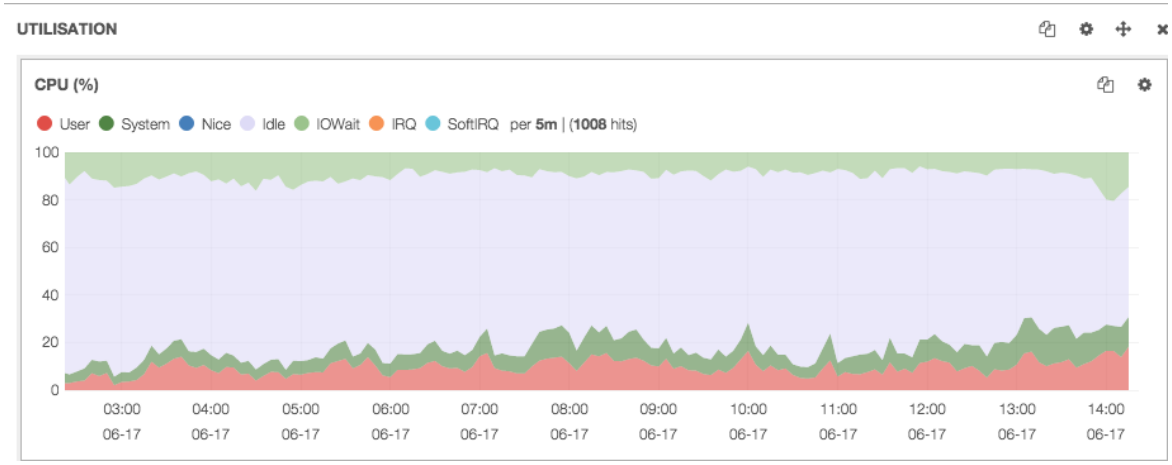
Type 1, after:

HWDB HS06	VM Size (cores)	Per VM HS06	Total HS06	Overhead	Overhead Reduction
357±16	4x 8	87±11	348	2.5%	68%
	2x 16	163.5±1	327	8.4%	52%
	1x 32	311±1	311	12.9%	37%

Swapping Hosts Detour (1/4)



ATLAS T0 batch VMs show IOwait 20-30%



Hypervisor in IOwait as well … swapping!

Swapping Hosts Detour (2/4)

- Why do the servers swap?
  - 2x 31GB should leave 2GB for the hypervisor
  - 1.5GB used by “something”, even on empty hypervisor

```
[root@p05792986e53504 ~]# echo 3 > /proc/sys/vm/drop_caches && sync && free -wm
              total        used        free      shared     buffers       cache   available
Mem:          64160        1315       61272         216           0        1571       61217
Swap:             0           0           0
```

```
[root@p05792986e53504 ~]# grep SUnreclaim /proc/meminfo
SUnreclaim:      1292408 kB
```

Swapping Hosts Detour (3/4)

- Where's my memory?

root@p05792986e53504:/proc

```
Active / Total Objects (% used)    : 18586976 / 20117301 (92.4%)
Active / Total Slabs (% used)      : 304473 / 304473 (100.0%)
Active / Total Caches (% used)     : 78 / 105 (74.3%)
Active / Total Size (% used)       : 1253908.57K / 1365327.83K (91.8%)
Minimum / Average / Maximum Object : 0.01K / 0.07K / 8.00K

  OBJS ACTIVE  USE OBJ SIZE  SLABS OBJ/SLAB CACHE SIZE NAME
16575360 15277296  92%    0.06K 258990       64   1035960K kmalloc-64
2287265 2111233  92%    0.05K  26909       85    107636K shared_policy_node
177152 176804  99%    0.03K   1384      128      5536K kmalloc-32
135184 135016  99%    0.07K   2414       56      9656K Acpi-ParseExt
131580 130730  99%    0.02K    774      170      3096K fsnotify_event_holder
115712 113197  97%    0.01K    226      512       904K kmalloc-8
108800 105124  96%    0.02K    425      256      1700K kmalloc-16
 65646  65399  99%    0.19K   1563       42     12504K dentry
 59058  58939  99%    0.08K   1158       51      4632K selinux_inode_security
 58644  58133  99%    0.11K   1629       36      6516K sysfs_dir_cache
 41734  13153  31%    0.57K    768       56     24576K radix_tree_node
 28672  28537  99%    1.00K    896       32     28672K xfs_inode
 26368  22520  85%    0.12K    422       64      3376K kmalloc-128
 24990  23701  94%    0.19K    595       42      4760K kmalloc-192
```

Swapping Hosts Detour (4/4)

Using systemtap to find slab cache user

```
#! /usr/bin/env stap

global slabs

probe vm.kmem_cache_alloc {
  slabs [execname(), bytes_req, call_site, caller_function]<<<1
}

probe timer.ms(1000)
{
  dummy = "";
  foreach ([name, bytes, site, call_function] in slabs) {
    if (dummy != name)
    printf("\nProcess:%s\n", name);
    printf("\nCall Site:%x - Call  Function:%s\n", site, call_function);
    printf("Slab_size:%d\tCount:%d\n", bytes,
                        @count(slabs[name, bytes, site, call_function]));
    dummy = name;
  }
  delete slabs
  printf("\n----------------------------\n\n")
}
```

```
…
Slab_size:64        Count:6
Slab_size:64        Count:8
Slab_size:64        Count:1
Slab_size:64        Count:1
Process: ksmd
Call Site: …
Slab_size:64        Count:518400
Slab_size:64        Count:550
Slab_size:64        Count:3
Slab_size:64        Count:1
Slab_size:64        Count:3
Slab_size:64        Count:5
Slab_size:64        Count:10
Slab_size:64        Count:9
Slab_size:64        Count:17
Slab_size:64        Count:13
Process: ksmd
Call Site: …
Slab_size:64        Count:697500
Slab_size:64        Count:16
Slab_size:64        Count:3
Slab_size:64        Count:24
Slab_size:64        Count:280
Process: ksmd
Call Site: …
Slab_size:64        Count:696386
Slab_size:64        Count:1
Slab_size:648       Count:2
Slab_size:648       Count:2
Slab_size:64        Count:1
…
```

This memory is not freed when KSM is disabled!

Hardware Type Differences

Type 1 optimised:

Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz

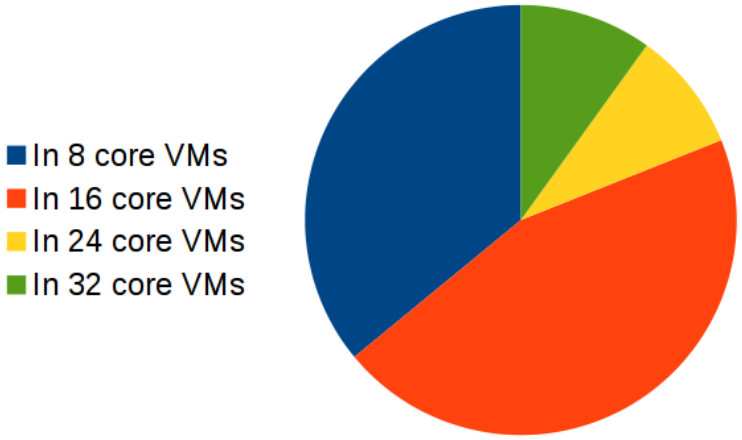
HWDB HS06	VM Size (cores)	Per VM HS06	Total HS06	Overhead	Overhead Reduction
357±16	4x 8	87±11	348	2.5%	68%
	2x 16	163.5±1	327	8.4%	52%
	1x 32	311±1	311	12.9%	37%

Type 2 optimised:

Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz

HWDB HS06	VM Size (cores)	Per VM HS06	Total HS06	Overhead	Overhead Reduction
344±1	4x 8	84±7	336	2.3%	65.7%
	2x 16	161±2	322	6.4%	22%
	1x 32	279±4	279	19%	13.6%

Core Distribution & Effective Impact



Take into account the core distribution over VM flavors to determine real loss (7-8%)

Small VMs (8 or 16 cores)	Large VMs (24 or 32 cores)
Closer to bare metal performance	Better memory flexibility
Closer to LSF limitations (less of a problem after instance split)	Higher performance penalty

General virtualization issue?

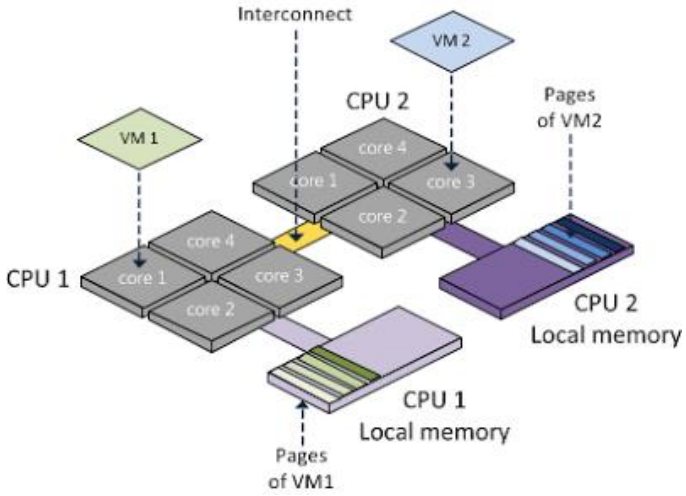
- Crosscheck w/ SLC6 VMs on Hyper-V
  - 0.8% HS06 loss on 4x 8-core
  - 3.3% HS06 loss on 1x 32-core SLC6 VM

- No general virtualization overhead issue!
  - Rather a feature or configuration issue

- What's the difference between the VMs on Hyper-V and the ones on KVM?

NUMA

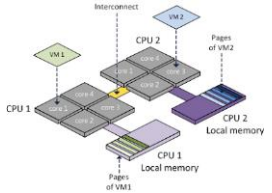
- Hyper-V VMs “see” underlying NUMA architecture



- Hyper-V VMs have vCPUs pinned to physical NUMA nodes
  - Pinned to sets that correspond to physical NUMA nodes

- In OpenStack, wider support for this comes with the release we currently prepare to deploy (Kilo)
  - Juno has some NUMA support as well
  - Consciously not configured, then disabled after ATLAS T0 incident (aka the “50% overhead” problem …)

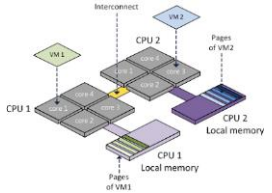
NUMA in the lab



- Standard SLC6 VMs on CC7 hypervisors …
  - "Production" tuning plus extras (e.g. KSM off, numad off)
  - Manual VM configuration: 4 NUMA nodes, 1-on-1 pinning

- … reduce the loss to ~3% of the bare metal performance as listed in the hardware DB
  - 16-core and 8-core VMs even better (within error bar)

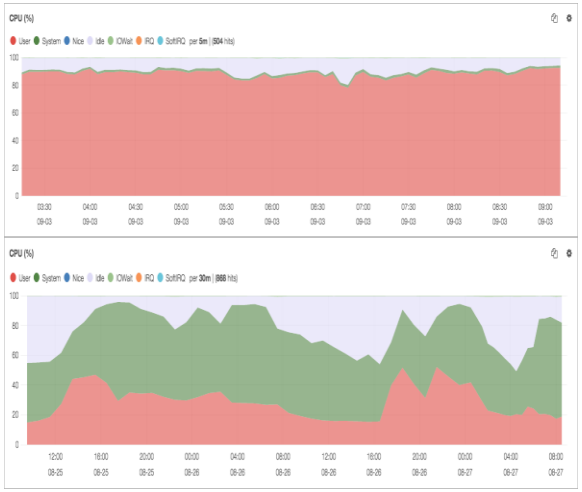
NUMA: from lab to prod



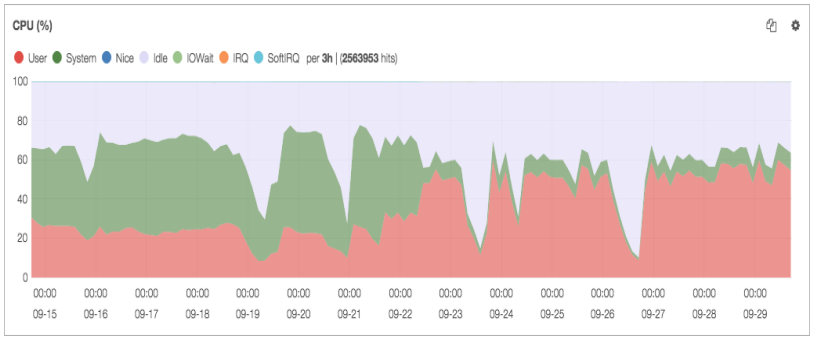
- Reproducible on NUMA-aware batch VMs!
  - ~3% with KSM off, system services off
  - ~4.5% with KSM off, system services on
  - ~5% with KSM on, system services on
- Need to assess impact of KSM in more detail
  - Needed in production for now
  - Tuning possible (NUMA-aware KSM, scanning frequencies)
- System services overhead around 1-2%
  - Hardware DB value measured without services
  - We need more statistics
  - Investigations on virtual and physical hardware ongoing

Issue w/ extremely slow nodes

- Small fraction of jobs 10x slower
  - Painful for the experiment workflows
  - VMs look OK, actually pretty good
  - Hosts: 30-50% system load, >100k IRQ/s (mostly TLB shoot-downs)



- Load attributed to qemu-kvm
  - ‘perf top’: 90% in _raw_spin_lock
  - ‘systemtap’: paging64_page_fault and kvm_mmu_pte* …



→ “EPT off” side-effect!

(not seen by HS06 nor in our pre-deployment tests)

EPT revisited

- Hardware support for management of guest page tables
  - Available in all modern processors (SLAT, NPT, RVI)
  - Avoid shadow page tables, i.e. double maintenance in software ("slow job" effect: s/w needed to do things that could be done in h/w)
- HS06: TLB misses can be more expensive with EPT on than with shadow page tables
  - Each page request in the guest translates into multiple page table walks in the host
  - NPT on/off on AMD made no difference! page table walk cache?

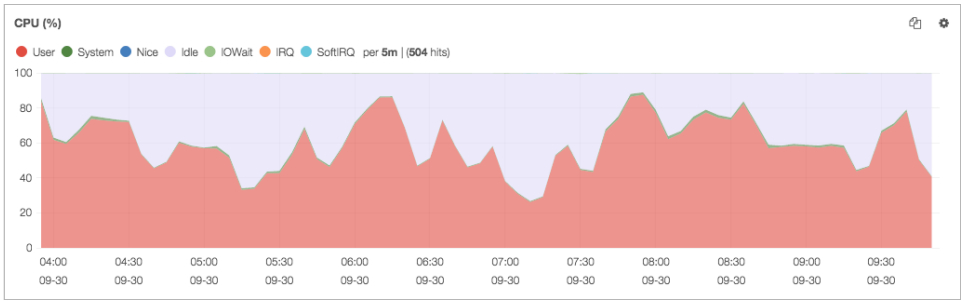
Back to the drawing board!

- Needed to combine optimizations with EPT on
- Huge pages a way out?
  - Idea: reduce the number of pages to be handled, increase hit ratio
- 1GB huge pages
  - Best HS06 results (with EPT on)
  - But: NUMA node distribution uneven when close to hardware limits (kernel memory allocation prevents reservation of contiguous 1GB pages)
- 2MB huge pages
  - Also one of the default sizes
  - Performance loss around 5% compared to bare metal on batch VMs

The “Kilo-1” configuration

- NUMA + Pinning
  - 1-to-1 vs. 1-to-N no difference
- 2MB huge pages
  - 1GB slightly better
- EPT on
  - EPT off still better in HS06

VM sizes (cores)	Before	After
4x 8	7.8%	3.3% (batch WN)
2x 16	16%	4.6% (batch WN)
1x 24	20%	5.0% (batch WN)
1x 32	20.4%	3-6% (bare SLC6 … batch WN)

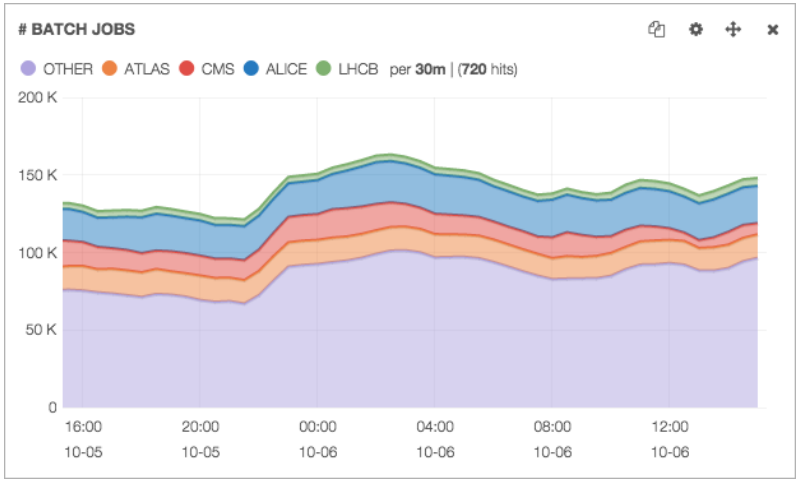


ATLAS T0 host with batch VM running the new config: throughput for recon jobs 20% higher!

OpenStack Kilo will fully support our desired configuration!

Pre-deployment testing

- The first optimizations were of course tested before being rolled out
  - ~30 batch nodes for some weeks
- The slow-job issue was not detected!
  - ATLAS: work load on separate cell
  - LHCb: issue affected “small fraction of a small fraction”
- A small fraction can cause a lot of trouble …



Summary & Conclusion

- We managed to reduce the virtualization HS06 overhead to a few percent compared to bare metal
  - On full node VMs!
  - NUMA + pinning + huge pages + EPT
- Pre-deployment testing difficult
  - EPT side-effects undetected despite weeks of testing
  - Similar issue with KSM and un-reclaimable kernel memory
- Continuous benchmarking needed!
  - See Sean's talk


CERN