GEOM/CAM scalability

1.000.000 IOPS on FreeBSD

Alexander Motin <mav@FreeBSD.org>
iXsystems, Inc.

2013-05 FreeBSD Developer Summit

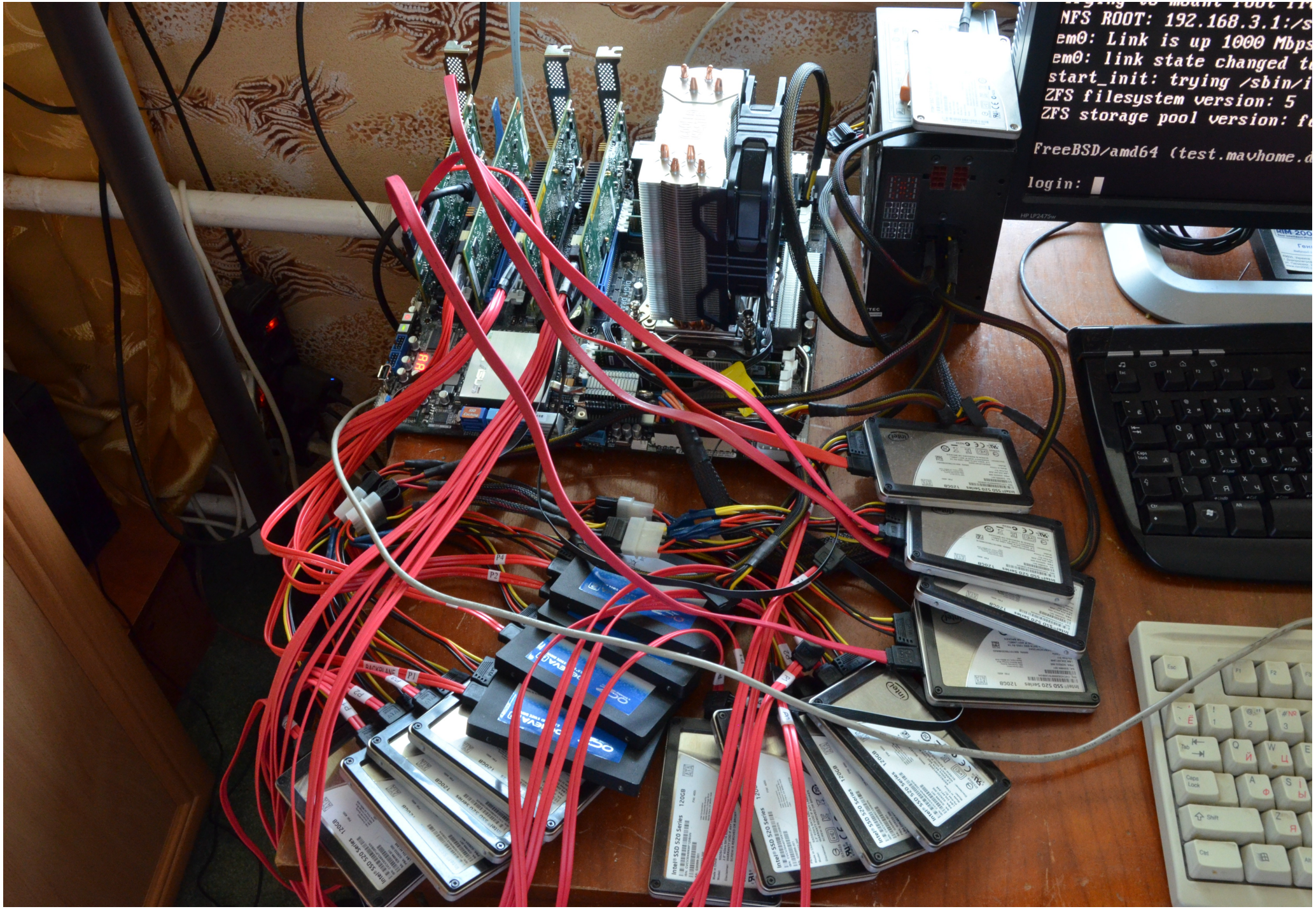
Goal: check / improve FreeBSD disk subsystem scalability, measured in number of I/Os per second.

Test rig:

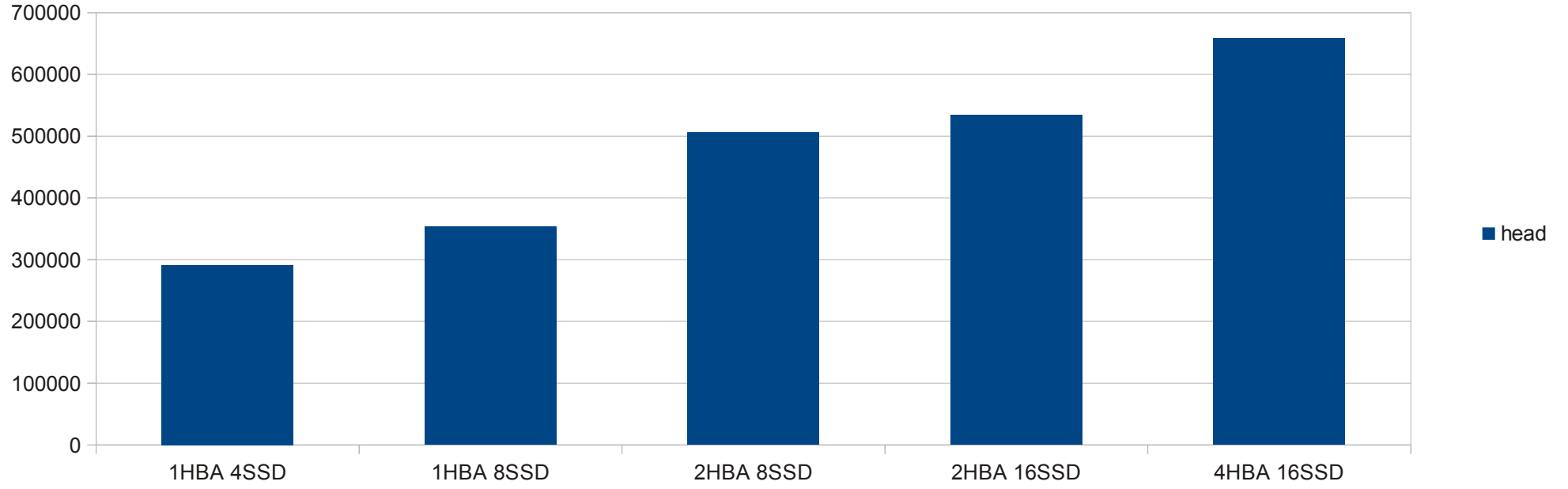
- Intel Core i7-3930K CPU @ 3.20GHz (6x2 cores)
- Asus P9X79 WS motherboard (4x PCIe x8 slots)
- 4x 8GB DDR3 RAM
- 4x LSI SAS HBA (2x SAS2308 + 2x SAS2008)
- 16x SSD (12x Intel 520 + 4x OCZ Deneva 2)

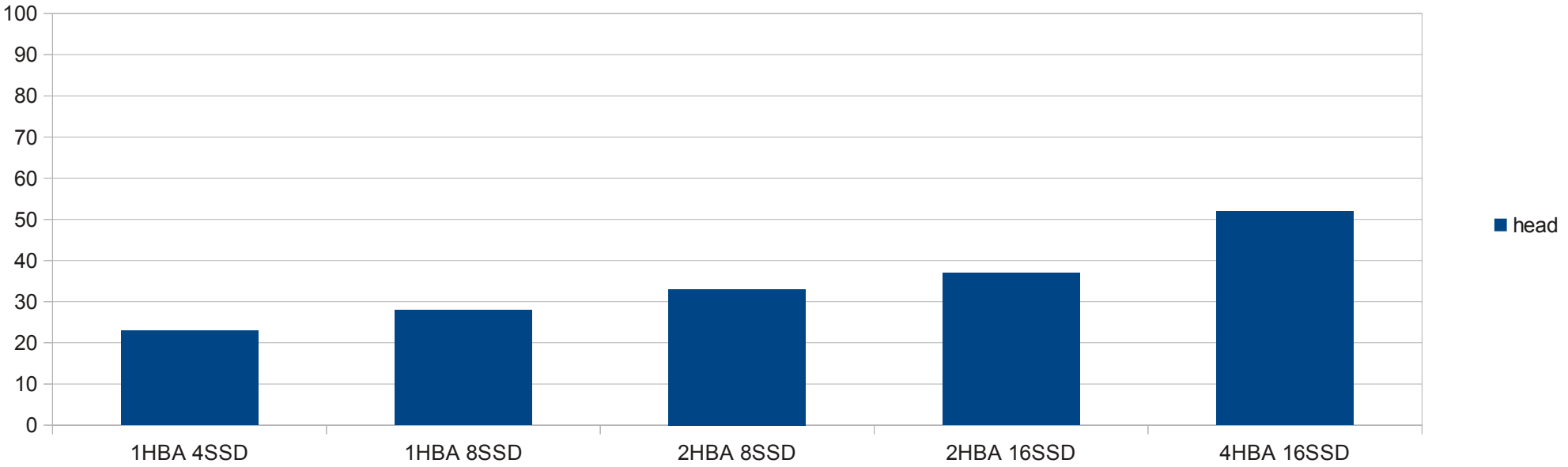
Test load: `dd if=/dev/daX of=/dev/null bs=512 &` – 16 instances per disk

The test rig:


NFS ROOT: 192.168.3.1:/s
em0: Link is up 1000 Mbps
em0: link state changed t
start_init: trying /sbin/i
ZFS filesystem version: 5
ZFS storage pool version: f
FreeBSD/amd64 (test.mavhome.d
login:

IOPS & CPU load with FreeBSD head of May 2013





`top -SHz` for the case of 4HBA 16SSD

```
CPU:  3.7% user,  0.0% nice, 36.7% system, 12.0% interrupt, 47.7% idle
  PID USERNAME   PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
   13 root        -8    -     0K    48K CPU6    6  14:06 88.96% geom{g_down}
   13 root        -8    -     0K    48K CPU2    2  10:11 77.78% geom{g_up}
   12 root       -68    -     0K   528K CPU3    3   7:59 53.08% intr{swi2: camb
   12 root       -88    -     0K   528K WAIT    9   0:20 15.67% intr{irq266: mp
   12 root       -88    -     0K   528K WAIT    8   1:58 13.38% intr{irq265: mp
   12 root       -88    -     0K   528K WAIT   10   0:21 13.38% intr{irq267: mp
   12 root       -88    -     0K   528K WAIT    5   5:16 12.79% intr{irq264: mp
```

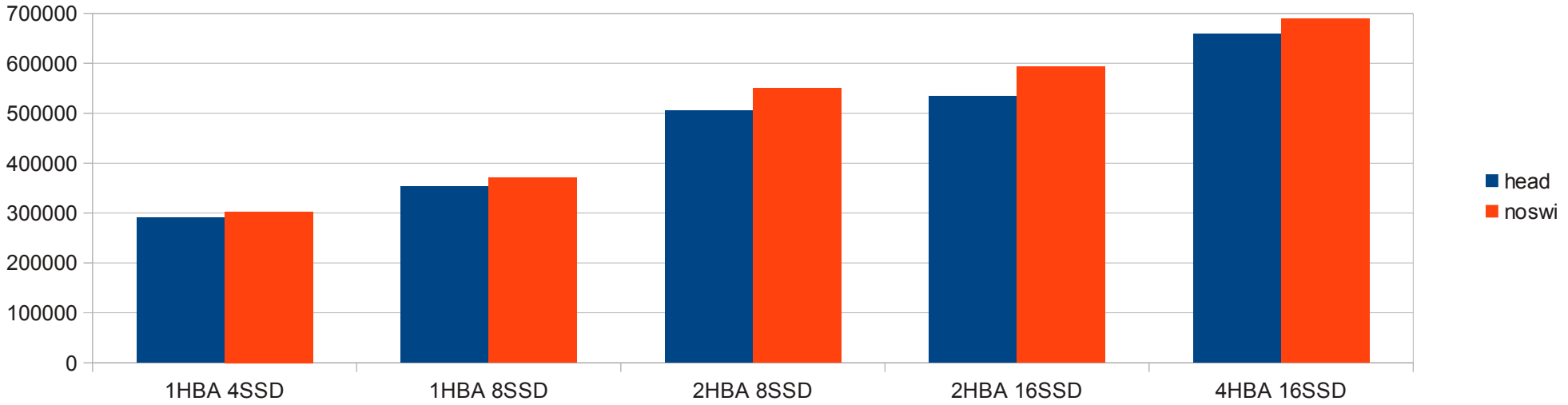
`pmcstat -t` for the case of 4HBA 16SSD

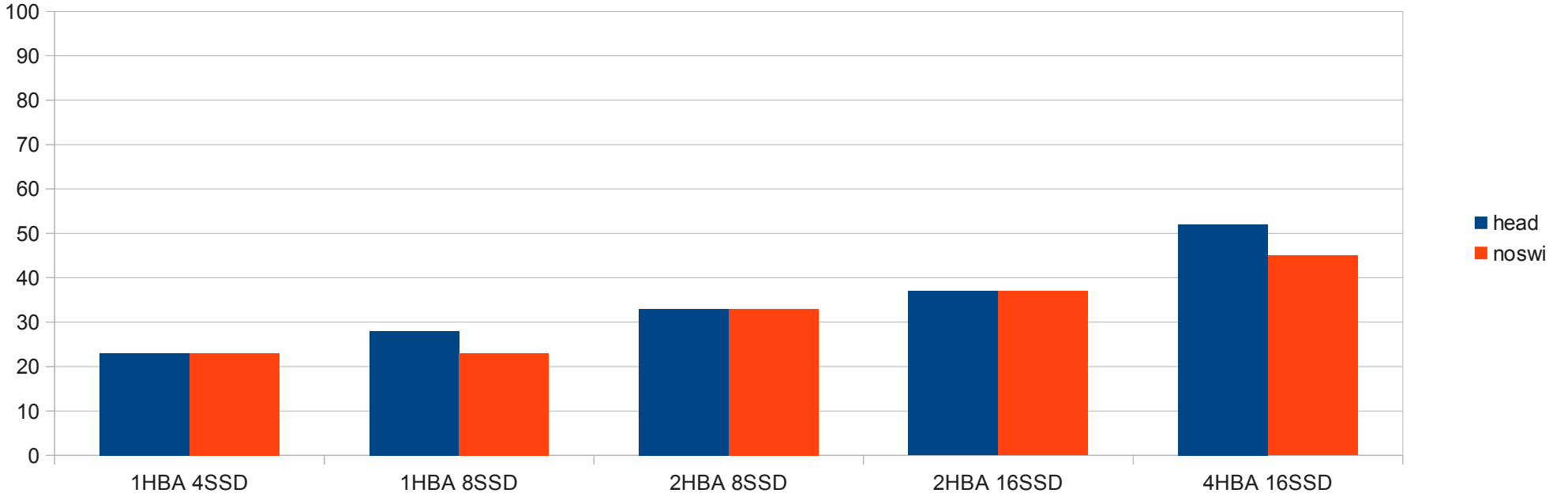
```
%SAMP IMAGE      FUNCTION             CALLERS
 13.2 kernel     cpu_search           cpu_search:11.0 sched_idletd:1.9
  6.7 kernel     cpu_idle             sched_idletd
  6.2 kernel     __mtx_lock_sleep     dastrategy:2.9 mps_intr_msi:1.4 ...
  4.7 kernel     cpu_switch           mi_switch
  3.2 kernel     thread_lock_flags_   sleepq_add:0.9 sched_idletd:0.8 ...
```

Results:

- LSI SAS2008 can handle 85K IOPS per device, but only 320K IOPS per HBA (as documented)
- LSI SAS2308 – 85K/450K (less then documented)
- Onboard Intel AHCI with 6 SATA ports is faster on this load then more expensive SAS – 100K+/500K+
- Several shared threads (g_up, g_down and CAM SWI) create SMP scalability bottleneck
- The OS bottleneck for this hardware is around 650K IOPS now. It is not easy, but possible to reach.
- More separate HBAs is always better because of separate locks and interrupt vectors (ahci(4) uses one lock per port, while mps(4) – one lock per HBA)

Step 1: avoid switch to CAM SWI by using xpt_batch_start() and xpt_batch_done() in mps(4)

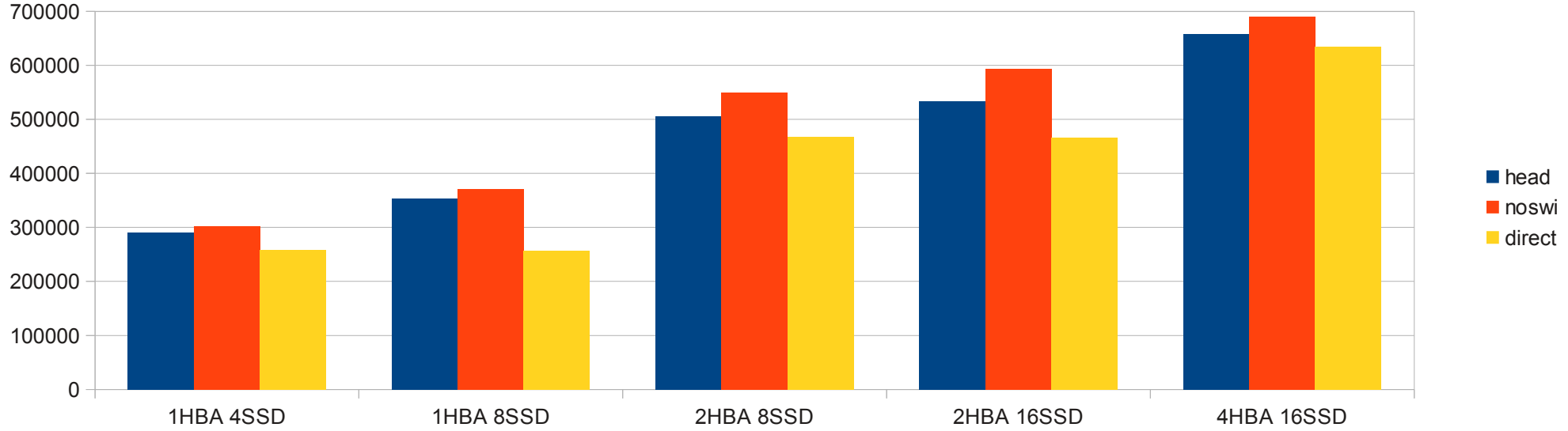


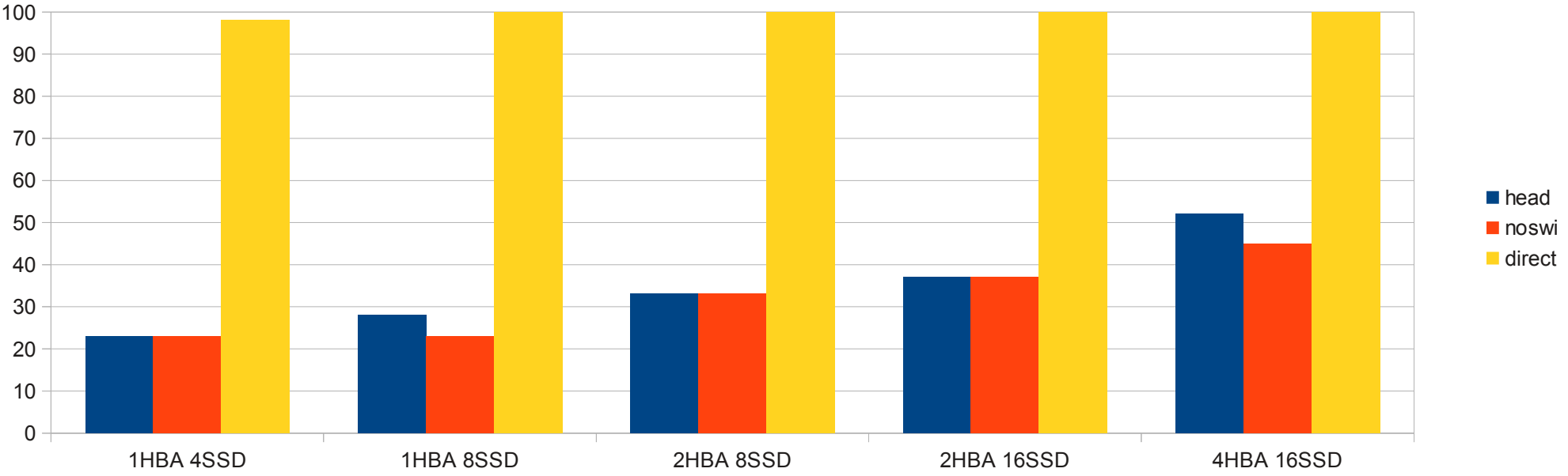


Result: Some scalability improvement, but g_up/g_down threads are even more congested

```
CPU:  4.4% user,  0.0% nice, 32.1% system,  7.6% interrupt, 55.8% idle
  PID USERNAME   PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
   13 root        -8    -     0K    48K CPU1    1   1:10 94.09% geom{g_down}
   13 root        -8    -     0K    48K CPU5    5   1:02 82.18% geom{g_up}
   12 root       -88    -     0K   528K CPU7    7   0:17 24.76% intr{irq266: mp
   12 root       -88    -     0K   528K CPU10  10   0:17 24.27% intr{irq267: mp
   12 root       -88    -     0K   528K CPU9    9   0:16 21.68% intr{irq264: mp
   12 root       -88    -     0K   528K WAIT    8   0:16 19.29% intr{irq265: mp
```

Step 2: avoid switch to g_up and g_down threads by quick and dirty GEOM hack



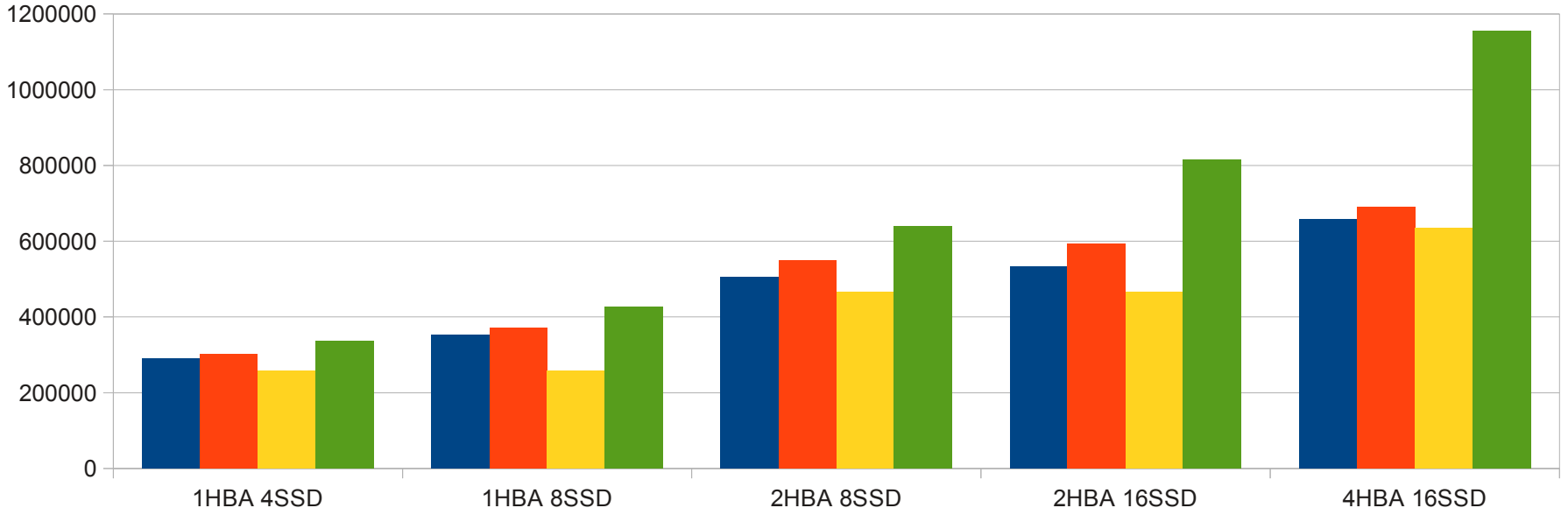


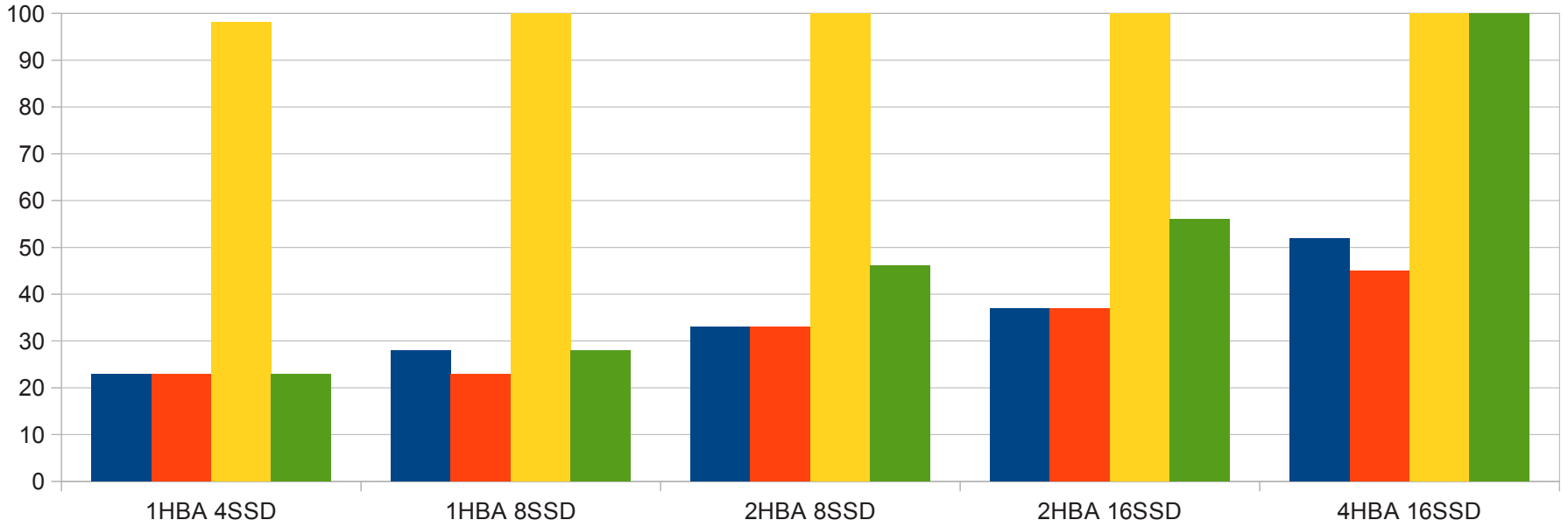
Result: Huge lock congestion on CAM SIM locks. Heavy adaptive lock spinning by multiple calling threads burned CPU and killed the performance.

```
CPU:  3.5% user,  0.0% nice, 78.3% system, 18.3% interrupt,  0.0% idle
  PID USERNAME   PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
   12 root       -88    -     0K   528K WAIT    8   3:27 73.68% intr{irq265: mp
   12 root       -88    -     0K   528K WAIT    9   0:27 70.07% intr{irq266: mp
   12 root       -88    -     0K   528K CPU7    7   5:38 53.56% intr{irq264: mp
   12 root       -88    -     0K   528K WAIT   10   0:18 38.57% intr{irq267: mp
 2053 root        27    0 12152K  1912K CPU4    4   0:03  9.38% dd
 2043 root        28    0 12152K  1912K CPU10  10   0:03  8.79% dd
 2055 root        27    0 12152K  1912K RUN    11   0:03  8.79% dd


%SAMP IMAGE      FUNCTION             CALLERS
 44.9 kernel     __mtx_lock_sleep     dastrategy:41.1 mps_intr_msi:3.0
  4.7 kernel     cpu_search           cpu_search
  3.4 kernel     _mtx_lock_spin_cooki turnstile_trywait
  2.4 kernel     cpu_switch           mi_switch
  1.5 kernel     dastrategy           g_disk_start
```

Step 3: Reduce CAM SIM lock congestion and adaptive spinning by reworking CAM locking





Result: Almost double performance boost in case of 4HBA 16SSD – 1.15M IOPS!

```
CPU:  8.6% user,  0.0% nice, 62.7% system, 27.8% interrupt,  0.9% idle
  PID USERNAME   PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
   12 root       -88    -     0K   528K CPU2    2   0:48 98.97% intr{irq267: mp
   12 root       -88    -     0K   528K CPU7    7   0:46 88.28% intr{irq264: mp
   12 root       -88    -     0K   528K CPU1    1   0:44 84.86% intr{irq265: mp
   12 root       -88    -     0K   528K WAIT    9   0:43 83.69% intr{irq266: mp
 1003 root        22    0 12152K  1912K physrd  0   0:02  3.27% dd
  994 root        22    0 12152K  1912K RUN     8   0:02  3.17% dd

%SAMP IMAGE      FUNCTION             CALLERS
  8.8 kernel     _mtx_lock_spin_cooki turnstile_trywait:7.7 sched_add:0.6
  8.4 kernel     cpu_search           cpu_search:6.6 sched_pickcpu:1.7
  4.8 kernel     __mtx_lock_sleep     xpt_run_devq
  3.4 kernel     xpt_run_devq         xpt_action_default
  3.3 kernel     cpu_switch           mi_switch
```

Much work still required to implement it properly...