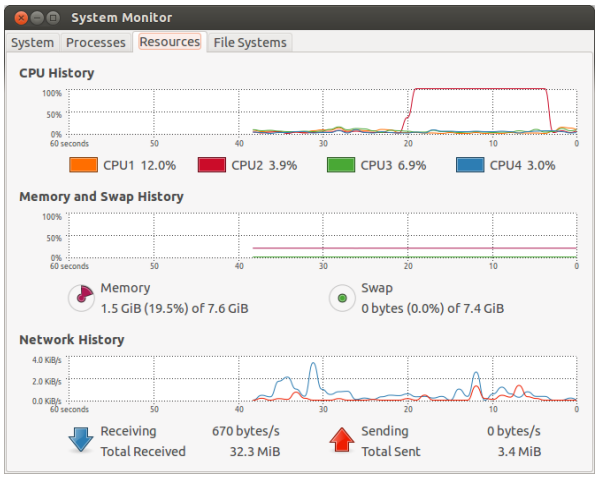
CAASTRO
ARC CENTRE OF EXCELLENCE
FOR ALL-SKY ASTROPHYSICS

Understanding Resource Utilisation of Computing Tasks

Jay Banyer
VivCourt Trading
(ex CAASTRO/USyd)

Credit: batterystop.com.au

VivCourt
VIVIENNE COURT TRADING



CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

Motivation

- Astronomical datasets are getting (much) larger with new telescopes and backends.
- Computer hardware is not keeping up (unless you go parallel).
- Tasks are going to take longer to run.
- Some tasks will become impossible on a standard desktop.

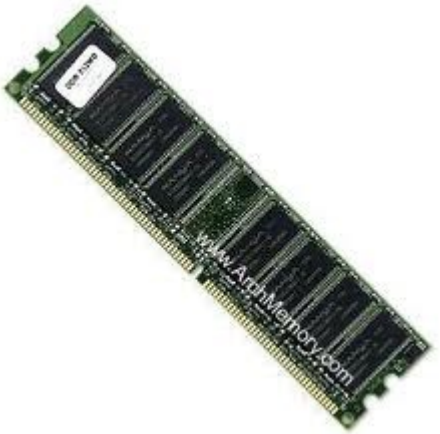
- This talk is about monitoring and speeding up tasks on standard equipment.
- Super computers and GPUs for another time!

CAASTRO
ARC CENTRE OF EXCELLENCE
FOR ALL-SKY ASTROPHYSICS

Overview

- Basic theory and monitoring tools for:
  - CPU
  - RAM
  - Disk
- Finding the bottleneck.
- How to complete your task faster.

Credit: sharkyextreme.com



Credit: clickbd.com

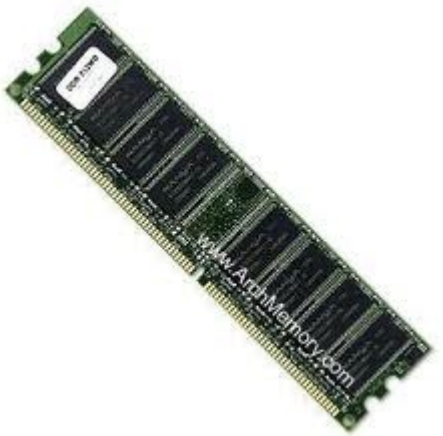
Credit: slashgear.com

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

Overview

- Basic theory and monitoring tools for:
  - CPU
  - RAM
  - Disk
- Finding the bottleneck.
- How to complete your task faster.

Credit: sharkyextreme.com



Credit: clickbd.com

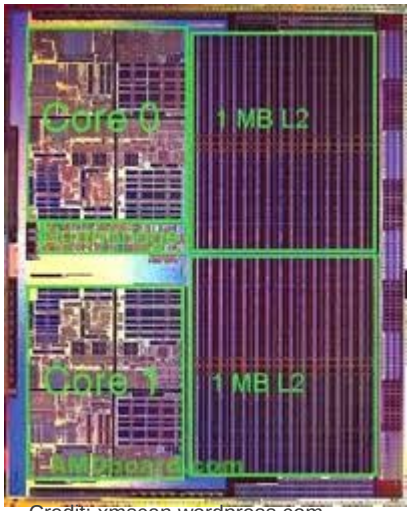

Credit: slashgear.com

CPU – sockets and cores

- All modern computers have multi-core CPUs. Even your iPhone (4) has multiple cores!
- Most laptops and desktops have a single CPU with 2, 4 or 6 cores.
- Many servers have multiple CPUs (aka sockets), each with 2, 4, 6 or 8 cores.
- For monitoring purposes the total number of cores matters and the number of CPUs does not.

Credit: sharkyextreme.com

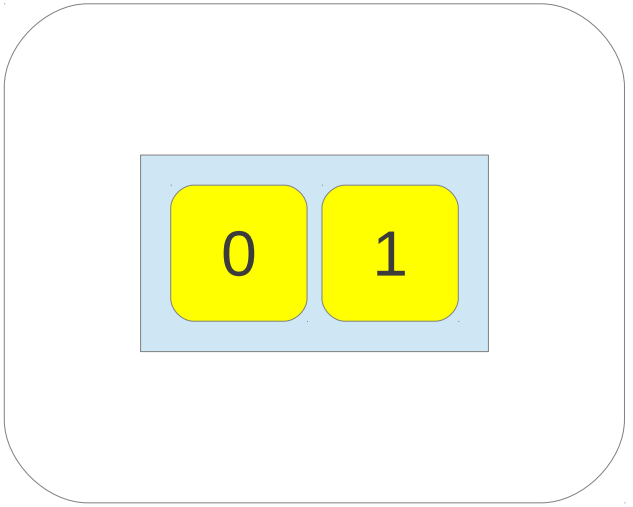
CPU with 2 cores



Credit: xmscan.wordpress.com

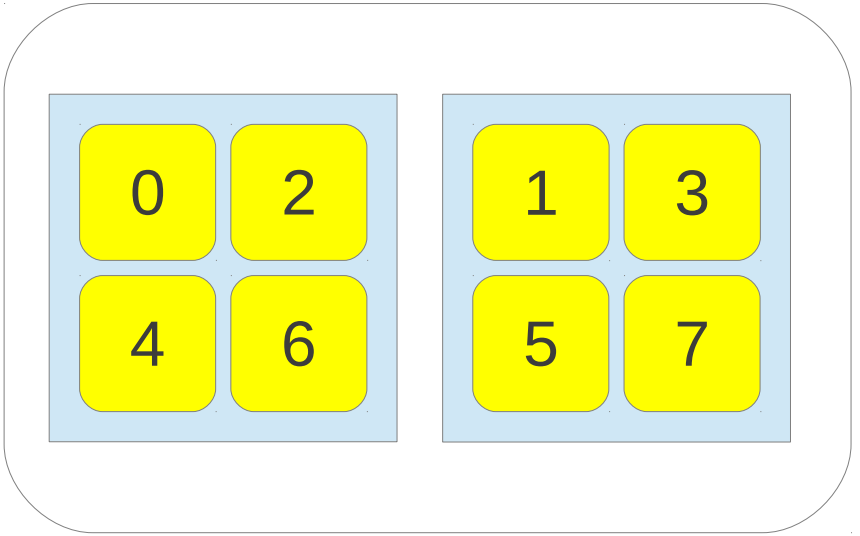
CPU – sockets and cores

Credit: batterystop.com.au



1 CPU
2 Cores

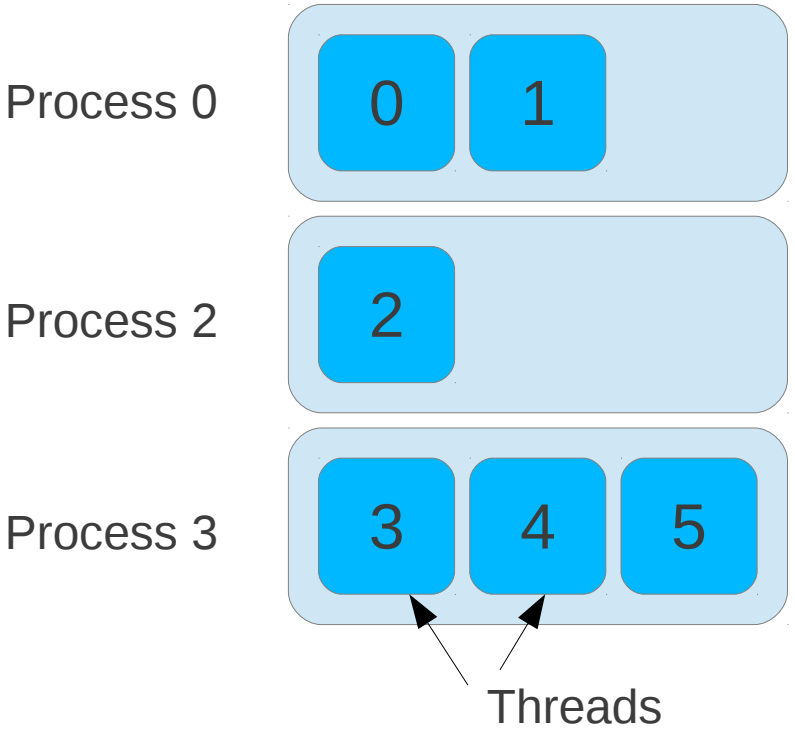
Credit: inetgiant.com



2 CPUs
8 Cores

Processes and Threads

- Software executes in a *thread*.
- A *process* contains one or more threads.
- Threads (not processes) are assigned to CPU cores.
- Each core can run one thread at any one time.
- All threads in a process share the same memory.



- PID = Process ID
- TID = Thread ID
- Same pool of numbers.
- TID of first thread in process equals PID.

top

jay@jay-e4300: ~/Dropbox/talks/sysload

```
top - 07:22:52 up 14:57,  3 users,  load average: 0.68, 0.27, 0.13
Tasks: 235 total,   2 running, 233 sleeping,   0 stopped,   0 zombie
Cpu(s): 51.7%us,  0.3%sy,  0.0%ni, 47.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   4001208k total,  1543700k used,  2457508k free,    69956k buffers
Swap:  4143100k total,        0k used,  4143100k free,   779024k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3981 jay       20   0  6312  380  296 R  100  0.0   0:55.96 burn
 1842 jay       20   0 1478m  88m  29m S    1  2.3   0:43.04 compiz
 2105 jay       20   0  449m  23m  11m S    1  0.6   0:26.32 unity-panel-ser
 1096 root      20   0  188m  25m 7816 S    0  0.6   0:38.93 Xorg
 1862 jay       20   0  812m  35m  18m S    0  0.9   0:03.87 nautilus
 1869 jay       20   0  416m  12m 8788 S    0  0.3   0:08.60 indicator-multi
 1876 jay       20   0  516m  70m  30m S    0  1.8   0:03.16 chromium-browse
 2108 jay       20   0  681m 7044 3508 S    0  0.2   0:25.26 hud-service
 4088 jay       20   0 17468 1404  952 R    0  0.0   0:00.13 top
    1 root      20   0 24436 2408 1348 S    0  0.1   0:01.03 init
    2 root      20   0     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S    0  0.0   0:00.45 ksoftirqd/0
    6 root      RT   0     0    0    0 S    0  0.0   0:00.00 migration/0
    7 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/0
   13 root       0 -20     0    0    0 S    0  0.0   0:00.00 cpuset
   14 root       0 -20     0    0    0 S    0  0.0   0:00.00 khelper
   15 root      20   0     0    0    0 S    0  0.0   0:00.00 kdevtmpfs
```

top - CPU

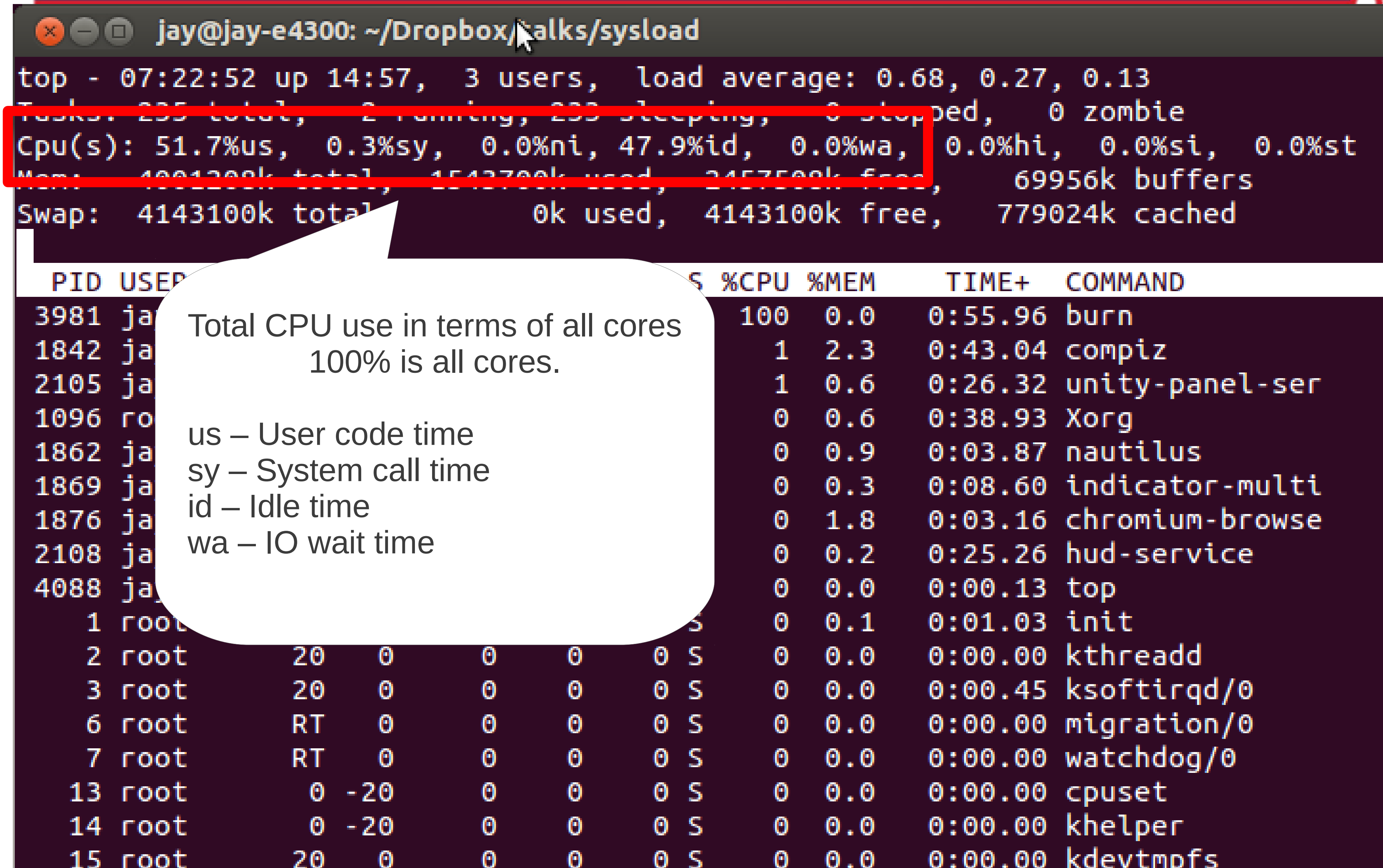
jay@jay-e4300: ~/Dropbox/talks/sysload

```
top - 07:22:52 up 14:57,  3 users,  load average: 0.68, 0.27, 0.13
Tasks: 235 total,   2 running, 233 sleeping,   0 stopped,   0 zombie
Cpu(s): 51.7%us,  0.3%sy,  0.0%ni, 47.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   4001208k total,  1543700k used,  2457508k free,    69956k buffers
Swap:  4143100k total,        0k used,  4143100k free,   779024k cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 3981 jay       20   0  6312  380  296 R   100  0.0   0:55.96 burn
 1842 jay       20   0 1478m  88m  29m S     1  2.3   0:43.04 compiz
 2105 jay       20   0  449m  23m  11m S     1  0.6   0:26.32 unity-panel-ser
 1096 root      20   0  188m  25m
 1862 jay       20   0  812m  35
 1869 jay       20   0  416m  12
 1876 jay       20   0  516m  70
 2108 jay       20   0  681m 704
 4088 jay       20   0 17468 1404
    1 root      20   0 24436 2408 1348 S     0  0.1   0:01.03 init
    2 root      20   0     0    0    0 S     0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S     0  0.0   0:00.45 ksoftirqd/0
    6 root      RT   0     0    0    0 S     0  0.0   0:00.00 migration/0
    7 root      RT   0     0    0    0 S     0  0.0   0:00.00 watchdog/0
   13 root       0 -20     0    0    0 S     0  0.0   0:00.00 cpuset
   14 root       0 -20     0    0    0 S     0  0.0   0:00.00 khelper
   15 root      20   0     0    0    0 S     0  0.0   0:00.00 kdevtmpfs
```

Process CPU use in terms of one core
100% is one core,
200% is two cores,
etc

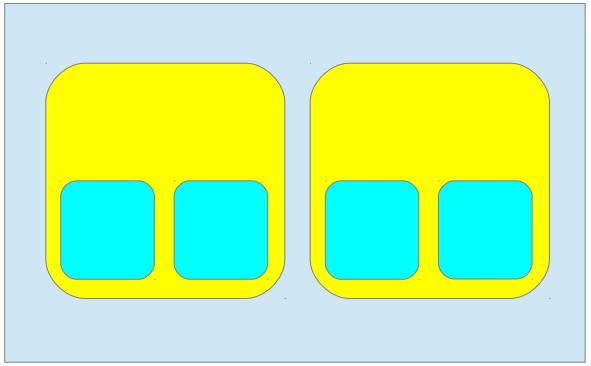
top - CPU



jay@jay-e4300: ~/Dropbox/talks/sysload
top - 07:22:52 up 14:57, 3 users, load average: 0.68, 0.27, 0.13
0 zombie
Cpu(s): 51.7%us, 0.3%sy, 0.0%ni, 47.9%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
69956k buffers
Swap: 4143100k total, 0k used, 4143100k free, 779024k cached
PID USER S %CPU %MEM TIME+ COMMAND
3981 100 0.0 0:55.96 burn
1842 1 2.3 0:43.04 compiz
2105 1 0.6 0:26.32 unity-panel-ser
1096 0 0.6 0:38.93 Xorg
1862 0 0.9 0:03.87 nautilus
1869 0 0.3 0:08.60 indicator-multi
1876 0 1.8 0:03.16 chromium-browse
2108 0 0.2 0:25.26 hud-service
4088 0 0.0 0:00.13 top
1 root 0 0.1 0:01.03 init
2 root 20 0 0 0 0 S 0 0.0 0:00.00 kthreadd
3 root 20 0 0 0 0 S 0 0.0 0:00.45 ksoftirqd/0
6 root RT 0 0 0 0 S 0 0.0 0:00.00 migration/0
7 root RT 0 0 0 0 S 0 0.0 0:00.00 watchdog/0
13 root 0 -20 0 0 0 S 0 0.0 0:00.00 cpuset
14 root 0 -20 0 0 0 S 0 0.0 0:00.00 khelper
15 root 20 0 0 0 0 S 0 0.0 0:00.00 kdevtmpfs
Total CPU use in terms of all cores
100% is all cores.
us – User code time
sy – System call time
id – Idle time
wa – IO wait time

CAASTRO
ARC CENTRE OF EXCELLENCE
FOR ALL-SKY ASTROPHYSICS

Hyper Threading



1 CPU
2 cores
4 *threads*

- Each true core can run two threads (sort of....).
- Each true core appears as two cores to Operating System, **top** etc.
- Present on most Intel CPUs, eg some Core i5/i7. Not Core 2.
- Intel claims 0 – 30% capacity increase depending on load.
- No benefit for single thread, but no real cost either.
- Typically disabled on HPC clusters.
- AMD's new *Bulldozer* architecture is even more confusing...

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

Am I Hyper Threaded?

```
jay@jay-e4300: ~
jay@jay-e4300:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    2
Socket(s):             1
```

lscpu

1 socket
2 [true] cores per socket
1 thread per core [not HyperThreaded]
2 CPUs [virtual/true cores]

```
jay@jay-e4300: ~
jay@jay-e4300:~$ cat /proc/cpuinfo | grep processor
processor       : 0
processor       : 1
jay@jay-e4300:~$ cat /proc/cpuinfo | grep cores
cpu cores       : 2
cpu cores       : 2
jay@jay-e4300:~$
jay@jay-e4300:~$
```

cat /proc/cpuinfo

2 processors [virtual cores]
2 [true] cpu cores
Infer HyperThreading is not present

Am I Hyper Threaded?

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

```
jay@marlin: ~
jay@marlin:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    2
Core(s) per socket:    2
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
```

lscpu

1 socket
2 [true] cores per socket
2 threads per core [HyperThreaded]
4 CPUs [virtual cores]

Mac: **sysctl hw**

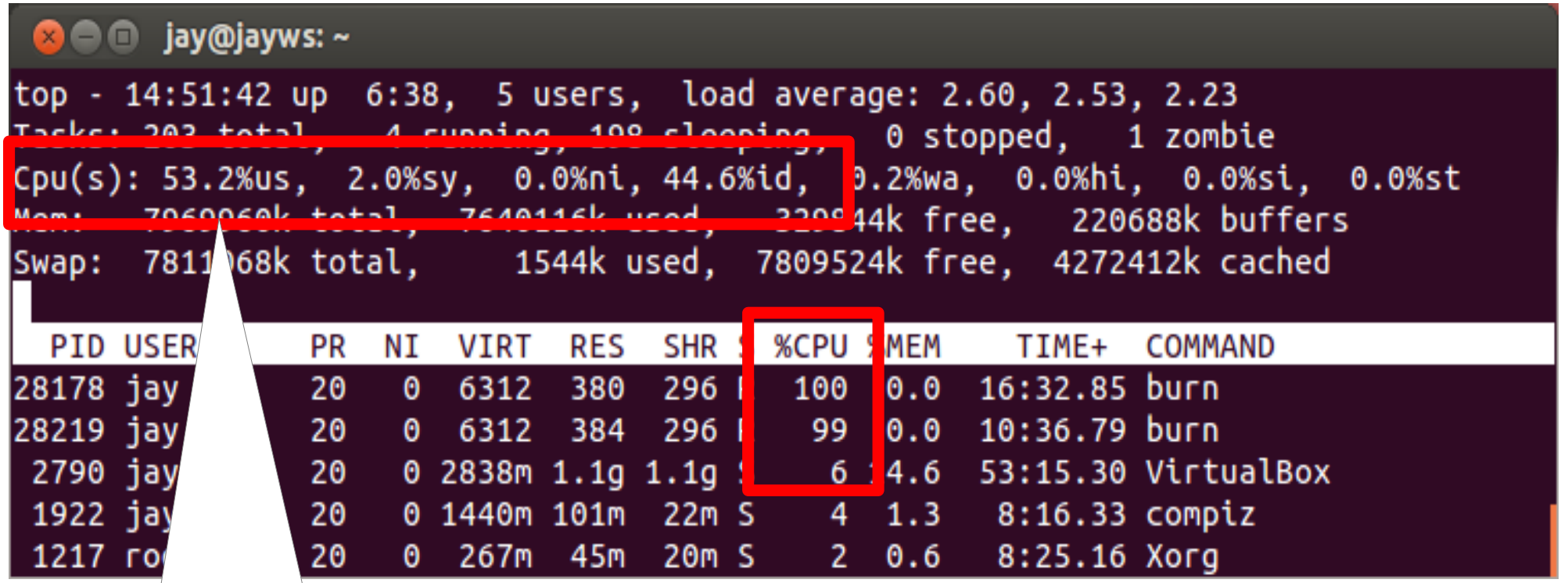
hw.physicalcpu_max: 2
hw.logicalcpu_max: 4
Infer HyperThreading is present

```
jay@marlin: ~
jay@marlin:~$ cat /proc/cpuinfo | grep processor
processor       : 0
processor       : 1
processor       : 2
processor       : 3
jay@marlin:~$ cat /proc/cpuinfo | grep cores
cpu cores       : 2
cpu cores       : 2
cpu cores       : 2
cpu cores       : 2
jay@marlin:~$
```

cat /proc/cpuinfo

4 processors [virtual cores]
2 [true] cpu cores
Infer HyperThreading is present

Hyper Threading - top

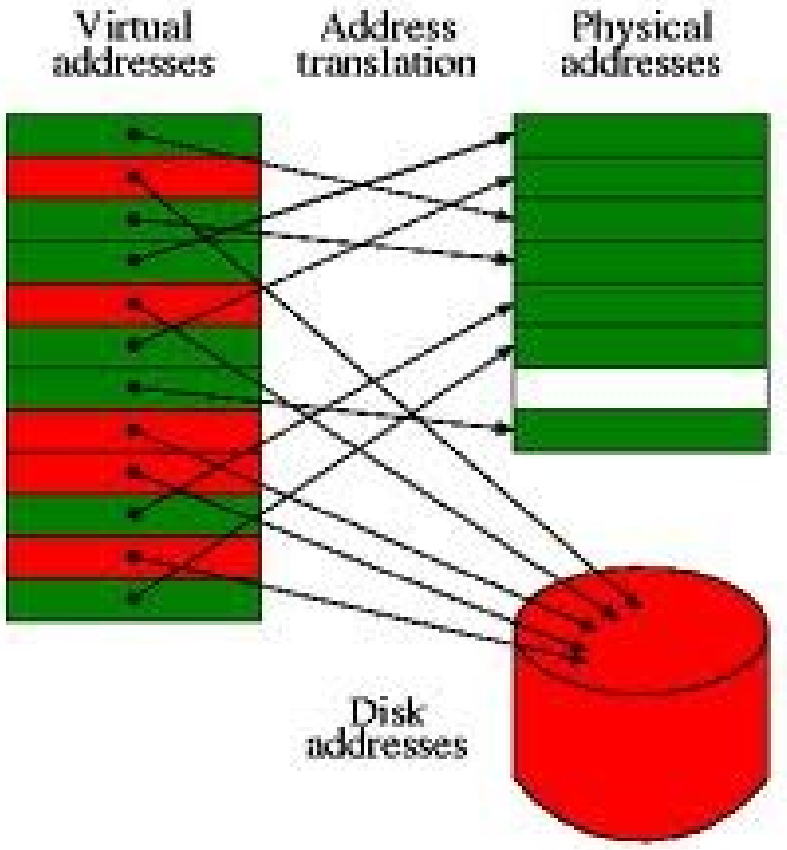


Top reports system is ~50% idle,
But not true! More like 0% idle...

- Machine has two true cores with Hyper Threading.
- Appears as four cores to OS.
- Two single-threaded processes each running at 100% CPU.

RAM - theory

- Each process has its own virtual memory address space.
- This memory is divided into blocks called pages.
- Each page is either:
  - In physical RAM (resident)
  - On disk (swap)
  - Nowhere (unused)
- Pages are swapped between disk and RAM as required – this is extremely slow and should be avoided.
- Excessive swapping is called “thrashing” and degrades system performace significantly. Indicates not enough RAM for workload.



Credit: brokenthorn.com

top - RAM

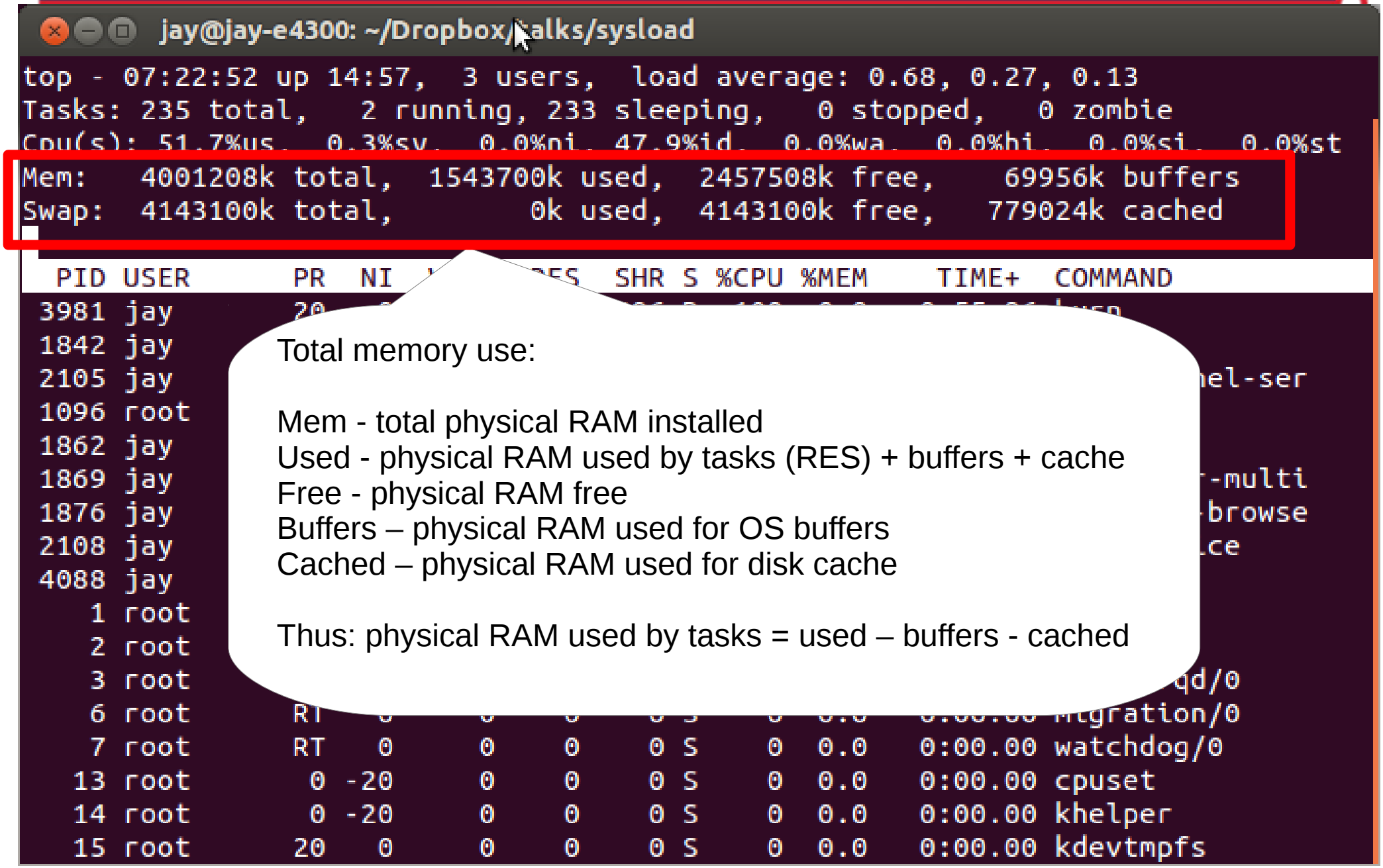
```
jay@jay-e4300: ~/Dropbox/talks/sysload
top - 07:22:52 up 14:57,  3 users,  load average: 0.68, 0.27, 0.13
Tasks: 235 total,   2 running, 233 sleeping,   0 stopped,   0 zombie
Cpu(s): 51.7%us,  0.3%sy,  0.0%ni, 47.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   4001208k total,  1543700k used,  2457508k free,    69956k buffers
Swap:  4143100k total,        0k used,  4143100k free,   779024k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3981 jay       20   0  6312  380  296 R  100  0.0   0:55.96 burn
 1842 jay       20   0 1478m  88m  29m S    1  2.3   0:43.04 compiz
 2105 jay       20   0  449m  23m  11m S    1  0.6   0:26.32 unity-panel-ser
 1096 root      20   0  188m  45m 7816 S    0  0.6   0:38.93 Xorg
 1862 jay       20   0                                0:03.87 nautilus
 1869 jay                                                    ndicator-multi
 1876 jay                                                    romium-browse
 2108 jay                                                    ud-service
 4088 jay                                                    pp
    1 root                                                   nit
    2 root                                                   threadd
    3 root                                                   softirqd/0
    6 root                                                   igration/0
    7 root      RT                                           watchdog/0
   13 root       0 -20     0    0    0 S    0  0.0   0:00.00 cpuset
   14 root       0 -20     0    0    0 S    0  0.0   0:00.00 khelper
   15 root      20   0     0    0    0 S    0  0.0   0:00.00 kdevtmpfs
```

Process memory use:

VIRT – total virtual memory used
RES – physical memory (RAM) used
%MEM – portion of total system RAM used (RES)

top - RAM



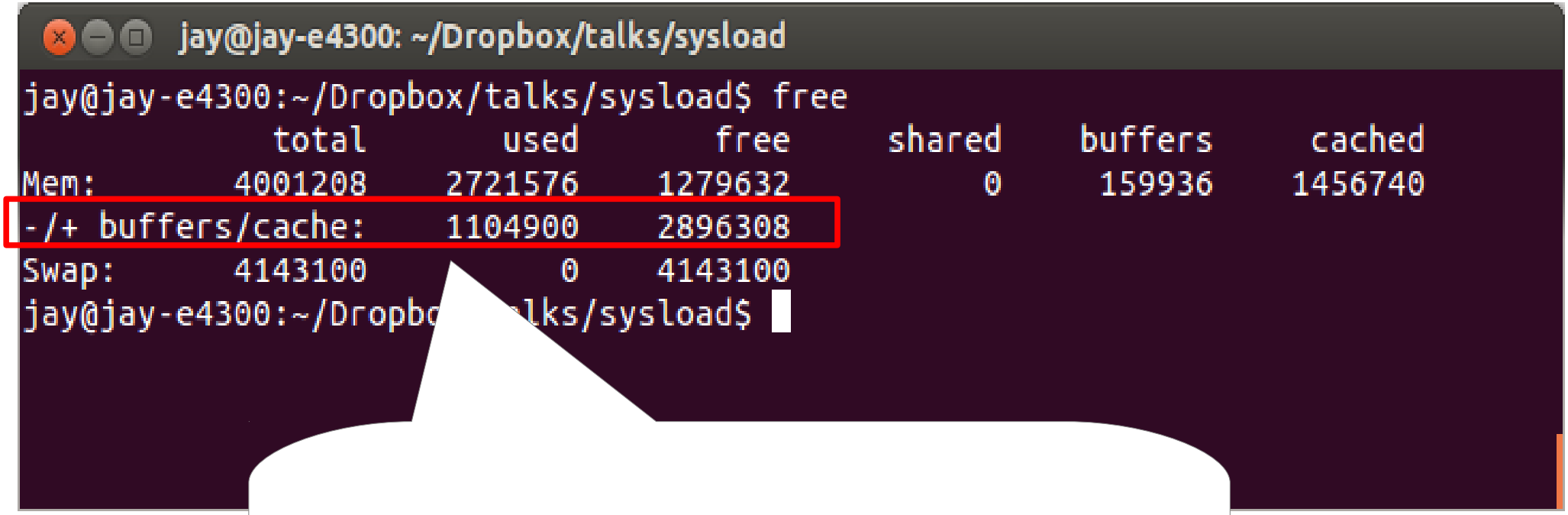
Total memory use:

Mem - total physical RAM installed
Used - physical RAM used by tasks (RES) + buffers + cache
Free - physical RAM free
Buffers – physical RAM used for OS buffers
Cached – physical RAM used for disk cache

Thus: physical RAM used by tasks = used – buffers - cached

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

free - RAM



free calculates RAM use ignoring buffers & cache.

The easiest way to see how much RAM your processes are really using.

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

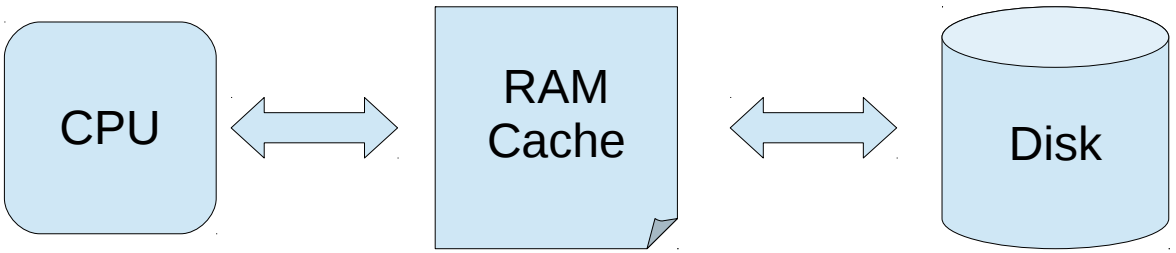
Disk - basics

- Persistant storage (HDD, SDD, RAID etc) is very slow compared to RAM.
- Sequential access is faster than random access.
- Throughput is typically limited by the physical device (eg magnetic HDD), not the interface (eg SATA).
- Exception to above is USB, which is very slow and limited by the USB interface.
- Read/write requests are serialised in one queue for each physical device.

Credit: slashgear.com

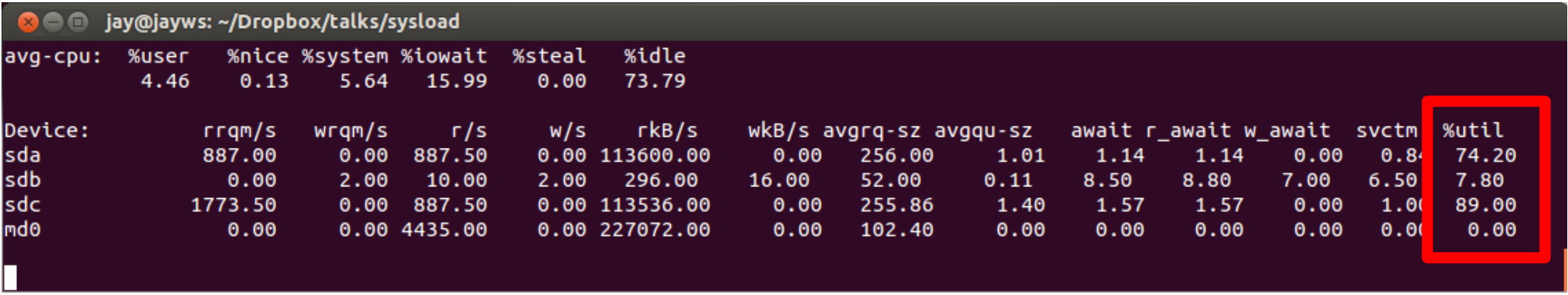
Disk - cache

- Linux has an effective disk cache.
- Both read and write is cached in RAM.
- The first read is slow, subsequent reads **much** faster if data in cache.
- Eg my laptop is >10x faster reading a big file from RAM cache than SSD.
- Writes are cached in RAM and written to disk later.
- If you read or write data that is much larger than the cache you get little benefit.
- Linux will use all the free RAM for cache as files are accessed.



Disk - iostat

- Linux: **iostat** (from sysstat package)
  - **iostat -x 2**

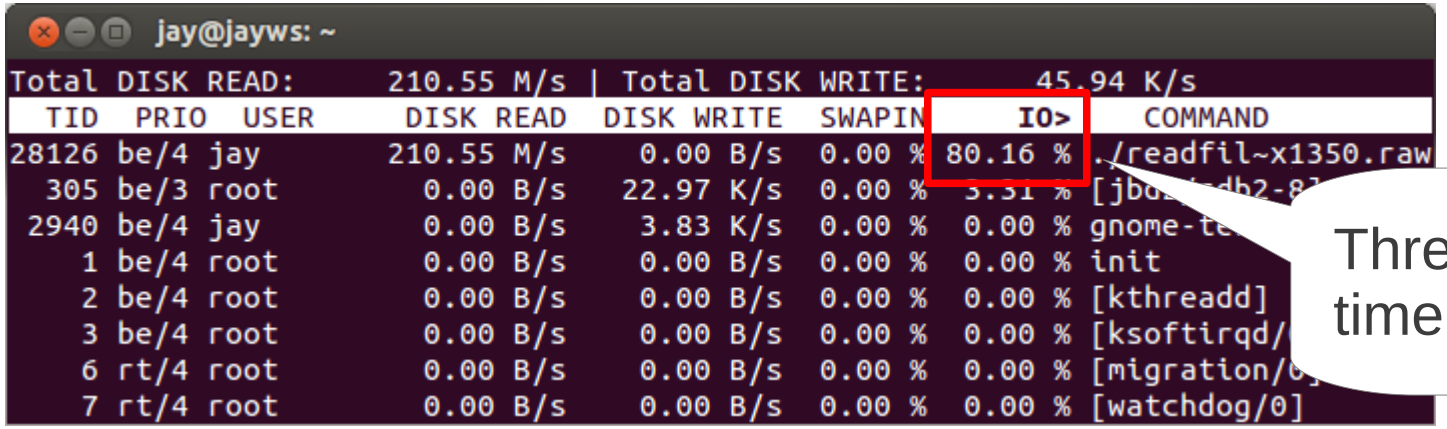


%util column:

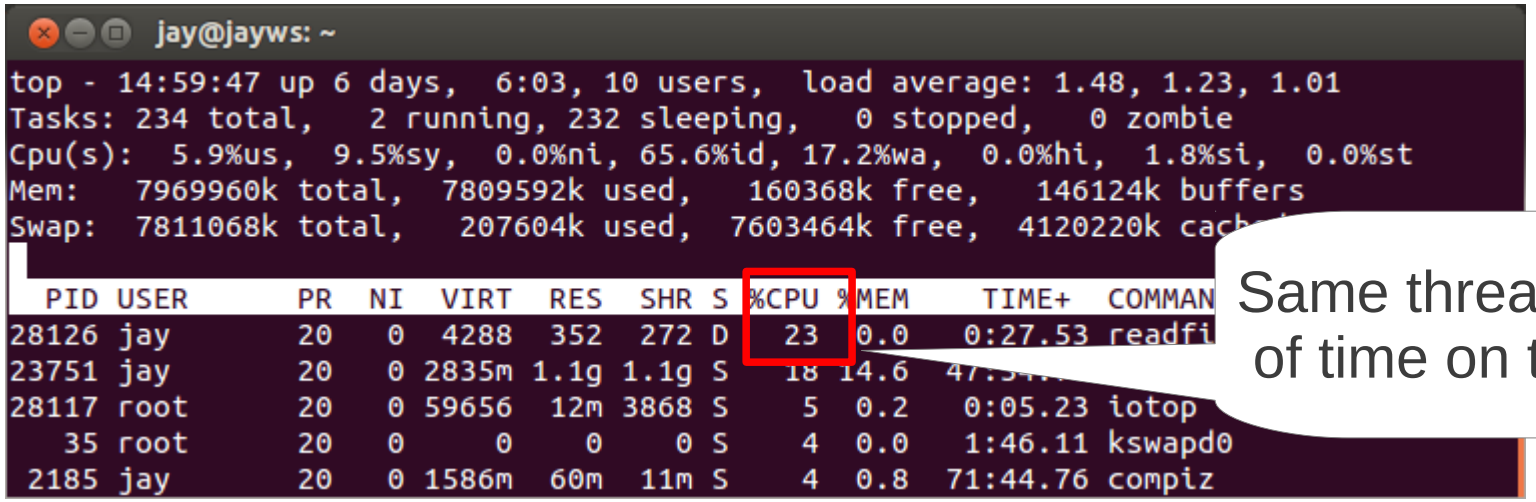
- Shows utilisation of disk (physical device).
- 100% means disk is fully utlised, ie bottleneck.
- I can't find equivalent on Mac :(
- Not available for network drives.
- Can't just use MB/s because:
  - Throughput changes depending on location of data on platter.
  - Throughput is much lower for random access patterns.

Disk - iotop

- Linux/Mac: **iotop** (requires root to run, use -P on Mac).
- Shows per-thread disk throughput and IO wait time.



Thread spending 80% of time waiting for IO (disk)



Same thread spending ~20% of time on the CPU

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

Network Drives

Credit: batterystop.com.au
Credit: fareastgizmos.com

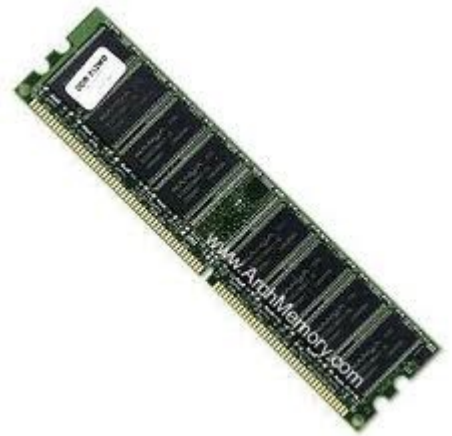
- Network drives are usually on RAID arrays.
- May have 5x more throughput than single drive, but 100x more users!?
- Random performance depends on network latency.
- Usually not a good choice for heavy IO. Prefer local disks.
- Tools:
  - **lsof <PID>** – list the files a process has open.
  - **mount** – check if a path is a network drive or not.

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

Overview

- Basic theory and monitoring tools for:
  - CPU
  - RAM
  - Disk
- Finding the bottleneck.
- How to complete your task faster.

Credit: sharkyextreme.com



Credit: clickbd.com


Credit: slashgear.com

Determining the bottleneck

- Is it CPU bound? Run **top**.
  - If a process is near 100% it is probably CPU bound, but you need to show threads (Shift-H) to be sure.
  - If a process is above 100% it must be multi-threaded. View threads with Shift-H. If a thread is near 100% it is CPU bound.
  - Is the CPU overloaded, ie more busy threads than cores? Check total CPU use (and idle) in **top**.
  - Watch out for Hyper Threading: 50% total load means the CPUs are almost fully utilised. You can (and usually should) run as many busy threads as HT cores but don't expect double the throughput: up to 30% extra in best case.

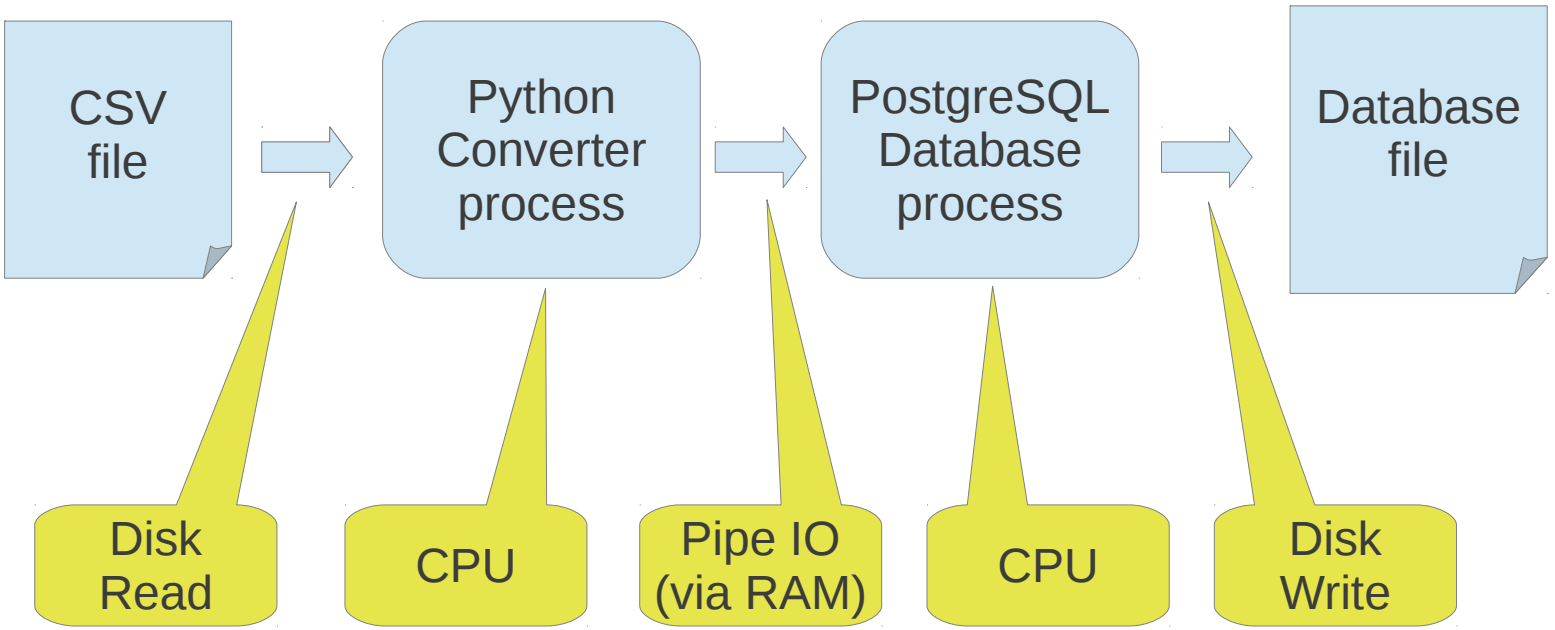
Determining the bottleneck

- Is it RAM limited? Run **free**.
  - If free RAM (not including buffers/cache) is low then your machine is out of RAM. Check RES column in **top** to find the culprit**.**
- Is it disk bound? Run **iostat -x 2** and **iotop (-P).**
  - **iostat**: if any device is near 100% util your workload is disk bound (assuming no unrelated task is generating significant disk load).
  - **iotop**: if any task is near 100% IO Wait it is disk-bound.
- A task may be bound by a combination of CPU and disk! Typically:
  - CPU + IO Wait ~= 100% (check **top** and **iotop**)

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

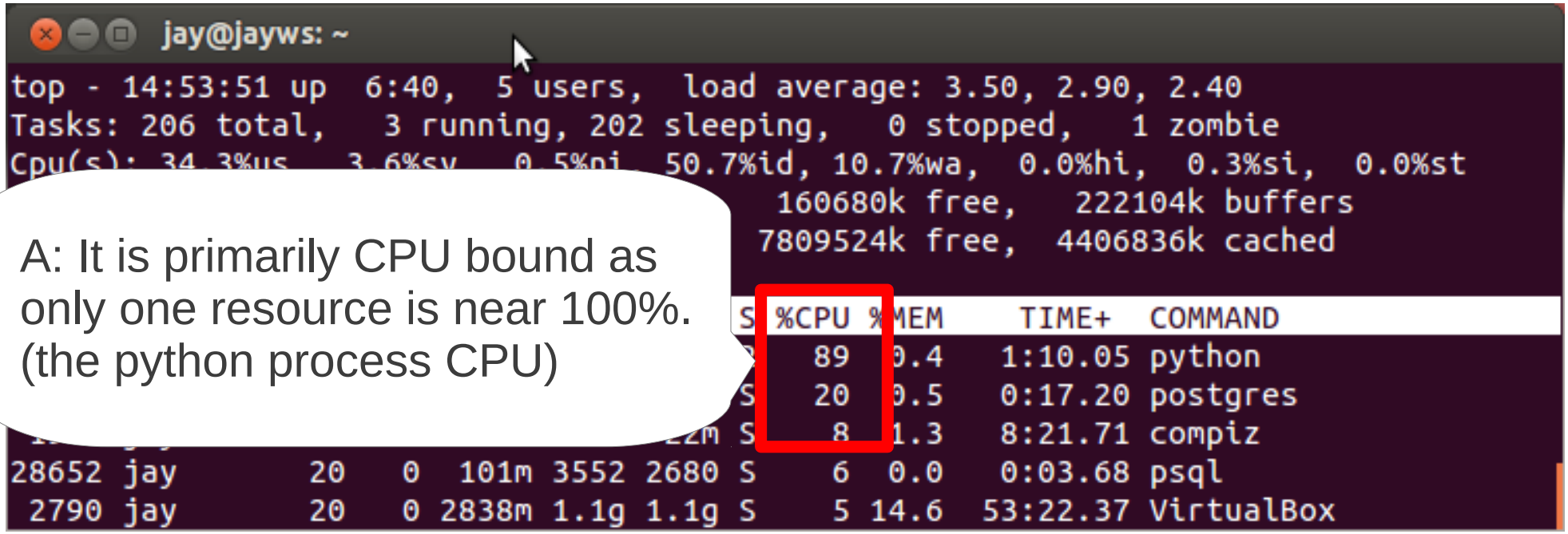
Example

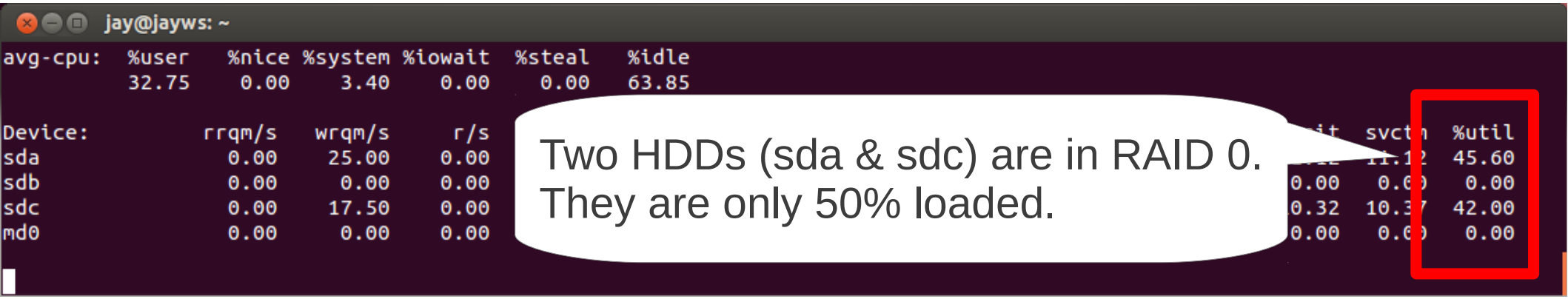
- A Python task to import a large CSV file into a SQL database.



Example

Q: Is this task CPU, RAM or Disk bound?



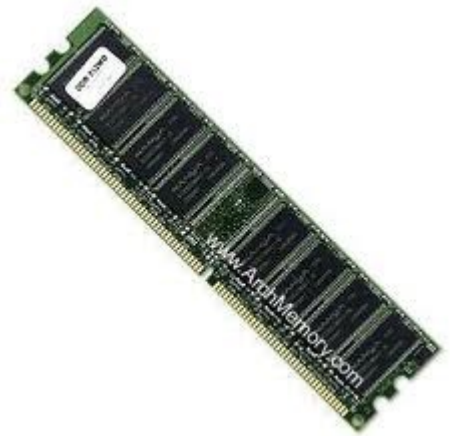


CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

Overview

- Basic theory and monitoring tools for:
  - CPU
  - RAM
  - Disk
- Finding the bottleneck.
- How to complete your task faster.

Credit: sharkyextreme.com



Credit: clickbd.com


Credit: slashgear.com

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

How to run faster – CPU bound

- Check for options to make it multi-threaded (although rare).
- If it's your code:
  - Do any easy optimisations.
  - Consider making it multi-threaded (another topic....).
- For single-threaded tasks:
  - Get a faster CPU. A modern CPU may be >2 times faster than one from 5 years ago. Clock rate != performance. See http://www.cpubenchmark.net/
  - Run multiple processes at once if possible (“poor man's multi-threading”).
- For multi-threaded tasks:
  - Get more cores if the software will scale.
  - Get a faster CPU.
- Making software scale across multiple cores is one of the biggest challenges in computing. Lots of software is still single-threaded.

How to run faster – RAM bound

- If it's your code optimise the memory allocation to minimise peak use.
- Get more RAM!

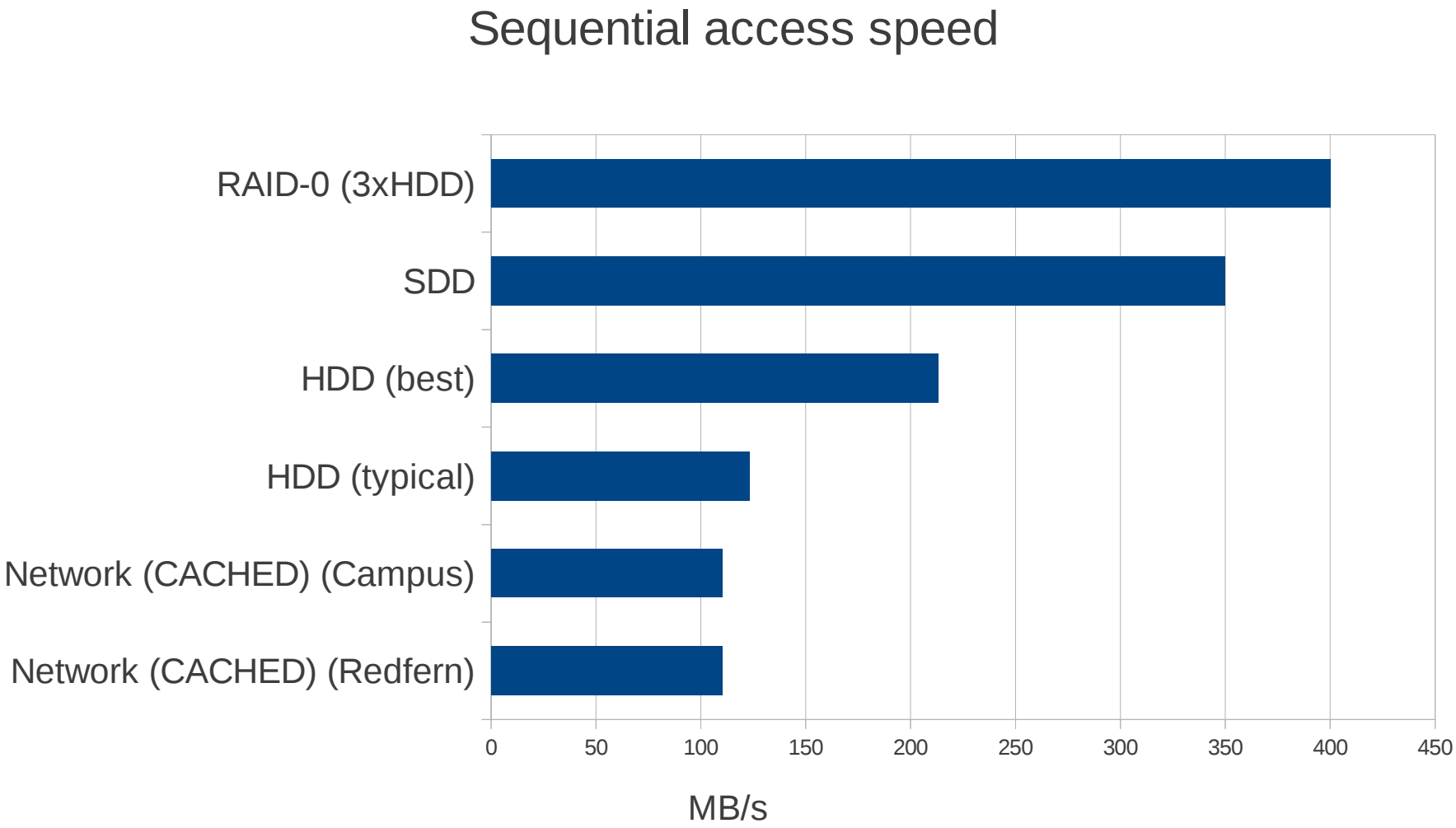
- An aside:
  - Most CPU-bound software is actually limited by memory access speed not CPU speed. A system with better memory performance may help.

How to run faster - disk bound

- If the task reads data multiple-times, or is blocked writing, consider if more RAM would improve caching.
- Consider changing data access pattern to make better use of cache if possible.
- Get a faster disk:
  - RAID arrays provide better sequential and random performance than a single drive.
  - Solid State Drives (SSDs) provide better sequential performance than a single HDD but not a big RAID.
  - SSDs provide vastly better random performance than HDD and RAID.
  - If your dataset is very large and mostly involves sequential access (typical for science?) then a RAID will deliver the best performance and value.

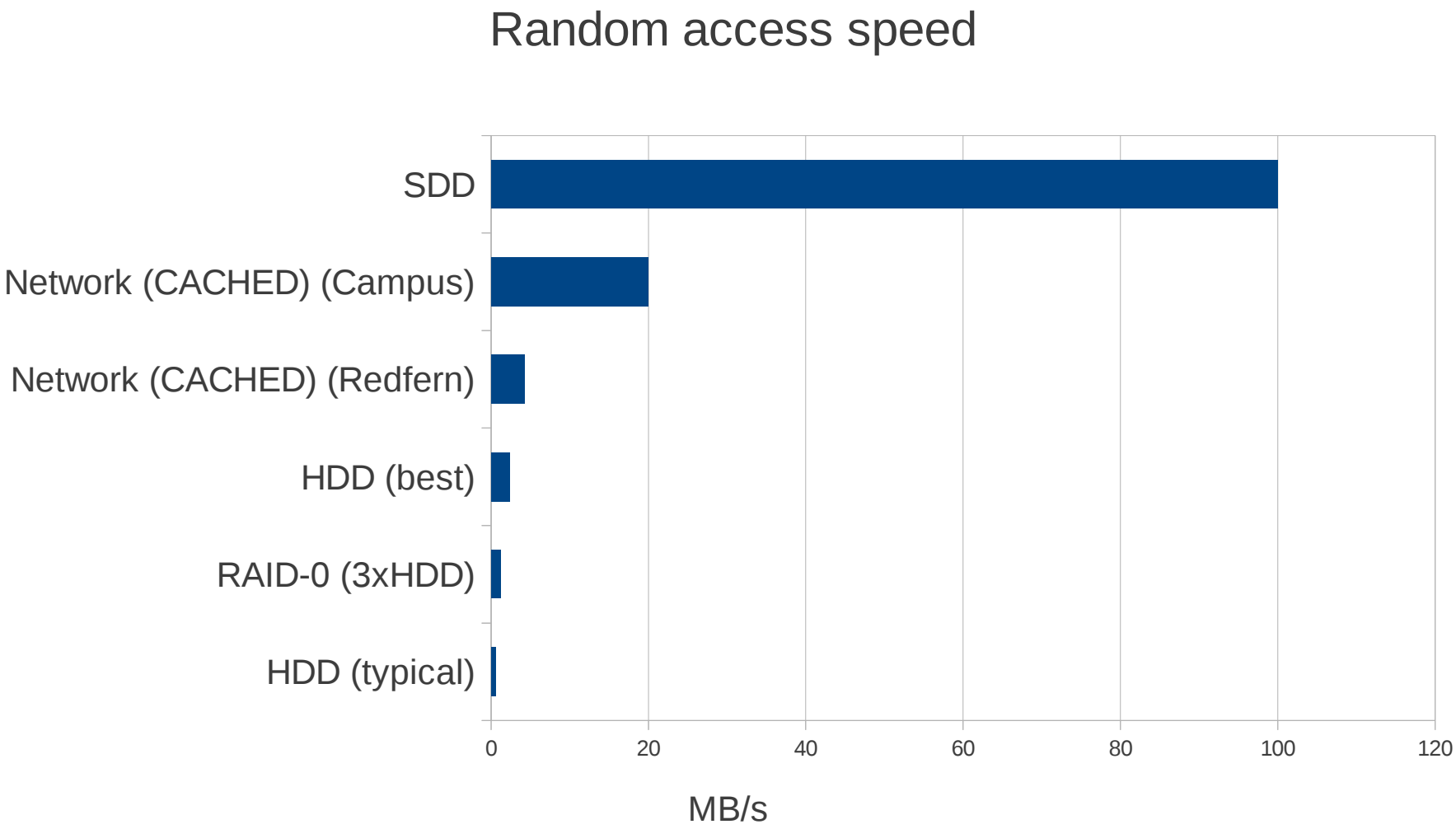
CAASTRO
ARC CENTRE OF EXCELLENCE
FOR ALL-SKY ASTROPHYSICS

Disk comparison



CAASTRO
ARC CENTRE OF EXCELLENCE
FOR ALL-SKY ASTROPHYSICS

Disk comparison



CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

The End

- Summary of tools:
  - CPU: **top, lscpu, sysctl, cat /proc/cpuinfo**
  - RAM: **top, free**
  - Disk IO: **top, iostat, iotop, lsof**

- If you understand how processes use computing resources and how to monitor them, you can:
  - Determine how to make your tasks complete faster
  - Not waste (your supervisor's) money on ineffective hardware upgrades
  - Be polite on a shared computer :)

Hard Disk Drives (HDD)

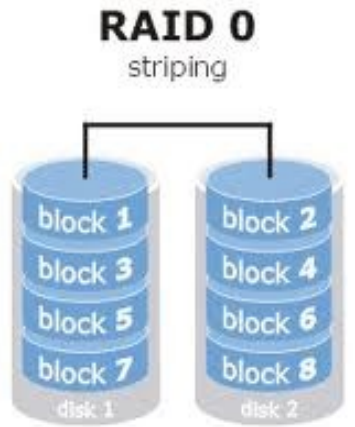
- Hard Disk Drives (HDDs) store data on rotating platters.
- They are relatively slow:
  - Sequential read/write: ~100MB/s. Eg 1GB file takes 10s to read.
  - Random read/write: ~1MB/s. Caused by time to move heads (seek) and platter to rotate.
- RAM is **much** faster: ~10,000MB/s sequential.

Credit: slashgear.com

Disk – RAID arrays

- Redundant Array of Independent Disks (RAID).
- Multiple disks combined to appear as one disk.
- Different ways to configure, called “levels”.
- All levels provide failure tolerence except RAID 0.
- Most levels also provide additional performance by spreading the load across multiple drives.

Credit: fareastgizmos.com



Credit: ajmatson.com

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

Solid State Drives (SSD)

- Use FLASH memory instead of magnetic storage. No moving parts.
- Up to 5x faster sequential performance than single HDD, eg ~500MB/s. Usually limited by drive interface eg SATA.
- >20x faster random performance than HDD.
- Much more expensive per GB than HDD (but getting cheaper):
  - SDD: ~$1.50 / GB
  - HDD: ~$0.10 / GB
- RAM still much faster: ~10,000MB/s. Cache still better!

CAASTRO
ARC CENTRE OF EXCELLENCE
FOR ALL-SKY ASTROPHYSICS

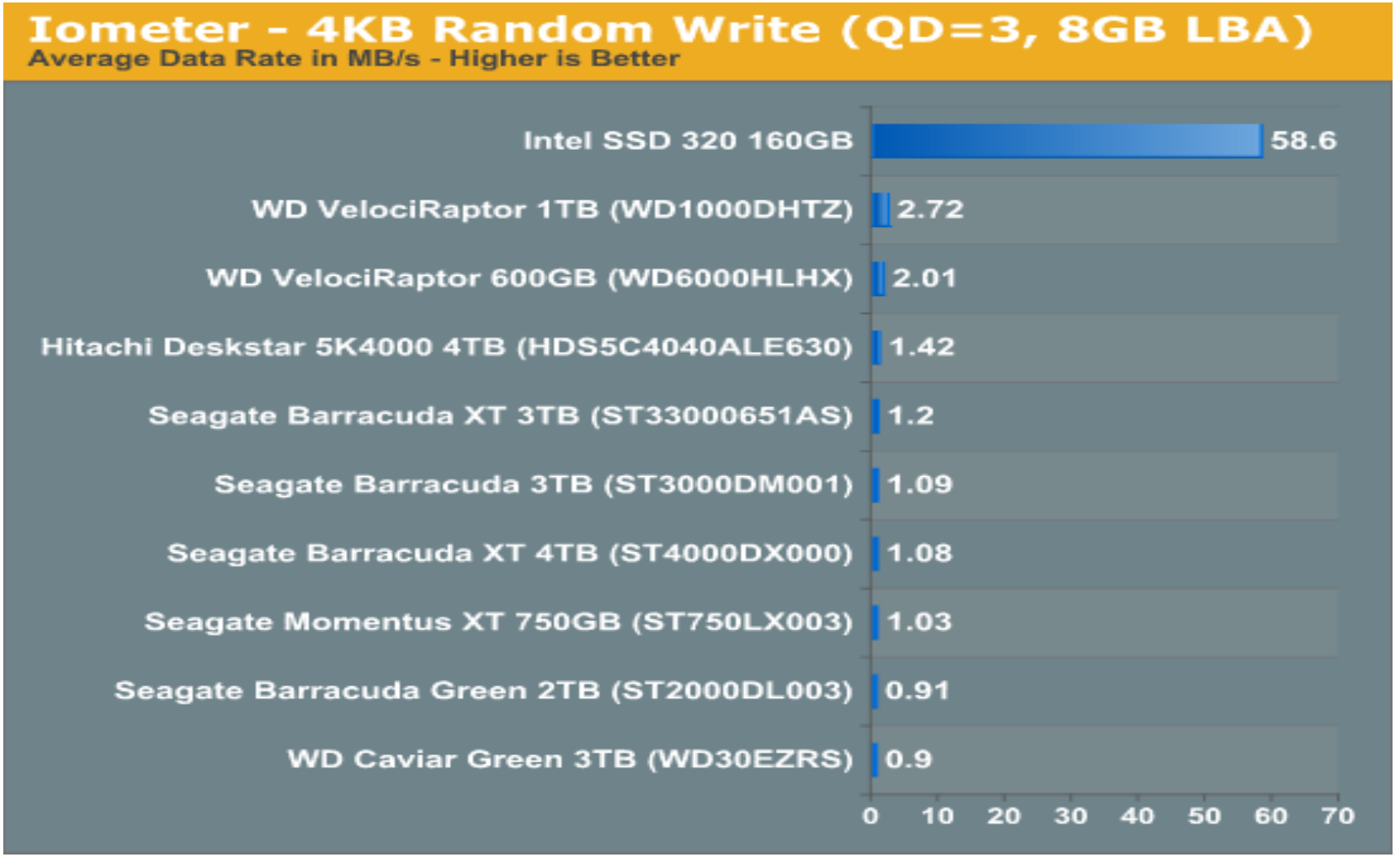
Solid State Drives (SSD)

Credit: anandtech.com

SSDs provide >2x sequential read/write speed of average HDD

CAASTRO
ARC CENTRE OF EXCELLENCE FOR ALL-SKY ASTROPHYSICS

Solid State Drives (SSD)



Credit: anandtech.com

SSDs provide >20x random read/write speed of average HDD.

This is their main advantage.