

# Large system usage HOW TO



George Magklaras PhD  
Biotek/NCMM IT  
USIT Research Computing Services

# Agenda

- Introduction: A Linux server as a collection of memory/disk/CPU
- What is the problem?
- memory and SWAP space
- What are the limits on biotin?
- The 'htop' application
- An algorithm/procedure to help you not overload the system and get your work done
- Examples

# Introduction

- Each server has a certain amount of CPU cores
  - `cat /proc/cpuinfo | grep processor | wc -l`
- A certain amount of RAM memory
  - `free`
- And a certain amount of disk space
  - `df -h`

# Introduction (2)

```
[georgios@biotin ~]$ cat /proc/cpuinfo | grep processor | wc -l
32
[georgios@biotin ~]$ free
              total        used         free       shared    buffers     cached
Mem:          264514940    202499088    62015852           0     5482912    189410216
-/+ buffers/cache:    7605960    256908980
Swap:         10485752           0     10485752
[georgios@biotin ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/internvg-root
                2.0G  727M  1.2G  38% /
tmpfs           127G   0  127G   0% /dev/shm
/dev/sda1       504M   99M  380M  21% /boot
/dev/mapper/internvg-opt
                13G   9.7G  1.8G  85% /opt
/dev/mapper/internvg-tmp
                63G  180M   60G   1% /tmp
/dev/mapper/internvg-usr
                39G   23G   14G  63% /usr
/dev/mapper/internvg-var
                7.9G  1.7G  5.9G  22% /var
/dev/mapper/VGpreben-LVpreben
                5.4T  3.7T  1.5T  72% /storage/mortharea
/dev/mapper/VGmills-LVmills
                13T  9.5T  2.5T  80% /storage/millsarea
/dev/mapper/VGEMBGalaxy-LVembgalaxy
                7.0T  4.7T  2.4T  67% /storage/tools
/dev/mapper/VGscratch-LVhurtado
                6.0T  4.1T  2.0T  68% /storage/hurtadoarea
/dev/mapper/VGscratch-LVscratch
                8.6T  1.5T  7.2T  17% /storage/scratch
192.168.8.126:/storage/staerkarea/
                3.0T  2.2T  884G  72% /storage/staerkarea
frigg.uio.no:/div/frigg/u6
                500G  151G  350G  31% /div/frigg/u6
frigg.uio.no:/div/frigg/u7
                500G  173G  327G  35% /div/frigg/u7
```

# What is the problem?

- The most important resource in a shared server is memory.
- We cannot overuse memory, because things will start crashing, affecting people's work.
- If you overuse CPU cores or disk space, this can be to a certain extent tolerable. Not with memory.
- Memory exhaustion crashes have been caused twice in the past. We need to learn how to monitor for memory usage and prevent this from occurring.

# Memory and SWAP space

- SWAP is hard disk space that is used to replace RAM.
- When the operating system kernel is running out of RAM (or it considers that certain programs are inactive), it transfers them to swap, until memory becomes available again.
- 'swap' kills performance: hard disk is much slower than RAM.
- When a system uses the SWAP space heavily, things are not optimal and it shows that memory is overused.

# Example of heavy SWAP usage

```
top - 13:41:43 up 46 days, 1:32, 3 users, load average: 1.27, 1.16, 1.17
Tasks: 1772 total, 2 running, 1759 sleeping, 11 stopped, 0 zombie
Cpu(s): 10.4%us, 0.4%sy, 0.0%ni, 89.1%id, 0.1%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 529247244k total, 497745684k used, 31501560k free, 12780k buffers
Swap: 10485752k total, 5242608k used, 5243144k free, 187682484k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
50058		20	0	293g	288g	1724	R	96.2	57.2	3799:54	bogart
49642	root	20	0	20464	2496	884	R	10.5	0.0	0:00.10	top
1	root	20	0	25684	792	532	S	0.0	0.0	0:53.80	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.60	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:39.75	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	1374:58	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:06.82	watchdog/0
7	root	RT	0	0	0	0	S	0.0	0.0	0:08.32	migration/1
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
9	root	20	0	0	0	0	S	0.0	0.0	286:54.16	ksoftirqd/1
10	root	RT	0	0	0	0	S	0.0	0.0	0:03.70	watchdog/1
11	root	RT	0	0	0	0	S	0.0	0.0	0:04.69	migration/2
12	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/2
13	root	20	0	0	0	0	S	0.0	0.0	287:27.51	ksoftirqd/2

# The limits on the biotin server

- Biotin has 256 Gigs of RAM memory.
- We can allocate 150 Gig to run for life science computing.
- The other 106 Gigs we need to run other critical services (VM cluster, file cache, etc)
- This means that amongst ALL of you, you should not exceed 150 Gigs of RAM.
- Cores and dedicated disk space we do not care. Memory we do care about. Limits have been enforced to ensure that failsafes function.



# The 'htop' application

```
1 [||||| 14.6%] 9 [ 0.0%] 17 [||||| 100.0%] 25 [||||| 100.0%]
2 [||||| 100.0%] 10 [||||| 100.0%] 18 [ 0.0%] 26 [ 1.9%]
3 [||||| 100.0%] 11 [||||| 100.0%] 19 [ 0.0%] 27 [||||| 100.0%]
4 [||||| 100.0%] 12 [ 0.0%] 20 [||||| 100.0%] 28 [||||| 100.0%]
5 [||||| 29.2%] 13 [ 0.6%] 21 [||||| 100.0%] 29 [||||| 100.0%]
6 [||||| 100.0%] 14 [||||| 100.0%] 22 [||||| 100.0%] 30 [ 0.0%]
7 [||||| 100.0%] 15 [ 1.9%] 23 [ 0.0%] 31 [||||| 100.0%]
8 [||||| 100.0%] 16 [||||| 100.0%] 24 [||||| 100.0%] 32 [ 0.0%]
Mem[||||| 181163/258315MB] Tasks: 227, 246 thr; 23 running
Swp[ 0/10239MB] Load average: 20.94 21.69 21.04
Uptime: 23 days, 03:25:25

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
24482 tonih 20 0 11.9G 11.7G 1744 R 100. 4.6 3h02:38 python makeCDT_V5_window.py TH74.x.f.bowtie2 sorted filtered.bed FAIRE_regions_II.cff FAIRE_regionsII FAIRE_siNT_CTR.cdt MCF7_input_Oslo_sorted.bed
53420 tonih 20 0 9110M 8953M 1744 R 100. 3.5 1h00:47 python makeCDT_V5_window.py TH72.f.Bowtie2_sorted_filtered_ST.bed FAIRE_regions_III.cff FAIRE_regionsIII FAIRE_siNT_EGF.cdt MCF7_input_Oslo_sorted.b
58727 tonih 20 0 8117M 7987M 1744 R 100. 3.1 33:09.54 python makeCDT_V5_window.py MCF7PI3K.f_sort_filtered_ST.bed FAIRE_regions_I.cff FAIRE_regionsI CHIP_FOXA1_PI3K.cdt MCF7_input_Oslo_sorted.bed hgl9_s
58634 tonih 20 0 8776M 8641M 1744 R 100. 3.3 34:02.60 python makeCDT_V5_window.py MCF7LAP.f_sort_filtered_ST.bed FAIRE_regions_I.cff FAIRE_regionsI CHIP_FOXA1_LAP.cdt MCF7_input_Oslo_sorted.bed hgl9_siz
51294 tonih 20 0 9101M 8966M 1744 R 100. 3.5 1h11:13 python makeCDT_V5_window.py MCF7PI3K.f_sort_filtered_ST.bed FAIRE_regions_II.cff FAIRE_regionsII CHIP_FOXA1_PI3K.cdt MCF7_input_Oslo_sorted.bed hgl9
57906 tonih 20 0 8559M 8389M 1744 R 100. 3.2 35:44.38 python makeCDT_V5_window.py TH70.f.Bowtie2_sorted_filterd_ST.bed FAIRE_regions_I.cff FAIRE_regionsI FAIRE_siNT_Heregulin.cdt MCF7_input_Oslo_sorted.
53154 tonih 20 0 10.2G 10.1G 1744 R 100. 4.0 1h03:14 python makeCDT_V5_window.py MCF7LAP.f_sort_filtered_ST.bed FAIRE_regions_III.cff FAIRE_regionsIII CHIP_FOXA1_LAP.cdt MCF7_input_Oslo_sorted.bed hgl9
24550 tonih 20 0 11.9G 11.8G 1744 R 100. 4.7 3h02:17 python makeCDT_V5_window.py TH70.f.Bowtie2_sorted_filterd_ST.bed FAIRE_regions_II.cff FAIRE_regionsII FAIRE_siNT_Heregulin.cdt MCF7_input_Oslo_sorte
52221 tonih 20 0 9413M 9252M 1744 R 100. 3.6 1h06:44 python makeCDT_V5_window.py TH74.x.f.bowtie2_sorted_filtered.bed FAIRE_regions_III.cff FAIRE_regionsIII FAIRE_siNT_CTR.cdt MCF7_input_Oslo_sorted.be
53311 tonih 20 0 10200M 10058M 1744 R 100. 3.9 1h01:47 python makeCDT_V5_window.py MCF7PI3K.f_sort_filtered_ST.bed FAIRE_regions_III.cff FAIRE_regionsIII CHIP_FOXA1_PI3K.cdt MCF7_input_Oslo_sorted.bed
58839 tonih 20 0 7346M 7197M 1744 R 100. 2.8 31:58.50 python makeCDT_V5_window.py TH72.f.Bowtie2_sorted_filtered_ST.bed FAIRE_regions_I.cff FAIRE_regionsI FAIRE_siNT_EGF.cdt MCF7_input_Oslo_sorted.bed h
58001 tonih 20 0 8920M 8784M 1744 R 100. 3.4 34:45.32 python makeCDT_V5_window.py MCF7CTR.f_sort_filtered_ST.bed FAIRE_regions_I.cff FAIRE_regionsI CHIP_FOXA1_CTR.cdt MCF7_input_Oslo_sorted.bed hgl9_siz
52376 tonih 20 0 10.0G 10108M 1744 R 100. 3.9 1h05:30 python makeCDT_V5_window.py TH70.f.Bowtie2_sorted_filterd_ST.bed FAIRE_regions_III.cff FAIRE_regionsIII FAIRE_siNT_Heregulin.cdt MCF7_input_Oslo_so
53061 tonih 20 0 10.4G 10.3G 1744 R 100. 4.1 1h03:58 python makeCDT_V5_window.py MCF7CTR.f_sort_filtered_ST.bed FAIRE_regions_III.cff FAIRE_regionsIII CHIP_FOXA1_CTR.cdt MCF7_input_Oslo_sorted.bed hgl9
51423 tonih 20 0 8403M 8249M 1744 R 100. 3.2 1h09:13 python makeCDT_V5_window.py TH72.f.Bowtie2_sorted_filtered_ST.bed FAIRE_regions_II.cff FAIRE_regionsII FAIRE_siNT_EGF.cdt MCF7_input_Oslo_sorted.bed
24789 tonih 20 0 12.0G 11.9G 1744 R 99.0 4.7 2h59:38 python makeCDT_V5_window.py MCF7LAP.f_sort_filtered_ST.bed FAIRE_regions_II.cff FAIRE_regionsII CHIP_FOXA1_LAP.cdt MCF7_input_Oslo_sorted.bed hgl9_s
24552 tonih 20 0 15.9G 14.8G 1748 R 99.0 5.9 3h02:10 python makeCDT_V5_window.py MCF7CTR.f_sort_filtered_ST.bed FAIRE_regions_II.cff FAIRE_regionsII CHIP_FOXA1_CTR.cdt MCF7_input_Oslo_sorted.bed hgl9_s
57786 tonih 20 0 7492M 7339M 1744 R 98.0 2.8 36:36.78 python makeCDT_V5_window.py TH74.x.f.bowtie2_sorted_filtered.bed FAIRE_regions_I.cff FAIRE_regionsI FAIRE_siNT_CTR.cdt MCF7_input_Oslo_sorted.bed hg
13500 tonih 20 0 110M 2156 1164 S 0.0 0.0 0:00.04 sshd: tonih@pts/1
13501 tonih 20 0 120M 2212 1624 S 0.0 0.0 0:00.09 -bash
13753 tonih 20 0 110M 2160 1164 S 0.0 0.0 0:00.01 sshd: tonih@pts/17
13754 tonih 20 0 120M 2172 1596 S 0.0 0.0 0:00.05 -bash
14082 tonih 20 0 110M 2156 1160 S 0.0 0.0 0:00.01 sshd: tonih@pts/19
14083 tonih 20 0 120M 2180 1600 S 0.0 0.0 0:00.06 -bash
14398 tonih 20 0 110M 2152 1160 S 0.0 0.0 0:00.06 sshd: tonih@pts/20
14399 tonih 20 0 120M 2176 1600 S 0.0 0.0 0:00.08 -bash
50969 tonih 20 0 110M 2160 1164 S 0.0 0.0 0:00.07 sshd: tonih@pts/21
50973 tonih 20 0 120M 2212 1620 S 0.0 0.0 0:00.11 -bash
51370 tonih 20 0 110M 2156 1164 S 0.0 0.0 0:00.01 sshd: tonih@pts/16
51371 tonih 20 0 120M 2196 1616 S 0.0 0.0 0:00.04 -bash
52115 tonih 20 0 110M 2160 1164 S 0.0 0.0 0:00.04 sshd: tonih@pts/23
52116 tonih 20 0 120M 2164 1592 S 0.0 0.0 0:00.07 -bash
52257 tonih 20 0 110M 2156 1164 S 0.0 0.0 0:00.03 sshd: tonih@pts/25
52258 tonih 20 0 120M 2160 1592 S 0.0 0.0 0:00.11 -bash
52393 tonih 20 0 110M 2160 1164 S 0.0 0.0 0:00.04 sshd: tonih@pts/26
52394 tonih 20 0 120M 2160 1592 S 0.0 0.0 0:00.07 -bash
53072 tonih 20 0 110M 2156 1164 S 0.0 0.0 0:00.03 sshd: tonih@pts/27
53073 tonih 20 0 120M 2164 1592 S 0.0 0.0 0:00.06 -bash
F1Help F2Setup F3Search F4Filter F5Tree F6SortB F7Nice F8Nice F9Kill F10Quit
```

# The 'htop' application (2)

- In the previous slide, if we add all the numbers under the 'RES' column for user 'tonih', we get approx 150 Gigs of RAM in total.
- And that was for a single user, which would be OK if nobody else was running on the system.
- Difference: `htop` versus `htop -u [username]`
- You have to account for others that might be executing programs at the same time as you.

# An algorithm/procedure to regulate memory consumption

- **Before** you execute your heavy programs on biotin:
  - 1)Run `htop` and locate the Mem horizontal line which measures actual RAM consumption.
  - 2)If the amount of RAM  $\geq$  **150 Gigs (153600 MB)**, **please wait** until it comes well under 100 Gigs.
  - 3)If the amount of RAM  $<$  **150 Gigs( 153600 MB)**, you can consume  $150 \text{ Gigs} - [\text{current\_consumption\_by others}]$   
**for all your processes to share.**
- How to gauge your current memory consumption:
  - Run `'htop -u [your_userid]'` and ADD UP all the figures under the RES column. This is how much you are currently using on the system.

# Example 1:

```
1  [||||| 4.5%] 9 [ 0.0%]
2  [ 0.0%] 10 [ 0.0%]
3  [ 0.0%] 11 [ 0.0%]
4  [ 0.0%] 12 [ 0.0%]
5  [ 0.0%] 13 [ 0.0%]
6  [ 0.0%] 14 [ 0.0%]
7  [ 0.0%] 15 [ 0.0%]
8  [ 0.0%] 16 [ 0.0%]
Mem [||||| 44275/258315MB]
Swp [ 0/10239MB]
```

The current memory consumption on 'biotin' is approximately 44 Gigs (44275). How many 'novoalign' processes can I start at the same time, without overloading the system?

Solution: If I know that a single novoalign process can take up to 12 Gigs of RAM (depended on the parameters and my data set), I have  $150 - 44 = 106$  Gigabytes. Which means I can start a max of 8 of them ( $106/12 \rightarrow$  rounded to the lower integer).

# Example 2:

```
1  [|||] 4.5% 9 [ 0.0%]
2  [ 0.0%] 10 [ 0.0%]
3  [ 0.0%] 11 [ 0.0%]
4  [ 0.0%] 12 [ 0.0%]
5  [ 0.0%] 13 [ | 0.6%]
6  [ 0.0%] 14 [ 0.0%]
7  [ 0.0%] 15 [ 0.0%]
8  [ 0.0%] 16 [ 0.0%]
Mem [||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 71703/258315MB]
Swp [ 0/10239MB]

PID USER      PRI  NI  VIRT   RES    SHR S  CPU% MEM%   TIME+  Command
5816 georgios   20   0 40.0G 19.5G  484 R 100.  7.7  0:38.34 memhog 40G
5808 georgios   20   0 40.0G 22.8G  484 R 100.  9.1  0:41.78 memhog 40G
5810 georgios   20   0 40.0G 20.7G  484 R 100.  8.2  0:40.64 memhog 40G
```

The current memory consumption on 'biotin' is 70 Gigs (71703 MB). User georgios seems to be running these large processes that keep growing in size. How many Python `make_CDT_window_V5.py` scripts can I safely start without overloading the system.

Solution: If I know that a single script of that type can take up to 15 Gigs of RAM (depended on the parameters and my data set), I should have  $150 - 70 = 80$  Gigabytes. However, because I see user georgios acquiring RAM rapidly, I look at the VIRT column and I consider this as the maximum amount of RAM -> 120 Gigs (3 x 40 Gigs). Therefore, I have  $150 - 120 = 30$  Gigs, thus only 2 of these Python scripts could be started safely, given the current circumstances.

# Questions?

