

# Monitoramento no Linux

## Avaliação de Desempenho

Prof: Paulo Maciel <prmm@cin.ufpe.br

Instrutor: Jamilson Dantas <jrd@cin.ufpe.br>

# Monitoramento LINUX

- Ferramentas Essenciais

- /proc

- Top

- Uptime

- Vmstat

- Free

- Sysstat

- iostat

- mpstat

- pidstat

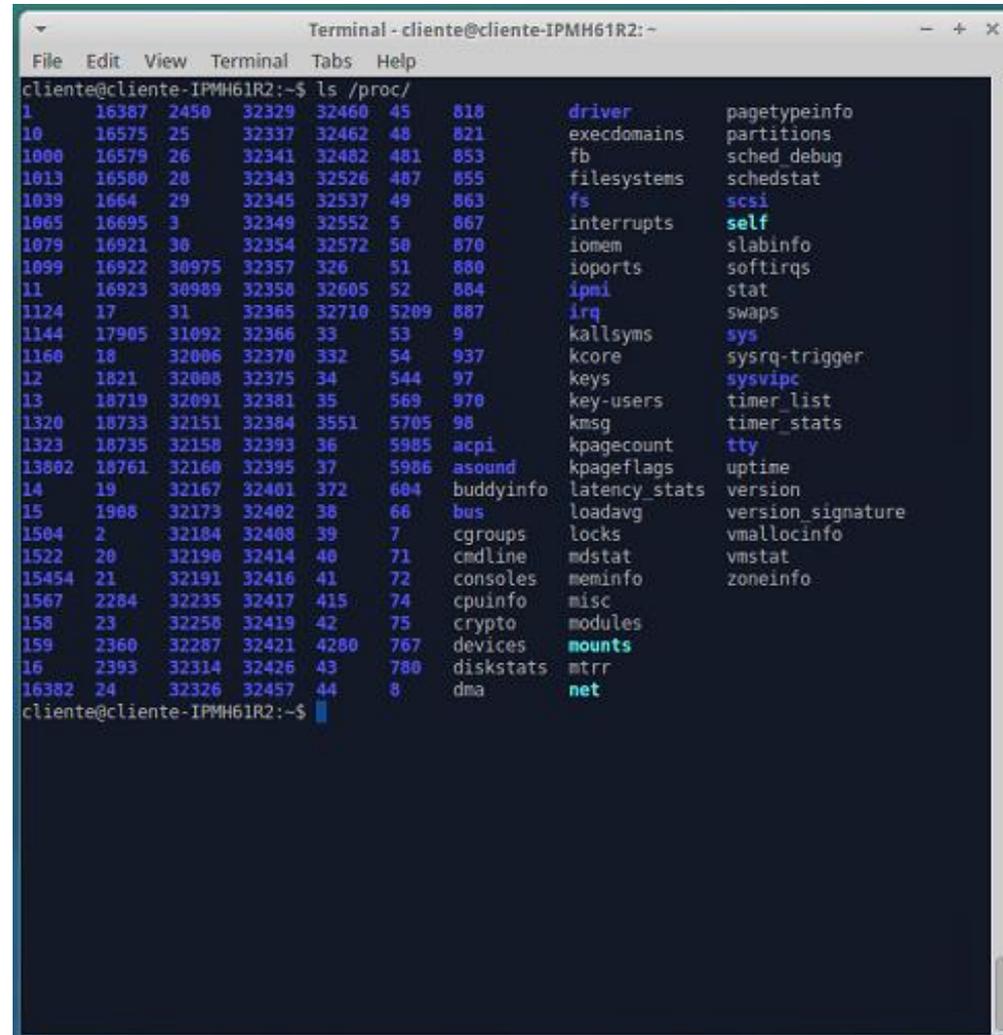
- Dstat

- tcpdump

# Utilizando o /proc

## ● /proc

- Pseudo-sistema de arquivos, existente no GNU/Linux e em vários outros SOs baseados no Unix
- Estruturado como uma hierarquia de diretórios e arquivos



```
Terminal - cliente@cliente-IPMH61R2:~  
File Edit View Terminal Tabs Help  
cliente@cliente-IPMH61R2:~$ ls /proc/  
1      16387 2450 32329 32460 45 818 driver pagetypeinfo  
10     16575 25 32337 32462 48 821 execdomains partitions  
1000   16579 26 32341 32482 481 853 fb sched debug  
1013   16580 28 32343 32526 487 855 filesystems schedstat  
1039   1664 29 32345 32537 49 863 fs scsi  
1065   16695 3 32349 32552 5 867 interrupts self  
1079   16921 30 32354 32572 50 870 iomem slabinfo  
1099   16922 30975 32357 326 51 880 ioports softirqs  
11     16923 30989 32358 32605 52 884 ipmi stat  
1124   17 31 32365 32710 5209 887 irq swaps  
1144   17905 31092 32366 33 53 9 kallsyms sys  
1160   18 32006 32370 332 54 937 kcore sysrq-trigger  
12     1821 32008 32375 34 544 97 keys sysvipc  
13     18719 32091 32381 35 569 970 key-users timer_list  
1320   18733 32151 32384 3551 5705 98 kmsg timer_stats  
1323   18735 32158 32393 36 5985 acpi kpagecount tty  
13802  18761 32160 32395 37 5986 asound kpageflags uptime  
14     19 32167 32401 372 604 buddyinfo latency_stats version  
15     1988 32173 32402 38 66 bus loadavg version_signature  
1504   2 32184 32408 39 7 cgroups locks vmallocinfo  
1522   20 32190 32414 40 71 cmdline mdstat vmstat  
15454  21 32191 32416 41 72 consoles meminfo zoneinfo  
1567   2284 32235 32417 415 74 cpuinfo misc  
158    23 32258 32419 42 75 crypto modules  
159    2360 32287 32421 4280 767 devices nmounts  
16     2393 32314 32426 43 780 diskstats mtrr  
16382  24 32326 32457 44 8 dma net  
cliente@cliente-IPMH61R2:~$
```

# Utilizando o /proc

## ● /proc

- Interface para estruturas de dados internas do kernel (núcleo do sistema)
  - Acessar dados sobre processos e outros recursos do SO
  - Alterar parâmetros do kernel em tempo de execução
- Vários contadores de desempenho disponíveis
  - **/proc/stat**
  - **/proc/meminfo**
  - **/proc/vmstat**
  - **/proc/diskstats**
  - **/proc/net/...**
  - **/proc/<pid>/...**

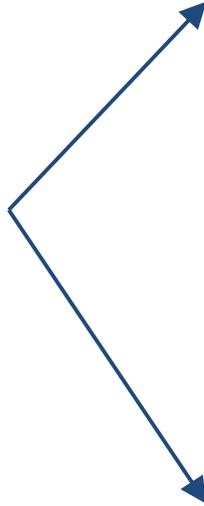
# /proc/meminfo

- /proc/meminfo

```
Terminal - cliente@cliente-IPMH61R2: ~
File Edit View Terminal Tabs Help

cliente@cliente-IPMH61R2:~$ cat /proc/meminfo
MemTotal:      3985152 kB
MemFree:       1442296 kB
Buffers:       232036 kB
Cached:        1666560 kB
SwapCached:    0 kB
Active:        1448692 kB
Inactive:      814416 kB
Active(anon):  365792 kB
Inactive(anon): 63312 kB
Active(file):  1082900 kB
Inactive(file): 751104 kB
Unevictable:   32 kB
Mlocked:       32 kB
SwapTotal:     4128764 kB
SwapFree:      4128764 kB
Dirty:         4 kB
Writeback:     0 kB
```

Informações bastante úteis para avaliar questões de desempenho



# /proc/meminfo

- Vamos monitorar a memória do sistema enquanto executamos um teste de stress.

```
stress --vm 1 --vm-bytes 10M -t 300s
```

Qtd de workers

Qtd de memória  
alocada/desalocada

Tempo total de  
execução

# /proc/meminfo

- Nós podemos monitorar em tempo real
- Ou salvamos em um arquivo (log) para visualizar depois

```
watch -n 5 cat /proc/meminfo
```

```
./monitora-memoria.sh
```

```
#!/bin/bash
tempo=0;
echo "MemFree Buffers Cached SwapFree">log-memoria.txt
while [ $tempo -lt 300 ]
do
    mfree=`cat /proc/meminfo | awk '/MemFree/{print $2}'`
    buff=`cat /proc/meminfo | awk '/Buffers/{print $2}'`
    cach=`cat /proc/meminfo | awk '/^Cached/{print $2}'`
    swapfree=`cat /proc/meminfo | awk '/SwapFree/{print $2}'`
    echo "$mfree $buff $cach $swapfree" >> log-memoria.txt
    sleep 5
    tempo=$((tempo + 5));
done
```

# /proc/<pid>/status

- Em muitas situações, é essencial medir o uso de recursos para um processo em específico

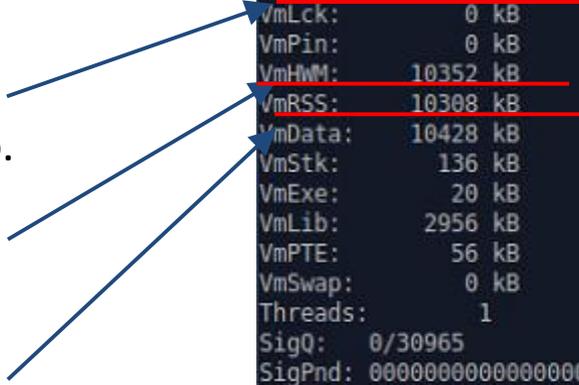
**VmSize**: toda a memória virtual usada pelo processo.

**VmHWM**: teto atingido pelo RSS

**Resident set size**: memória física (RAM) usada pelo processo.

```
root@cliente-IPMH61R2:/home/cliente# cat /proc/18953/status
Name:  stress
State:  R (running)
Tgid:  18953
Ngid:  0
Pid:   18953
PPid:  18952
TracerPid:  0
Uid:   1000 1000 1000 1000
Gid:   1000 1000 1000 1000
FDSize: 64
Groups: 4 24 27 30 46 108 124 125 1000
VmPeak: 17680 kB
VmSize: 17680 kB
VmLck:  0 kB
VmPin:  0 kB
VmHWM: 10352 kB
VmRSS: 10308 kB
VmData: 10428 kB
VmStk:  136 kB
VmExe:  20 kB
VmLib:  2956 kB
VmPTE:  56 kB
VmSwap:  0 kB
Threads: 1
SigQ:  0/30965
SigPnd: 0000000000000000

```



Como obter o PID:

```
root@cliente-IPMH61R2:/home/cliente# pgrep -n stress
18953
root@cliente-IPMH61R2:/home/cliente#
```

# Monitoramento LINUX

- Comando **TOP**

- Fornecer uma visão em tempo real do sistema em execução

- Sintaxe: top [opções]

- **-d** atraso Especifica o atraso em segundos entre as atualizações de tela. O padrão é 5 segundos.

- **-i** ignora processos ociosos.

- **-n** num Exibe num interações e depois termina.

- **-b** Roda em modo de batch. Útil para mandar a saída de top para outros programas ou um arquivo.

# Monitoramento LINUX

Comando **top** – opções interativas

**h** Gera um tela de ajuda

**k** Termina um processo (será pedido seu PID)

**q** Sai do programa

# Monitoramento LINUX

- **PID**: O identificador de cada processo
- **USER**: Usuário
- **PR**: Prioridade da Tarefa
- **NI**: Valor Nice da tarefa
- **VIRT**: Memória virtual usada
- **RES**: Memória física usada
- **SHR**: Memória compartilhada usada
- **S**: Estado da tarefa (**s** = sleeping, **R** = running, **T** = stopped, **Z** = zombie, etc.)
- **%CPU**: % de tempo de CPU
- **%MEM**: % de memória física
- **TIME+**: Tempo total de atividade da tarefa desde que ela foi iniciada
- **COMMAD**: Nome do processo

# Monitoramento LINUX

- Comando **uptime**

–Mostra o tempo atual, há quanto tempo o **sistema** está **rodando**, quantos **usuários** estão **logados** atualmente e as médias de carga do sistema nos últimos 1, 5 e 15 minutos.

```
cliente@cliente-IPMH61R2:~$ uptime
15:07:10 up 34 days, 10:53,  4 users,  load average: 0,20, 0,22, 0,23
cliente@cliente-IPMH61R2:~$
```

# Monitoramento LINUX

- Comando **vmstat**

- Este comando reporta informações sobre processos, memória, paginação, blocos de I/O, traps e atividades de CPU.

- Vmstat** [opções]

- S** M usa a unidade MB em vez do padrão KB

- a** Mostra memória ativa e inativa

- d** Mostra estatísticas de discos

- p** Partição Mostra informações de R/W na partição especificada

- s** Mostra estatísticas em formato de tabela

# Monitoramento LINUX

- **Vmstat** – campos

## 1.Procs

- r**: N<sup>o</sup> de processos esperando para rodar
- b**: N<sup>o</sup> de processos em dormência ininterrupta

## 2.Memory

- Swpd**: memória virtual usada
- Free**: memória livre
- Buff**: memória usada como buffer
- Cache**: memória usada como cache

## 3.Swap

- si**: memória trocada a partir do disco
- so**: memória trocada para o disco

# Monitoramento LINUX

- **Vmstat** – campos

## 1.io

- bi**: Blocos recebidos de um dispositivos de bloco (blocos/s)

- bo**: Blocos enviados a um dispositivo de bloco (blocos/s)

## 2.System

- in**: nº de interrupções por segundo, incluindo clock

- cs**: nº de mudanças de contexto por segundo

## 3.Cpu

- us**: Tempo gasto rodando código que não é kernel

- sy**: Tempo gasto rodando código do kernel

- id**: Tempo gasto em ociosidade

- wa**: Tempo gasto esperando por I/O

# Monitoramento LINUX

- Comando **free**

- Exibe a quantidade de memória livre e usada no sistema

Sintaxe: free [opções]

- b** Mostra o uso da memória em bytes

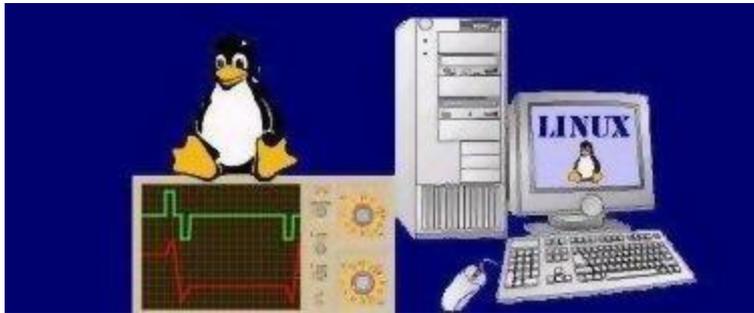
- k** uso da memória em KB

- m** em MB

- t** Exibe uma linha que mostra os totais

- s** n Operação contínua em intervalos de n segundos

# Utilizando o Sysstat



- O **sysstat** é um pacote de utilitários para coleta de dados de desempenho
  - **iotstat**: Disco e I/O em geral
  - **mpstat**: Processador e memória
  - **pidstat**: Monitoramento por processo

# Monitoramento LINUX

- Comando **iostat**

- Mostra informações sobre o uso da **CPU** e várias estatísticas sobre E/S do sistema.

Sintaxe: iostat [opções]

- c** Mostra apenas estatísticas da CPU

- d** Mostra apenas estatísticas de I/O de disco

- p sda** Mostra apenas estatística para sda

# Monitoramento LINUX

- Comando **mpstat**

- Exibe estatísticas sobre todos os processadores existentes na máquina

Sintaxe: mpstat [opções]

- P ALL – exibir estatísticas para todas as CPUs

- [Num] [num] – tempo de coleta dos dados e loop

# Monitoramento LINUX

- Comando **pidstat**

- Com o pidstat podemos monitorar as informações que encontram-se no /proc/<pid>/...,

Sintaxe: pidstat [opções]

- d** estatísticas de I/O

- u** Utilização de CPU

- p** <PID> número do processo

- r** page faults e utilização da memória

- [num] [num] intervalo em segundos e número de relatórios.

# Monitoramento LINUX

- Comando **pidstat**

```
stress --vm 1 --vm-bytes 10M -t 300s
```

```
root@cliente-IPMH61R2:/home/cliente# pidstat -p 21671 -r 2 3
Linux 3.13.0-37-generic (cliente-IPMH61R2)      21-11-2014      _x86_64_      (4 CPU)

12:09:12      UID      PID  minflt/s  majflt/s      VSZ      RSS      %MEM  Command
12:09:14      1000     21671     0,00     0,00     7308     432     0,01  stress
12:09:16      1000     21671     0,00     0,00     7308     432     0,01  stress
12:09:18      1000     21671     0,00     0,00     7308     432     0,01  stress
Average:      1000     21671     0,00     0,00     7308     432     0,01  stress
root@cliente-IPMH61R2:/home/cliente#
```

# Monitoramento LINUX

- Comando **dstat**

- Permite efetuar monitoramento e verificar performance do sistema Linux, possuindo características dos comandos **top**, **vmstat**, **free**, **iostat** combinadas.

Sintaxe: `dstat [opções]`

Dstat n permite ajustar o intervalo de atualização para n segundos

- m** uso de memória

- c** estatística de CPU

- d** Estatística de disco

- i** interrupções

- n** estatísticas de uso de rede

- fs** estatísticas do sistema de arquivos

- ntp** mostra a hora a partir de um servidor de NTP

# Monitoramento LINUX

- Monitorando Rede
  - Tcpcap

```
tcpdump -i eth0
```

```
tcpdump -w capture.cap
```

```
tcpdump -n dst host 192.168.1.1
```

```
tcpdump -n src host 192.168.1.1
```

```
tcpdump -nn -ni eth0 src host 192.168.10.254 -w  
/tmp/teste2.pcap
```

# Monitoramento LINUX (nmon)

- Nigel's performance **Monitor**;
- Por default o nmon inicia em modo live com monitoramento via terminal

```
nmon-16a-----[H for help]-----Hostname=vm26-----Refresh= 2secs -----08:53.14-----

-----
          For help type H or ...
          nmon -?  - hint
          nmon -h  - full details

          To stop nmon type q to Quit

-----

DISTRIB_DESCRIPTION="Ubuntu 15.04"
PowerVM POWER8 (architected), altivec supported CHRP IBM,8284-22A
PowerVM Entitlement=1.00  VirtualCPUs=2 LogicalCPUs=16
PowerVM SMT=8 Capped=0
Processor Clock=3425.000000MHz                Little Endian

Use these keys to toggle statistics on/off:
  c = CPU          l = CPU Long-term        - = Faster screen updates
  m = Memory       V = Virtual memory       + = Slower screen updates
  d = Disks        n = Network              j = File Systems
  r = Resource     N = NFS                  . = only busy disks/procs
  k = Kernel       t = Top-processes       h = more options
                                     q = Quit
```

# Monitoramento LINUX

- Com a utilização da flag `-f` é possível realizar o armazenamento em arquivo;
- O arquivo de saída do `nmon` é um arquivo de texto com extensão `nmon` e no formato CSV;

# Monitoramento LINUX

- Os comandos são organizados da seguinte forma:

```
$ nmon -f -s "seconds" -c "count"
```

- Onde `-f` é salvar em arquivo (file), `-s` o intervalo entre as capturas e `-c` o número de capturas.

- A saída vai para o diretório atual, no seguinte formato:

```
<hostname>_<date>_<time>.nmon
```

# Monitoramento LINUX

- É possível especificar o local onde será armazenado o arquivo de saída através da flag -m

```
$ nmon -f -s "seconds" -c "count" -m /home/...
```

- E agendar o início do experimento

```
$ 0 0 * * * /usr/local/bin/nmon -f -s 120 -c 720 -m /home
```

0 hora, 0 minutos, \* dias no mês, \* mês, \* dia da semana.

# Monitoramento LINUX

- **Exemplo 1: Monitorar o sistema por um minuto**

1min = 60 segundos;

60s com intervalos entre amostras de 5s;

$60/5 = 12$  coletas;

## **Carga de trabalho:**

```
$ stress -vm 2 -vmbytes 10M -t 50s
```

## **Monitoramento**

```
$ nmon -f -s 5 -c 12 -m /home/carlos/Downloads/nmonchart-master
```

# Monitoramento LINUX

- Exemplo 1: Monitorar o sistema por um minuto

Os dados obtidos via nmon podem ser analisados através de várias ferramentas (excel, R, Mathematica) muitos dados tornam a análise mais precisa, porém implicam em uma grande massa de dados que em sua análise tende a consumir uma grande quantidade de memória;

- [developers.google.com/chart](https://developers.google.com/chart) é uma biblioteca que pode facilitar um pouco se utilizar em conjunto do nmonchart

# Monitoramento LINUX

- Exemplo 1: Monitorar o sistema por um minuto

Os dados obtidos via nmon podem ser analisados através de várias ferramentas (excel, R, Mathematica) muitos dados tornam a análise mais precisa, porém implicam em uma grande massa de dados que em sua análise tende a consumir uma grande quantidade de memória;

- [developers.google.com/chart](https://developers.google.com/chart) é uma biblioteca que pode facilitar um pouco se utilizar em conjunto do nmonchart

# Monitoramento LINUX

- Exemplo 1: Monitorar o sistema por um minuto
  - 1) É preciso baixar e extrair o nmonchart <https://github.com/aguther/nmonchart> e instalar o via yum ou apt-get ksh;
  - 2) Extrair nmonchart-master;
  - 3) Atribuir permissões de leitura e escrita ao nmonchart.sh;
  - 4) Executar  
\$ ./nmonchart nmonfile.nmon output.html

# Referências

- Man-pages do Linux
- Site do iostat:
  - <http://sebastien.godard.pagesperso-orange.fr>
- Jain, Raj. "The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling." *New York: John Willey (1991)*.
- Lilja, David J. *Measuring computer performance: a practitioner's guide*. Cambridge University Press, 2005.