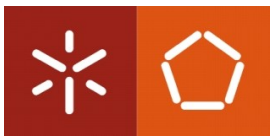
Monitorização

Engenharia de Sistemas da Computação

Computação Paralela Distribuída

Mestrado em Engenharia Informática

Universidade do Minho



Duarte Nuno Ferreira Duarte

pg27715

Índice

Introdução

Este trabalho demonstra a utilização de comandos de monitorização do sistema tais como o vmstat, top e ps. Este comandos são úteis para obter valores de desempenho das aplicações e diferentes nós do Search. Estes comandos são capazes de gerar estatísticas de utilização de uma aplicação, utilizador ou sistema e serão úteis neste caso para obter os dados de uma simples aplicação.

Código de teste

O código apresentado de seguida é o código que será utilizado pelos comandos do sistema.

```
void multiply_matrices() {
        int i, j, k ;
        for (i = 0 ; i < MSIZE ; i++) {
                for (j = 0 ; j < MSIZE ; j++) {
                        float sum = 0.0 ;
                        for (k = 0 ; k < MSIZE ; k++) {
                             sum = sum + (matrix_a[i][k] *
matrix_b[k][j]);
printf("x\n");
                }
               matrix_r[i][j] = sum ;
                }
        }
```

Como é possível ver o código é o mais básico que existe de multiplicação de matrizes não possui qualquer tipo de otimização mas faz o esperado. Apenas o printf é de estranhar mas só aparece ali para que a aplicação decorra tempo suficiente para ser monitorizada.

Dados Obtidos

São apresentados os três comandos que vão ser testados no nó `compute-431-10` e no nó `compute-321-1`.

Compute-431-10

VMSTAT

Antes da aplicação estar a decorrer os dados do VMSTAT eram os seguintes:

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0  11200 44337260 252896 160604    0    0     0     0    0    0  4  0 96  0
```

Como é possivel observer pela tabela acima o numero de processos à espera para executar e os processos que não voltam a executar estão ambos a zero pois não existe nenhum processo por parte do utilizador. Ao nivel da memoria os valores não serão comentados nesta altura. Ao nivel de swap e IO todos os valores se encontram a zero pois não existe qualquer processo a executar. Ao nivel de CPU a utilização é apenas por parte do sistema ao executar o código do kernel.

Durante a execução o VMSTAT já devolvia os seguintes valores:

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0  11200 44331884 253084 161636    0    0     0     0    0    0  4  0 96  0  0
```

Durante a execução é então possível ver que já existe um processo na fila de processos à espera para executar. Em termos de memória a única parte que sofreu alteração considerável foi a da memória livre e isto deve-se à execução da multiplicação de matrizes pois esta aplicação tem alguma exigência ao nível da memoria. Em swap os valores mantêm-se a zero como seria de esperar, não é suposto (para o tamanho utilizado) que haja valores a terem de ir para disco. Ao nível de IO os valores mantêm-se também a zero como seria de esperar pois a aplicação não tem nada que use chamadas de IO. Ao nível de CPU os valores mantiveram-se sendo que aqui eram esperadas mudanças pois a execução da aplicação devia alterar estes valores pelo menos um bocado.

TOP

Antes da aplicação estar a decorrer os dados do TOP eram os seguintes:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	19348	1144	908	S	0.0	0.0	0:01.47	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.04	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:03.90	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:04.29	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.05	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:05.43	watchdog/0
7	root	RT	0	0	0	0	S	0.0	0.0	0:54.14	migration/1
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.05	migration/1
9	root	20	0	0	0	0	S	0.0	0.0	3:49.93	ksoftirqd/1
10	root	RT	0	0	0	0	S	0.0	0.0	0:04.10	watchdog/1
11	root	RT	0	0	0	0	S	0.0	0.0	0:32.03	migration/2
12	root	RT	0	0	0	0	S	0.0	0.0	0:00.05	migration/2
13	root	20	0	0	0	0	S	0.0	0.0	1:18.82	ksoftirqd/2
14	root	RT	0	0	0	0	S	0.0	0.0	0:04.13	watchdog/2
15	root	RT	0	0	0	0	S	0.0	0.0	0:02.49	migration/3
16	root	RT	0	0	0	0	S	0.0	0.0	0:00.05	migration/3
17	root	20	0	0	0	0	S	0.0	0.0	1:00.41	ksoftirqd/3

Como é possível observar pela tabela acima não existem processos a executar neste nó.

Durante a execução o TOP já devolvia os seguintes valores:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4956	pg27715	20	0	6984	3368	364	R	72.9	0.0	0:30.71	app
4646	root	20	0	27096	7372	64	R	35.5	0.0	0:15.06	pbs_mom
119	root	20	0	0	0	0	S	15.0	0.0	1:15.05	events/20
105	root	20	0	0	0	0	S	3.7	0.0	1:19.63	events/6
120	root	20	0	0	0	0	S	1.9	0.0	1:08.79	events/21
4962	pg27715	20	0	15292	1424	828	R	1.9	0.0	0:00.01	top
1	root	20	0	19348	1144	908	S	0.0	0.0	0:01.47	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.04	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:03.90	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:04.29	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.05	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:05.43	watchdog/0
7	root	RT	0	0	0	0	S	0.0	0.0	0:54.14	migration/1
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.05	migration/1
9	root	20	0	0	0	0	S	0.0	0.0	3:49.93	ksoftirqd/1
10	root	RT	0	0	0	0	S	0.0	0.0	0:04.10	watchdog/1
11	root	RT	0	0	0	0	S	0.0	0.0	0:32.03	migration/2

Quando foi colocada a aplicação a executar esta passou a utilizar quase todos os recursos deste nó, é possível observar na primeira linha, a linha correspondente à aplicação, que 72.9 % do CPU passou a ser utilizado pela aplicação e a memória não sofreu qualquer alteração. Um dado curioso é o associado ao VIRT que se trata da quantidade de memoria virtual utilizada pela aplicação e ainda o RES que se trata da memória permanente utilizada pela aplicação.

PS

Antes da aplicação estar a decorrer os dados do PS AUX eram os seguintes:

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
pg27715    4647  0.0  0.0 108580  2096 pts/0    S+   12:01   0:00 -bash
pg27715    4856  0.0  0.0 108444  1936 pts/1    S    12:10   0:00 -bash
pg27715    4999  1.0  0.0 110240  1132 pts/1    R+   12:17   0:00 ps aux
```

Observando a tabela acima conclui-se que o utilizador não tinha nenhuma aplicação dele a decorrer. As únicas tarefas da parte do utilizador eram a própria bash e ainda a chamada do comando ps aux. Esta tabela foi obtida com o 'ps aux' e depois foram filtrados os processos do utilizador para que esta não fosse muito extensa.

Durante a execução o PS AUX já devolvia os seguintes valores:

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
pg27715    4647  0.0  0.0 108580  2096 pts/0    S    12:01   0:00 -bash
pg27715    4856  0.0  0.0 108444  1936 pts/1    S    12:10   0:00 -bash
pg27715    5004 75.0  0.0   6984  3368 pts/0    R+   12:18   0:02 ./app
pg27715    5005  1.0  0.0 110236  1124 pts/1    R+   12:18   0:00 ps aux
```

Enquanto decorria a aplicação é possível olhar para a terceira linha de resultados e ver que a aplicação estava a utilizar 75% do CPU.
Um facto interessante a visualizar corresponde a coluna STAT que apresenta o estado da execução neste caso o S por parte da bash significa que se encontra a espera que um evento conclua e o R por parte da aplicação e da chamada do comando significam que está a executar.

Compute-321-1

VMSTAT

Antes da aplicação estar a decorrer os dados do VMSTAT eram os seguintes:

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0  19620 7758328 109616 104912    2    2     2     3    1    1 41  1 58  0  0
```

Como é possivel observer pela tabela acima o numero de processos à espera para executar e os processos que não voltam a executar estão ambos a zero pois não existe nenhum processo por parte do utilizador. Ao nivel da memoria os valores não serão comentados nesta altura. Ao nivel de swap e IO todos os valores se encontram a valores proximos de zero pois não existe qualquer processo a executar por parte do utilizador. Ao nivel de CPU a utilização é apenas por parte do utilizador mas deve-se ao facto de ter outro comando a utilizer o CPU o resto está em idle.

Durante a execução o VMSTAT já devolvia os seguintes valores:

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 2  0  19620 7755708 109624 104912    2    2     2     3    1    1 41  1 58  0  0
```

Durante a execução é então possível ver que já existem processos na fila de processos à espera para executar. Em termos de memória a única parte que sofreu alteração considerável foi a da memória livre e isto deve-se à execução da multiplicação de matrizes pois esta aplicação tem alguma exigência ao nível da memoria. Em swap os valores mantêm-se a iguais como seria de esperar, não é suposto (para o tamanho utilizado) que haja valores a terem de ir para disco. Ao nível de IO os valores mantêm-se também no mesmo que estavam como seria de esperar pois a aplicação não tem nada que use chamadas de IO excepto os printf referidos acima. Ao nível de CPU os valores mantiveram-se sendo que aqui eram esperadas mudanças pois a execução da aplicação devia alterar estes valores pelo menos um bocado.

TOP

Antes da aplicação estar a decorrer os dados do TOP eram os seguintes:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
19878	nobody	20	0	154m	1416	1088	S	0.3	0.0	114:34.68	gmond
24057	pg27715	20	0	15164	1332	932	R	0.3	0.0	0:00.03	top
1	root	20	0	19348	552	328	S	0.0	0.0	0:00.94	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.04	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:47.51	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	2571:20	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.03	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:03.45	watchdog/0
7	root	RT	0	0	0	0	S	0.0	0.0	0:31.86	migration/1
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.03	migration/1
9	root	20	0	0	0	0	S	0.0	0.0	732:29.16	ksoftirqd/1
10	root	RT	0	0	0	0	S	0.0	0.0	0:02.77	watchdog/1
11	root	RT	0	0	0	0	S	0.0	0.0	1:39.93	migration/2
12	root	RT	0	0	0	0	S	0.0	0.0	0:00.03	migration/2
13	root	20	0	0	0	0	S	0.0	0.0	491:16.04	ksoftirqd/2
14	root	RT	0	0	0	0	S	0.0	0.0	0:04.74	watchdog/2
15	root	RT	0	0	0	0	S	0.0	0.0	0:26.62	migration/3
16	root	RT	0	0	0	0	S	0.0	0.0	0:00.03	migration/3
17	root	20	0	0	0	0	S	0.0	0.0	375:20.75	ksoftirqd/3
18	root	RT	0	0	0	0	S	0.0	0.0	0:02.62	watchdog/3
19	root	RT	0	0	0	0	S	0.0	0.0	0:13.27	migration/4
20	root	RT	0	0	0	0	S	0.0	0.0	0:00.03	migration/4
21	root	20	0	0	0	0	S	0.0	0.0	207:19.13	ksoftirqd/4
22	root	RT	0	0	0	0	S	0.0	0.0	0:03.00	watchdog/4
23	root	RT	0	0	0	0	S	0.0	0.0	0:11.82	migration/5
24	root	RT	0	0	0	0	S	0.0	0.0	0:00.03	migration/5
25	root	20	0	0	0	0	S	0.0	0.0	268:23.96	ksoftirqd/5
26	root	RT	0	0	0	0	S	0.0	0.0	0:02.95	watchdog/5
27	root	RT	0	0	0	0	S	0.0	0.0	0:03.03	migration/6
28	root	RT	0	0	0	0	S	0.0	0.0	0:00.03	migration/6
29	root	20	0	0	0	0	S	0.0	0.0	218:41.67	ksoftirqd/6
30	root	RT	0	0	0	0	S	0.0	0.0	0:02.51	watchdog/6
31	root	RT	0	0	0	0	S	0.0	0.0	0:06.49	migration/7
32	root	RT	0	0	0	0	S	0.0	0.0	0:00.03	migration/7
33	root	20	0	0	0	0	S	0.0	0.0	235:33.38	ksoftirqd/7
34	root	RT	0	0	0	0	S	0.0	0.0	0:02.39	watchdog/7
35	root	20	0	0	0	0	S	0.0	0.0	2:15.92	events/0

Como é possível observar pela tabela acima existem 2 processos a executar neste nó e um deles é o próprio comando.

Durante a execução o TOP já devolvia os seguintes valores:

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
24064 pg27715   20   0  6984 3368  364 R 43.3  0.0   0:03.69 app
    4 root      20   0     0    0    0 S 29.5  0.0   2571:21 ksoftirqd/0
23957 root      20   0 27836 8112   72 S 23.6  0.1   0:03.42 pbs_mom
   36 root      20   0     0    0    0 S  7.9  0.0   6:16.40 events/1
   39 root      20   0     0    0    0 S  5.9  0.0   3:10.84 events/4
   21 root      20   0     0    0    0 S  2.0  0.0 207:19.14 ksoftirqd/4
23956 root      20   0 27836 8720  680 S  2.0  0.1   0:00.29 pbs_mom
    1 root      20   0 19348  552  328 S  0.0  0.0   0:00.94 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.04 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:47.51 migration/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.03 migration/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:03.45 watchdog/0
    7 root      RT   0     0    0    0 S  0.0  0.0   0:31.86 migration/1
    8 root      RT   0     0    0    0 S  0.0  0.0   0:00.03 migration/1
    9 root      20   0     0    0    0 S  0.0  0.0 732:29.19 ksoftirqd/1
   10 root      RT   0     0    0    0 S  0.0  0.0   0:02.77 watchdog/1
   11 root      RT   0     0    0    0 S  0.0  0.0   1:39.93 migration/2
   12 root      RT   0     0    0    0 S  0.0  0.0   0:00.03 migration/2
   13 root      20   0     0    0    0 S  0.0  0.0 491:16.04 ksoftirqd/2
   14 root      RT   0     0    0    0 S  0.0  0.0   0:04.74 watchdog/2
   15 root      RT   0     0    0    0 S  0.0  0.0   0:26.62 migration/3
   16 root      RT   0     0    0    0 S  0.0  0.0   0:00.03 migration/3
   17 root      20   0     0    0    0 S  0.0  0.0 375:20.97 ksoftirqd/3
   18 root      RT   0     0    0    0 S  0.0  0.0   0:02.62 watchdog/3
   19 root      RT   0     0    0    0 S  0.0  0.0   0:13.27 migration/4
   20 root      RT   0     0    0    0 S  0.0  0.0   0:00.03 migration/4
   22 root      RT   0     0    0    0 S  0.0  0.0   0:03.00 watchdog/4
   23 root      RT   0     0    0    0 S  0.0  0.0   0:11.82 migration/5
   24 root      RT   0     0    0    0 S  0.0  0.0   0:00.03 migration/5
   25 root      20   0     0    0    0 S  0.0  0.0 268:23.98 ksoftirqd/5
   26 root      RT   0     0    0    0 S  0.0  0.0   0:02.95 watchdog/5
   27 root      RT   0     0    0    0 S  0.0  0.0   0:03.03 migration/6
   28 root      RT   0     0    0    0 S  0.0  0.0   0:00.03 migration/6
   29 root      20   0     0    0    0 S  0.0  0.0 218:41.67 ksoftirqd/6
   30 root      RT   0     0    0    0 S  0.0  0.0   0:02.51 watchdog/6
   31 root      RT   0     0    0    0 S  0.0  0.0   0:06.49 migration/7
   32 root      RT   0     0    0    0 S  0.0  0.0   0:00.03 migration/7
```

Quando foi colocada a aplicação a executar esta passou a utilizar grande parte dos recursos deste nó, é possível observar na primeira linha, a linha correspondente à aplicação, que 43.3 % do CPU passou a ser utilizado pela aplicação e a memória não sofreu qualquer alteração. Era esperada aqui uma maior utilização de CPU mas isto deve-se ao facto de a medição ter sido feita no inicio da execução. Um dado curioso é o associado ao VIRT que se trata da quantidade de memoria virtual utilizada pela aplicação e ainda o RES que se trata da memória permanente utilizada pela aplicação.

PS

Antes da aplicação estar a decorrer os dados do PS AUX eram os seguintes:

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
pg27715   23884  0.0  0.0 108444  1876 pts/0    S    12:43   0:00 -bash
pg27715   23958  0.0  0.0 108448  1936 pts/1    S+   12:45   0:00 -bash
pg27715   24068  1.0  0.0 110236  1116 pts/0    R+   12:48   0:00 ps aux
```

Observando a tabela acima conclui-se que o utilizador não tinha nenhuma aplicação dele a decorrer. As únicas tarefas da parte do utilizador eram a própria bash e ainda a chamada do comando ps aux. Esta tabela foi obtida com o 'ps aux' e depois foram filtrados os processos do utilizador para que esta não fosse muito extensa.

Durante a execução o PS AUX já devolvia os seguintes valores:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
pg27715  23884  0.0  0.0 108444  1876 pts/0    S    12:43   0:00 –bash
pg27715  23958  0.0  0.0 108448  1936 pts/1    S    12:45   0:00 –bash
pg27715  24069 80.7  0.0   6984  3364 pts/1    R+   12:48   0:03 ./app
pg27715  24070  0.0  0.0 110240  1120 pts/0    R+   12:48   0:00 ps aux
```

Enquanto decorria a aplicação é possível olhar para a terceira linha de resultados e ver que a aplicação estava a utilizar aproximadamente 81% do CPU.
Um facto interessante a visualizar corresponde a coluna STAT que apresenta o estado da execução neste caso o S por parte da bash significa que se encontra a espera que um evento conclua e o R por parte da aplicação e da chamada do comando significam que está a executar.

Conclusão

Foi uma boa experiencia utilizar estes comandos que estão em qualquer sistema LINUX. São comandos extremamente úteis na medida que nos podem auxiliar na monitorização de uma aplicação num ambiente. E saber se esse ambiente é o ideal para ela. São ferramentas que tem grande potencial podendo ser uma ajuda enorme no desenvolvimento de software na medida melhorar a adaptação de uma aplicação a um determinado sistema no objectivo de obter melhores desempenhos.