Ricardo Alexandre Bernardes Rodrigues

Métricas e Ferramentas Livres para Análise de Capacidade em Servidores Linux

Monografia apresentada ao Departamento de Pós Graduação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação Lato Sensu em Administração de Redes Linux para a obtenção do título de especialista em Administração de Redes Linux.

Orientador
Prof. Msc. Sandro Pereira Melo

LAVRAS
MINAS GERAIS – BRASIL
2009

Ricardo Alexandre Bernardes Rodrigues

Métricas e Ferramentas Livres para Análise de Capacidade em Servidores Linux

Monografia apresentada ao Departamento de Pós Graduação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação Lato Sensu em Administração de Redes Linux para a obtenção do título de especialista em Administração de Redes Linux.

Aprovada em 19 de Abril de 2009

_____
Profa. Dra. Marluce Rodrigues Pereira

_____
Prof. Msc. Herlon Ayres Camargo

_____
Prof. Msc. Sandro Pereira Melo
(orientador)

LAVRAS
MINAS GERAIS – BRASIL
2009

Dedico esta monografia à meus pais e a minha esposa.

Agradecimentos

Agradeço a todos que de maneira direta ou indireta me apoiaram no desenvolvimento deste documento e na conclusão desta especialização.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações e Siglas

CLI – *Command Line Interface*

CPU – *Central Processing Unit*

GB – *Gigabytes*

GUI – *Graphical Unit Interface*

HBA – *Host Bus Adapter*

I/O – *Input / Output*

ITIL – *Information Technology Infrastucture Library*

IRQ – *Interrupt Request*

KB - *Kilobytes*

MB – *Megabytes*

PID – *Process ID*

RAM – *Random Access Memory*

SAN – *Storage Area Network*

UID – *User ID*

Resumo

Esta monografia visa abordar conceitos fundamentais envolvidos na análise de capacidade e performance de servidores Linux. Neste contexto são apresentadas as principais métricas utilizadas na medição do consumo de recursos físicos e lógicos de servidores, descrevendo as ferramentas para *CLI* mais apropriadas para este tipo de análise. São também apresentadas técnicas para a identificação de gargalos no consumo de recursos físicos em servidores, através do uso das ferramentas corretas para situações específicas, tema que também é abordado em um estudo de caso que possibilita a interligação e o melhor entendimento dos conceitos e ferramentas apresentados.

Capítulo 1

Introdução

Durante os últimos anos, o Linux tem ocupado um espaço cada vez maior nos *datacenters* das mais variadas corporações pelo mundo. Isto faz com que este sistema operacional seja utilizado para atender as mais diversas finalidades que vão deste sistemas embarcados como *firewall*, telefones celulares e *mainframes*, a servidores com os mais diversos perfis.

Para que seja possível garantir a melhor performance e a manutenção da capacidade em vários perfis de ambientes computacionais sob o sistema operacional Linux, a utilização das ferramentas corretas no processo de monitoração e análise é fundamental para a obtenção dos melhores resultados.

Hoje os administradores podem contar com diversas ferramentas criadas exclusivamente para este fim, para interface gráfica ou de texto.

1.1 Objetivo

Este documento tem como principal objetivo apresentar conceitos e técnicas utilizados na análise de capacidade de servidores Linux, bem como as ferramentas para interface de texto mais utilizadas na monitoração da performance e análise de comportamento do sistema operacional.

Justifica-se o foco em ferramentas para linha de comandos pelo fato de serem o meio de diagnóstico mais acessível aos administradores por acompanharem a maioria das distribuições Linux, além de imporem menor impacto na utilização de recursos, pois não dependem da instalação de agentes e nem utilizam-se de protocolos de monitoração como o SNMP, que por questões de segurança e garantia de performance não são permitidos em muitos ambientes de alta criticidade.

1.2 Motivação

Muitos analistas e administradores de ambientes computacionais, deixam de atentar-se à análise de capacidade e performance de seus ambientes, optando muitas vezes por investimentos em *hardware* e até mesmo na substituição de aplicações para contornar problemas de performance que afetam diretamente o negócio. Muitas vezes um estudo detalhado sobre o comportamento do ambiente, a identificação dos principais gargalos e a utilização das ferramentas corretas pode dar ao ambiente a performance desejada e evitar investimentos desnecessários.

1.3 Estado da Arte

Segundo matéria publicada no jornal ComputerWorld [COMPUTERWORLD, 2008], com a difusão da biblioteca de melhores práticas *ITIL*, muitas empresas brasileiras tem dado maior atenção à gerencia da capacidade de processamento e armazenamento de seus

ambientes computacionais.

Conforme descrito no livro "*Service Delivery*" [OGC, 2001] que faz parte da versão 2 desta biblioteca, este processo passa pela criação de áreas cuja principal responsabilidade é assegurar que exista sempre capacidade de *TI* rentável e que esta esteja alinhada às necessidades atuais e futuras do negócio.

Com o aumento do foco na capacidade e garantia de performance, o uso de ferramentas de monitoração e análise tem se tornado indispensável no trabalho diário dos administradores.

Hoje pode-se contar com diversas ferramentas direcionadas exclusivamente a este fim, gráficas ou para linha de comando, livres ou proprietárias.

Um bom entendimento, e a correta interpretação das ferramentas clássicas para linha de comando portáveis para a maioria dos ambientes Unix ou baseados em Unix já atende a maioria das necessidades de administradores na manutenção da capacidade, performance e identificação de gargalos nestas plataformas.

1.4 Organização dos Capítulos

O capítulo 2 objetiva fundamentar teoricamente os principais conceitos e técnicas envolvidos em processos de análise de capacidade e performance em ambientes Linux.

O capítulo 3 aborda as principais métricas e ferramentas utilizadas na análise de capacidade e performance de servidores Linux, detalhando

suas principais funcionalidades e dando exemplos práticos de sua utilização.

O capítulo 4 descreve as melhores formas de identificação de gargalos e problemas de performance em servidores Linux.

O capitulo 5 apresenta o processo de análise de capacidade e identificação de gargalos em um servidor de banco de dados .

O capítulo 6 é a conclusão final deste trabalho, propondo trabalhos futuros a serem iniciados a partir desta monografia.

Capítulo 2

Fundamentação Teórica

2.1 Conceitos envolvidos na análise de capacidade de servidores Linux

O sistema operacional Linux começou a conquistar o mercado corporativo principalmente no quesito que se referia ao baixo custo de sua implementação versus seu desempenho em pequenos servidores.

Hoje vem sendo adotado cada vez mais por grandes corporações em ambientes de alta complexidade e capacidade de processamento, onde é capaz de oferecer performance, disponibilidade, estabilidade e escalabilidade a um custo inferior se comparado a soluções proprietárias.

Por se tratar de um sistema operacional portável para diferentes perfis de aplicação, como servidores *web*, de *email*, de arquivos ou de banco de dados, o Linux pode não oferecer o máximo de performance em determinado ambiente. Motivo pelo qual se torna essencial, a monitoração do ambiente com foco na performance da aplicação, sistema operacional e *hardware*.

Em algumas situações, pode ser necessária a execução de parametrizações e de otimizações, nas aplicações ou sistema operacional, afim de obter o máximo aproveitamento dos recursos de *hardware* disponíveis.

Vários fatores podem influenciar no desempenho de um ambiente computacional, podendo estes serem lógicos ou físicos.

Dentre os fatores lógicos pode-se destacar o desenvolvimento de aplicações com foco também na performance, o modelo de dados utilizado, o padrão definido para o acesso a discos, parametrização e customização do sistema operacional de acordo com o perfil do ambiente, dentre outros.

Como fatores físicos, pode-se citar o barramento, a velocidade dos processadores, da memória física, a capacidade de armazenamento, a velocidade dos discos, se estes são internos ou externos, o *throughput* de rede, etc.

A análise de capacidade e performance na maioria das vezes se faz necessária para identificar qual ou quais são os principais ofensores em uma possível queda de performance ou estouro na capacidade de processamento de um ambiente. Nestes casos vários fatores são considerados, e o principal deles é o perfil de consumo da principal aplicação que roda neste ambiente.

Apesar de os administradores hoje poderem contar com várias ferramentas para o acompanhamento da performance tanto de aplicações quanto de componentes físicos individuais em servidores, na maioria das vezes estas ferramentas utilizam as mesmas métricas para medição da performance, cabendo ao próprio usuário, após conhecer as principais métricas utilizadas, escolher a ferramenta que melhor se adapte às suas necessidades.

A seguir são descritas as principais métricas utilizadas na medição

da capacidade e performance de servidores Linux, permitindo ao administrador ter fundamentos para melhor escolher as ferramentas a serem utilizadas no desenvolvimento de suas atividades.

2.2 Métricas de performance

Conforme descrito em [CILIENDO, 2007], por se tratar de um sistema operacional de código aberto, o Linux dispõe de uma infinidade de ferramentas para análise de capacidade, e muitas destas ferramentas utilizam-se das mesmas métricas para a medição da performance de um servidor.

Tendo conhecimento das principais métricas utilizadas, caberá ao administrador escolher a ferramenta que melhor atenda as suas necessidades de acordo com suas preferências pessoais, e da quantidade e nível de detalhes de informações que estas fornecerem.

2.2.1 Métricas de processamento

Detalham, através de dados fornecidos pelo próprio sistema operacional, o perfil de consumo dos processadores em um sistema, possibilitando identificar a existência de pontos de contenção neste recurso que possam acarretar na queda de performance de um ambiente [JOHNSON, 2004].

As principais métricas utilizadas no acompanhamento da performance de processadores referem-se ao tempo de *CPU* dedicado a

processos do sistema operacional, da aplicação, o nível de carga imposto ao recurso, a presença de filas de processos aguardando processamento, dentre outros. A tabela 1 detalha as principais métricas utilizadas na monitoração de performance para *CPU*:

Tabela 1: Métricas para utilização de *CPU*

Métrica	**Finalidade**
CPU Utilization	Percentual de *CPU* em uso.
user time	Percentual de *CPU* em uso por processos em nível de usuário.
system time	Percentual de *CPU* em uso por processos em nível de sistema.
waiting	Percentual de *CPU* utilizado por processos a espera pela conclusão de operações de *I/O* do sistema.
idle time	Percentual de *CPU* ocioso, aguardando por tarefas.
nice time	Tempo gasto pela *CPU* no processo de *re-nicing*, que é a mudança na ordem e prioridade dos processos em execução.
load average	Média de carga do ambiente.
runable processes	Processos que estão prontos para serem executados.
blocked	Processos aguardando resposta para finalizarem algum tipo de operação.

context switch	Entradas e saídas de processos e tarefas em *CPU* durante o processamento.

São considerações importantes a respeito das métricas de utilização de *CPU*:

- na maioria dos casos, quando o consumo dos processadores em um servidor permanece acima dos 80% por um longo período, o administrador deverá analisar a existência de gargalos que possam impactar no consumo deste recurso [CILIENDO, 2007];
- como exemplo de operações em nível de sistema tem-se as operações do próprio *Kernel,* incluindo *IRQs*. É importante que o servidor utilize o menor tempo possível de *CPU* com operações em nível de sistema, o contrário pode significar a existência de gargalos no ambiente [CILIENDO, 2007];
- através da métrica *waiting* o administrador pode observar se o seu servidor está gastando tempo excessivo à espera por operações de *I/O*. Altos índices de *wait for I/O* indicam a existência de gargalo no processo de *I/O* do ambiente;
- valores referentes a métrica *load average* podem ser obtidos através da execução dos comandos `top` ou `uptime`, os números exibidos referem-se a valores médios dos processos esperando execução ou em execução nos últimos períodos de 1, 5 e 15 minutos [CILIENDO, 2007];

- um sistema que possui *load averages* significantemente mais altos que o número de *CPUs* está com excesso de carga ou travado por algum gargalo ou falta de recurso critico. Por outro lado, um sistema com *load average* mais baixo que o número de *CPUs* está provavelmente folgado [JOHNSON, 2004];
- a métrica *Runable processes* não deve exceder 10 vezes a quantidade de processadores físicos por um longo período de tempo [CILIENDO, 2007];
- um excessivo número de processos em estado *Blocked* pode representar um gargalo de *I/O* [CILIENDO, 2007];
- a mudança de contexto ocorre quando o *Kernel* suspende um processo que está em execução no processador, transfere-o para algum lugar da memória cache, muda seu contexto recuperando outro processo a partir da memória cache restaurando-o em *CPU*, dando então continuidade ao processamento [CILIENDO, 2005] ;

2.2.2 Métricas de memória

Para que se possa analisar de maneira correta a utilização de memória em servidores Linux, bem como interpretar de maneira correta as principais métricas para monitoração deste recurso, é importante ter conhecimento de como o Linux gerencia sua memória virtual.

A memória virtual, não mapeia apenas áreas de memória física, ela é formada pela memória física e a área de disco destinada ao *swap*. Caso uma aplicação necessite alocar uma grande quantidade de memória, o

gerenciador de memória virtual pode utilizar áreas de swap para este fim, não havendo necessidade que o servidor apresente falta de memória física para que a área de swap seja utilizada. Por este motivo, a utilização da área de swap nem sempre indica a ausência de memória física [CILIENDO, 2005] .

Por padrão, o gerenciador de memória virtual aloca toda a memória física disponível para cache de disco. As aplicações não costumam escrever diretamente em disco, e sim nesta área reservada pelo gerenciador de memória virtual. As informações contidas nestas áreas são transferidas para disco quando necessário, ou quando o tamanho do dado em processamento exceda a área disponibilizada para cache de disco. Por este motivo é comum ver servidores que dispõe de vários *gigabytes* de memória física, apresentarem apenas poucos *megabytes* livres [CILIENDO, 2005] .

A tabela 2 detalha as principais métricas utilizadas na monitoração de memória:

Tabela 2: Métricas de memória

Métrica	**Finalidade**
free memory	Memória livre.
swap usage	Percentual da área de swap utilizada.
buffer e *cache*	Espaço alocado para *cache* de disco ou dispositivos de bloco.

Quando for levantar a memória livre de um sistema, o administrador deve somar o valor da métrica *free memory* ao montante de

memória alocado para *buffer* e *cache* para que se saiba o total de memória disponível para uso.

Como o gerenciador de memória virtual do Linux inicialmente trata todas as áreas de memória como uma área única, nem sempre a utilização da área de *swap* indica gargalos de memória. Neste caso, deve-se observar a métrica *swap in/out* para identificar gargalos de memória. Valores acima de 200 a 300 páginas por segundo durante um longo período indicam possível gargalo de memória [CILIENDO, 2005] .

2.2.3 Métricas de rede

Estas métricas fornecem dados que permitem ao administrador analisar de forma detalhada o perfil da utilização dos dispositivos de rede em um servidor, possibilitando ainda a identificação de gargalos e problemas na infra-estrutura de rede do ambiente. A tabela 3 detalha as principais métricas utilizadas na monitoração de interfaces de rede:

Tabela 3: Métricas para interfaces de rede

Métrica	**Finalidade**
packets received and sent	Informa ao administrador a quantidade de pacotes recebidos e enviados por uma interface de rede.
bytes received and sent	Informa o número de *bytes* recebidos e enviados por uma interface de rede.

colisions per second	Informa o número de colisões ocorridas na rede em que a interface em questão está conectada.
packets dropped	Contador dos pacotes descartados pelo *Kernel*, quer seja por uma configuração de *firewall* ou pela falta de espaço no *buffer* da rede.

2.2.4 Métricas para dispositivos de blocos

Fornecem dados que permitem medir o nível de utilização dos discos em um ambiente, a identificação de tempos de resposta, fila de processos aguardando a execução ou conclusão de operações de *I/O*, *throughput* de dados em operações de leitura e escrita, dentre outras.

A correta interpretação destas métricas permite ao administrador identificar os níveis de utilização dos dispositivos de armazenamento, o perfil de utilização destes recursos para cada tipo de aplicação.

Contenções de *I/O* em disco impactam diretamente na carga imposta aos recursos de *CPU* e memória de servidores, pois, processos aguardando a execução de operações de *I/O* em disco ocupam desnecessariamente tempos de *CPU* e memória enquanto a operação não é concluída [JOHNSON, 2004]. A tabela 4 detalha as principais métricas utilizadas na análise de dispositivos de blocos:

Tabela 4: Métricas para dispositivos de blocos

Métrica	**Finalidade**
iowait	Percentual de *CPU* ocupado por processos aguardando a finalização de operações de *I/O*.
average queue length	Média de processos aguardando em fila para efetuar operações de *I/O* em disco.
average wait	Média de tempo gasto desde a chamada do processo até a conclusão de uma operação de *I/O*.
transfers per second	Número de operações de *I/O* executadas por segundo, sendo consideradas operações de leitura e escrita.
blocks read/write per second	Volume de leituras e escritas por segundo em blocos de 1024 bytes para o *Kernel 2.6*, ou em blocos de 512 bytes a 4 KB para *Kernels* anteriores ao 2.6.
KB per second read/write	Operações de leitura e escrita em um dispositivo de bloco, sendo expressas em *KB,* indicando o volume de dados gravados ou lidos no dispositivo.

Capítulo 3

Ferramentas de monitoração

Por ser um sistema operacional de código aberto e flexível, o Linux dispõe de diversas ferramentas para monitoração de capacidade e performance. Algumas, como o `sar`, `vmstat` e o próprio `top`, são versões para Linux de ferramentas criadas para ambientes Unix, outras foram desenvolvidas especificamente para o Linux [CILIENDO, 2005] .

A principal fonte de informações para a maioria das ferramentas de monitoração desenvolvidas para Linux está no *filesystem* virtual `/proc` que mantém interface direta com o *Kernel* em execução.

É importante que administradores de ambientes Linux estejam familiarizados com tais ferramentas, pois, isso permitirá que entendam melhor o perfil de cada ambiente e localizem com maior facilidade possíveis gargalos de performance.

3.1 Principais ferramentas utilizadas

A tabela 5 mostra as principais ferramentas para *CLI* utilizadas na análise e monitoração de performance:

Tabela 5 : Ferramentas para análise e monitoração de performance

Ferramenta	**Principal métrica/funcionalidade**
`top`	Atividade de processos
`vmstat`	Atividade geral do sistema
`uptime`	*Load Average* do sistema
`ps` / `pstree`	Atividade de processos
`free`	Utilização de memória
`iostat`	Atividade em disco e *CPU*
`sar`	Coleta e reporta atividade geral do sistema
`mpstat`	Atividade em *CPU*

3.2 Detalhamento

3.2.1 O comando `top`

O utilitário `top` acompanha a maioria das distribuições Linux, permitindo que se tenha uma visão geral e dinâmica dos processos do ambiente, bem como dos principais recursos consumidos por estes.

O comando `top` pode ser utilizado como um primeiro passo para que se identifique quais processos estão em execução no ambiente e como eles estão utilizando a capacidade de processamento do mesmo.

Este utilitário oferece recursos adicionais, como a possibilidade de se customizar o tempo de *refresh* da monitoração, sendo o padrão a cada 3 segundos, geração das informações em arquivos no formato texto, exibição de forma customizada, como em ordem hierárquica, dentre

outros. A gravação das informações coletadas em arquivos texto permite que se acompanhe o histórico da execução de processos, podendo-se identificar de forma detalhada o perfil de execução do processo desejado, e caso seja identificado mudanças no comportamento do processo como consumo excessivo de algum recurso de *hardware* o administrador poderá lançar mão de outras ferramentas para detalhar sua análise. A figura 1 mostra o `top` em execução:

```
top - 17:40:12 up 187 days, 15:34,  6 users,  load average: 7.82, 11.95, 12.28
Tasks: 278 total,  10 running, 262 sleeping,   0 stopped,   6 zombie
Cpu(s): 75.7% us,  8.8% sy,  0.0% ni,  0.7% id, 13.8% wa,  0.0% hi,  1.0% si
Mem:  15585296k total, 15575984k used,     9312k free,   100032k buffers
Swap: 20971488k total,   453872k used, 20517616k free, 14220000k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
31513 oracle    25   0 1947m 1.8g 1.7g R   99 11.8   0:52.82 oracle
29223 oracle    25   0 1935m 1.7g 1.7g R   85 11.8   1:17.66 oracle
32546 oracle    18   0 1935m 1.7g 1.7g R   59 11.7   0:03.96 oracle
10091 oracle    15   0 1934m 1.7g 1.7g S   20 11.7   0:06.78 oracle
32412 oracle    18   0 1937m 1.7g 1.7g R   15 11.7   0:01.59 oracle
11059 oracle    16   0 1936m 1.7g 1.7g D   14 11.7 443:46.81 oracle
15674 oracle    16   0 1934m 1.7g 1.7g R   12 11.7   0:22.59 oracle
11011 oracle    15   0 1954m 1.7g 1.7g S    6 11.7  76:09.97 oracle
11007 oracle    15   0 1934m 1.7g 1.7g S    6 11.6  59:42.30 oracle
11003 oracle    15   0 1934m 1.7g 1.7g S    5 11.6  56:16.54 oracle
11005 oracle    15   0 1934m 1.7g 1.7g S    3 11.6  52:14.53 oracle
11009 oracle    15   0 1934m 1.7g 1.7g S    3 11.6  52:02.76 oracle
 4197 root      34  19     0    0    0 R    3  0.0   3325:30 kipmi0
11019 oracle    16   0 1934m 1.7g 1.7g R    2 11.7 442:41.53 oracle
11021 oracle    16   0 1934m 1.7g 1.7g R    2 11.7 441:44.35 oracle
11023 oracle    16   0 1933m 1.7g 1.7g S    2 11.7 442:39.17 oracle
```

Figura 1: `top` em execução

O `top` em execução divide-se em duas sessões. A sessão superior contém informações relacionadas ao status do ambiente como um todo, *uptime*, *load average*, contadores de processos, status da *CPU*, e estatísticas de utilização para memória física e *swap*. A sessão inferior exibe estatísticas em nível de processos [RED HAT, 2009] .

Por padrão os processos são exibidos em ordem crescente no que se refere ao consumo atual de recursos. O administrador pode modificar a prioridade na execução de determinado processo usando o comando `renice` para definir maior ou menor prioridade para cada processo. Caso seja identificado um processo travado, ou ocupando *CPU* em demasia, pode-se matá-lo com o comando `kill`.

Cada coluna exibida pelo `top` possui um significado específico, conforme detalhado abaixo [TOP, 2002]:

- *PID* - Identificação do processo.
- *USER* - Nome do usuário responsável pela execução do processo.
- *PRI* - Prioridade do processo.
- *NI* - Nível de prioridade do processo. O Linux suporta níveis na prioridade de execução de processos que vão de 19 a -20, o valor padrão é zero.
- *SIZE* - Quantidade de memória usada pelo processo em *KB*.
- *RSS* - Quantidade de memória física utilizada, em *KB*.
- *SHARE* - Quantidade de memória compartilhada com outros processos, em *KB*.
- *STAT* - Status do processo: S=*sleeping*, R=*running*, T=*stopped or traced*, D=*interruptible sleep,* Z=*zombie*.
- *%CPU* - Porcentagem de *CPU* utilizada.
- *%MEM*- Porcentagem de memória física utilizada.
- *TIME* - Total do tempo de *CPU* usado pelo processo desde que

foi iniciado.

- *COMMAND* - Comando usado para iniciar o processo.

3.2.1.2 Operando o `top` em modo seguro

Alguns comandos interativos devem ser utilizados com certo cuidado. Para evitar o uso destes comandos o administrador pode contar com a utilização do `top` em modo seguro [ANDRADE, 2009] .

Para executar o `top` em modo seguro ele deve ser iniciado com a opção `-s`, conforme figura 2.

```
$ top -s
```

Figura 2: Execução do `top` em modo seguro

Esta forma de inicialização do `top` pode ser automatizada com a criação de um alias.

3.2.1.3 Limitando interações

O `top` permite que se verifique o comportamento dos processos por um intervalo de tempo definido [ANDRADE, 2009] .

```
$ top -n 5
```

Figura 3: Determinando intervalo e o número de atualizações para o `top`

No exemplo da figura 3, a opção "`-n`", com seu argumento indica 5 interações. Considerando-se o tempo padrão de 3 segundos entre as atualizações, isto significa que esta observação ocorrerá num intervalo total de 15 segundos.

3.2.1.4 Gravando em arquivo

Conforme já informado, o administrador poderá gerar a saída do comando `top` em arquivos texto.

```
$ top -b -n 5 > /tmp/teste.txt
```

Figura 4: Direcionando o `top` para arquivo texto

No exemplo da figura 4 o `top` gravará sua saída em um arquivo de *log*, usando-se a opção `-b` para este propósito, e a opção `-n`, com seu argumento, indica cinco interações [PRITCHARD, 2007]. Os resultados serão redirecionados para `/tmp/teste.txt` para uso posterior.

3.2.1.5 Alterando o tempo de atualização

Caso precise alterar o intervalo de atualização dos processos pelo `top`, o administrador poderá utilizar o comando:

```
$ top -d 10.5
```

Figura 5: Mudando o tempo de *refresh* no `top`

No exemplo da figura 5, o parâmetro "`-d`" fará com que o programa atualize a lista de processos a cada 10 segundos e cinco décimos, podendo estes valores serem alterados de acordo com a necessidade do momento.

Caso se deseje alterar este parâmetro durante a execução do `top`, deve-se pressionar '`d`' ou '`s`', na linha de comandos, isto ativará um *prompt* para seja digitado o tempo de atualização desejado. O uso destes comandos será desabilitado enquanto o `top` estiver sendo executado em modo seguro.

3.2.1.6 Acompanhando um usuário

O `top` permite que se acompanhe um processo de usuários específicos [ANDRADE, 2009] .

```
$ top -u rodrric
```

Figura 6: Listando processos de um usuário específico com o `top`

Sendo iniciado com o parâmetro `-u` seguido do nome do usuário conforme figura 6, o `top` exibirá somente os processos que pertencerem ao usuário informado. Neste caso, pode-se perder algum processo que envolva esta conta de usuário mas que esteja sendo executado com `sudo`,

o que altera o usuário efetivo do processo, ou mesmo um sistema de arquivos desta conta de usuário, mas manipulado por um processo de terceiros. Para garantir a exibição destes processos, deve-se utilizar o parâmetro `-U` em caixa alta.

Caso se deseje visualizar processos de determinado usuário durante a execução do `top`, deve-se pressionar '`u`' ou '`U`' na linha de comandos, informando em seguida o login do usuário ou seu *PID*.

3.2.1.7 Reordenando os dados

O `top` permite que se ordene a lista de processos por determinado campo, os comandos '`F`' ou '`O`' levam à tela para seleção da coluna de ordenação, bastando clicar na letra que indicar o campo desejado para efetivar a alteração [ANDRADE, 2009] . Esta ordenação ocorre por valores internos, podendo gerar uma ordenação incompreensível em campos como o do terminal associado ao processo.

Uma forma mais rápida de ordenar é utilizar os seguintes comandos:

- `M` - Porcentagem de memória
- `N` - Número de identificação de processo
- `P` - Porcentagem de uso da *CPU*
- `T` - Tempo de utilização da *CPU*

3.2.1.8 Alterando a quantidade de processos exibidos

Caso o administrador deseje exibir uma determinada quantidade de processos, ele deve utilizar os comandos ‘`n`’ ou ‘`#`’ para informar a quantidade máxima de processos a serem exibidos, informado `0` para retornar à exibição padrão [ANDRADE, 2009] .

3.2.1.9 Acompanhando processos

Para a exibição de processos específicos, basta iniciar o `top` com o parâmetro “`-p`” seguido pelo *PID* do processo, conforme figura 7.

```
$ top -p 2350 -p 3900
ou
$ top -p 2350, 3900
```

Figura 7: Definindo o *PID* dos processos a serem exibidos pelo `top`

As duas linhas farão com que o `top` acompanhe apenas os processos de *PID 2350* e *3900*, a diferença é que a primeira forma permite listar até 20 processos, não havendo limite de processos na segunda forma.

3.2.1.10 Matando ou enviando um sinal para um processo

Pressionando o comando ‘`k`’ o `top` exibirá um prompt para

indicar o *PID* de algum processo, após receber o *PID* o `top` questionará qual o sinal deverá ser enviado ao referido processo, podendo o administrador informar tando o valor numérico quanto o nome do sinal, sendo por padrão enviado o sinal o 15, *SIGTERM*. O comando '`k`' não poderá ser utilizado se o `top` estiver em modo seguro [ANDRADE, 2009] .

3.2.1.11 Alterando a prioridade de um processo

Para alterar a prioridade de execução de um processo deve-se pressionar o comando '`r`' na linha de comandos para informar o *PID* de um processo e o novo valor de `nice` a ser atribuído a ele [ANDRADE, 2009] . Usuários comuns não podem aumentar a prioridade de seus processos, atribuindo a eles um valor menor do que seu valor corrente. Este comando também não pode ser utilizado quando o `top` estiver em modo seguro.

Para informações completas sobre outros parâmetros, e obtenção de maiores detalhes sobre a utilização deste comando, recomenda-se consulta a sua página de manual.

3.2.2 O comando `uptime`

O comando `uptime`, fornece ao administrador o *load average* dos últimos 1, 5 e 15 minutos. O *load average* representa a carga média imposta à *CPU* do sistema [JOHNSON, 2004].

Através desta métrica o administrador pode identificar de forma rápida problemas que não estejam relacionados à capacidade de processamento do servidor, como problemas de lentidão, ou aumento no tempo de resposta para o usuário apresentados por aplicações em rede.

Caso o servidor de aplicação analisado apresente níveis normais no *load average*, deve-se analisar a possibilidade de existência de problemas no ambiente de rede.

O `uptime` fornece também informações referentes ao tempo em que o servidor não é reiniciado e a quantidade de usuários ativos no ambiente.

```
[capacity@Crato ~]$ uptime
 17:59:01 up 186 days, 15:53,  5 users,  load average: 5.94, 7.27, 8.32
[capacity@Crato ~]$
```

Figura 8: Saída do comando `uptime`

A saída contida na figura 8 informa que a média de processos em execução no último minuto é de 5.94 processos, nos últimos 5 minutos 7.27 processos e nos últimos 15 minutos 8.32 processos.

Estes valores nos permitem chegar a algumas conclusões referentes ao comportamento do ambiente:

- em média, durante o último minuto, 5.95 processos estavam rodando ou esperando por recursos na máquina;
- de maneira geral a carga está tendendo a diminuir, já que as médias vem caindo;

- pela saída do comando `uptime` nota-se que a máquina possui uma carga média de processamento, aproximadamente 7 processos em execução. O fato de este servidor possuir 8 processadores mostra que há menos de um processo em execução por processador, o que em princípio deixa claro a ausência de gargalos no processamento;

Estes valores não levam em consideração o número de processos, sendo também importante salientar que processos em estado de *waiting* podem estar aguardando por qualquer coisa, podendo ser de uma resposta de outra aplicação a disponibilização de um recurso específico, como *CPU*, disco ou rede [JOHNSON, 2004].

Para informações completas sobre outros parâmetros, e obtenção de maiores detalhes sobre a utilização deste comando, recomenda-se consulta a sua página de manual.

3.2.3 O comando `vmstat`

O `vmstat` monitora o desempenho do sistema em tempo real. Ele fornece informações que auxiliam o administrador no acompanhamento da atividade do ambiente através de métricas específicas que vão de *page faults*, *context switches*, atividades de *swap*, ao perfil do consumo de *CPU* [JOHNSON, 2004]. A freqüência na exibição das informações é definida pelo administrador através de parâmetros especificados quando o comando é executado.

```
[capacity@Crato ~]$ vmstat 2 5
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
13  1 453872  14048  72624 14214384    0    0  1459  5293    0     0 59 14 22  5
 8  1 453872  17792  72704 14222560    0    0   434 13468 4553   983 92  8  0  0
 9  0 453872  26432  72752 14220448    0    0  1518 11288 5335  1854 87 12  0  0
 8  1 453872   9728  72832 14214176    0    0  3588 50254 6009  4387 82 17  1  0
11  1 453872   8256  72608 14210272    0    0 12734 49836 5719  2754 94  5  1  1
[capacity@Crato ~]$
```

Figura 9: Saída do comando `vmstat`

O `vmstat` pode fornecer informações de forma regular e em intervalos definidos, permitindo que o administrador tenha uma visão mais dinâmica da atividade do sistema.

Na figura 9, o comando `vmstat` envia uma saída a cada 2 segundos em 5 interações, sendo que, cada linha representa uma imagem do comportamento do ambiente no momento em que ela é exibida [JOHNSON, 2004], não devendo portanto ser comparada diretamente com métricas que exibem médias de comportamento como o *load average* exibido através do comando `uptime`. Mas, ambas as visões são importantes e trazem benefícios na análise de capacidade e performance de um ambiente.

A tabela 6 detalha as métricas contidas na saída deste comando `[VMSTAT, 1994]`:

Tabela 6: Métricas medidas pelo `vmstat`

Sessão	**Coluna**	**Significado**
`procs`	`r`	Número de processos em execução aguardando acesso à CPU.

	`b`	Número de processos bloqueados.
`memory`	`swpd`	Total de memória virtual utilizado, em KB.
	`free`	Total de memória livre, em *KB*
	`buff`	Memória alocada para buffers, em *KB*
	`cache`	Memória alocada para cache de disco, em *KB*
`swap`	`si`	*swap in*, dados transferidos do disco para a memória, em KB.
	`so`	*swap out*, dados transferidos do da memória para o disco, em KB.
`io`	`bi`	blocos lidos em dispositivos, em *KB*.
	`bo`	blocos escritos em dispositivos, em *KB*.
`system`	`in`	Número de interrupções por segundo.
	`cs`	Número de *context switches* por segundo.
`cpu`	`us`	Porcentagem do tempo de *CPU* utilizado em nível de usuário.
	`sy`	Porcentagem do tempo de *CPU* utilizado em nível de sistema.
	`id`	Porcentagem do tempo ocioso de *CPU*.
	`wa`	Porcentagem do tempo de *CPU* ocupado por processos em estado *waiting*.

Conforme informado na sessão `cpu` da tabela 6, o `vmstat` fornece uma visão de como o processador está sendo utilizado pelo sistema operacional, pelas aplicações em execução, o tempo ocioso, bem como a ocupação deste recurso por processos em espera por *I/O*.

O fato de cada aplicação possuir um perfil no consumo de recursos físicos, faz com que na maioria dos casos não exista uma linha de regra

que indique se os valores exibidos por cada uma das métricas exibidas pelo `vmstat` são normais ou não [CILIENDO, 2007].

Neste caso, é importante que o administrador acompanhe o comportamento do ambiente de forma que ele possa entender o comportamento de cada métrica e suas alterações na medida que o tempo de resposta, ou a carga no ambiente comece a aumentar [JOHNSON, 2004]. O `vmstat` auxilia o administrador a identificar qual ferramenta melhor o auxiliará na depuração e identificação das causas de uma possível degradação no desempenho do ambiente.

Como exemplo, pode-se imaginar um cenário onde os usuários reclamam do tempo de resposta de uma aplicação quando é imposta certa carga de trabalho [JOHNSON, 2004].

Através do `uptime` observa-se que a carga média no sistema está relativamente baixa, e através do `vmstat` observa-se ainda que o número de tarefas em execução também está baixo, além de o sistema apresentar certa ociosidade na utilização de *CPU*.

Neste caso, entende-se que o problema esteja relacionado a aplicação, e não a capacidade de processamento do servidor.

Por algum motivo, as requisições do usuário estão sendo bloqueadas a espera de um evento que não se completa.

Algumas aplicações comunicam-se usando a técnica de semáforo enquanto disparam uma solicitação e aguardam sua conclusão [JOHNSON, 2004]. Pode ser que neste caso alguma tarefa foi enviada a um servidor *back-end* ou outra aplicação que, por algum motivo parou sua atividade, resultando em um bloqueio da aplicação do usuário que

aguarda um sinal de que a operação foi concluída para que as informações sejam retornadas ao usuário [JOHNSON, 2004].

Uma situação como esta pode fazer com que o administrador se atente ao servidor de aplicação que por algum motivo não esteja conseguido completar as requisições direcionadas a ele.

Em outro cenário [JOHNSON, 2004], tem-se um alto *load average* no sistema, além de um excessivo consumo de *CPU* chegando a picos de 100% por longos períodos. Neste caso o administrador deverá lançar mão de outra ferramenta que deverá lhe auxiliar a identificar qual, ou quais processos estão consumindo maior parte dos recursos de *CPU* do ambiente.

O `ps` e o `top` poderão fornecer estas informações ao administrador, o comando `ps`, fornecerá ao administrador uma lista de todos os processos em execução, e o `top` fornece esta lista com atualizações automáticas, além de outras informações referentes ao consumo de recursos do ambiente.

Caso o `vmstat` reporte um alto consumo de *CPU* por parte da aplicação que roda no servidor, o administrador pode utilizar alguma aplicação que o permita fazer um *debug* da aplicação, afim de identificar possíveis gargalos em sua atividade. Por outro lado, caso seja reportado alto consumo de *CPU* por parte do sistema operacional, o administrador pode lançar mão de ferramentas como o *strace*, que possibilita ao administrador monitorar as chamadas de sistema que estão sendo executadas. Caso seja reportado pelo `vmstat` uma alta porcentagem de processos aguardando por operações de *I/O*, o administrador poderá usar

os aplicativos `sar` ou `iostat` que lhe auxiliarão na identificação dos dispositivos que estão recebendo maior carga de *I/O*, permitindo que se chegue à fonte desta alta carga.

Para informações completas sobre outros parâmetros, e obtenção de maiores detalhes sobre a utilização deste comando, recomenda-se consulta a sua página de manual.

3.2.4 Os comandos `ps` e `pstree`

O comando `ps` é o comando mais conhecido e utilizado em situações onde é necessário uma análise de ambiente [CILIENDO, 2007].

Ele fornece uma lista dos processos existentes, acompanhado de várias informações importantes sobre o perfil de execução do processo, como o seu *PID*, *UID*, consumo de *CPU* e memória, tempo de execução, status do processo, sua prioridade, linha de comando que gerou o processo, dentre outras que podem ser definidas de acordo com os parâmetros fornecidos ao comando.

Informações essenciais em um processo de análise de capacidade e performance são obtidas com com o uso dos parâmetros "`aux`" conforme exemplo na figura 10.

```
[capacity@martins sa]$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
4051     30568  0.0  0.0  55040  2816 ?        S    17:30   0:00 /bin/sh /opt/ngin/interfaces/
4051     30569  0.1  0.0 319136 52448 ?        Sl   17:30   0:03 /usr/java/j2sdk1.4.2_13/bin/java
oracle   30591 21.4 12.2 8338912 8179296 ?     Ss   18:08   0:00 oraclemartins (LOCAL=NO)
oracle   30773  0.0 12.2 8338976 8186352 ?     Ss   Mar11   0:00 oraclemartins (LOCAL=NO)
oracle   30809 45.9 12.2 8343184 8188560 ?     Ss   18:08   0:00 oraclemartins (LOCAL=NO)
(... a saida continua ...)
```

Figura 10: Saída do comando `ps aux`

Nesta saída tem-se as colunas [PS, 2004]:

- *USER*- Usuário responsável pelo processo em execução.
- *PID* - Process *ID*, identificação do processo.
- *%CPU* - Porcentagem de *CPU* utilizada pelo processo.
- *%MEM* - Porcentagem de memória física utilizada pelo processo.
- *VSZ* - Memória virtual utilizada pelo processo em *KB*.
- *RSS* - Memória física utilizada pelo processo em *KB*.
- *TTY* - Terminal em que o processo está sendo executado.
- *STAT* - Status do processo. Durante sua execução o processo pode passar por vários estados, como *running*, *stopped, sleep*, *zombie*, dentre outros.
- *START* - Hora que o processo foi iniciado.
- *TIME* - Tempo de execução em *CPU* do processo.
- *COMMAND* - Comando que deu origem ao processo.

O comando `pstree` permite que se visualize todos os processos no formato de árvore hierárquica de forma que seja possível visualizar a interligação de cada processo com seus subprocessos [CILIENDO, 2007].

Para informações completas sobre outros parâmetros, e obtenção de maiores detalhes sobre a utilização destes comandos, recomenda-se consulta a sua página de manual.

3.2.5 O comando `free`

O comando `free` exibe o total de memória livre e utilizada no sistema. Fornece também informações sobre *buffers* e *cache* utilizados pelo *Kernel* [CILIENDO, 2005] . Na figura 11 pode-se ver em detalhes o significado de cada informação contida na saída do comando `free -m`.

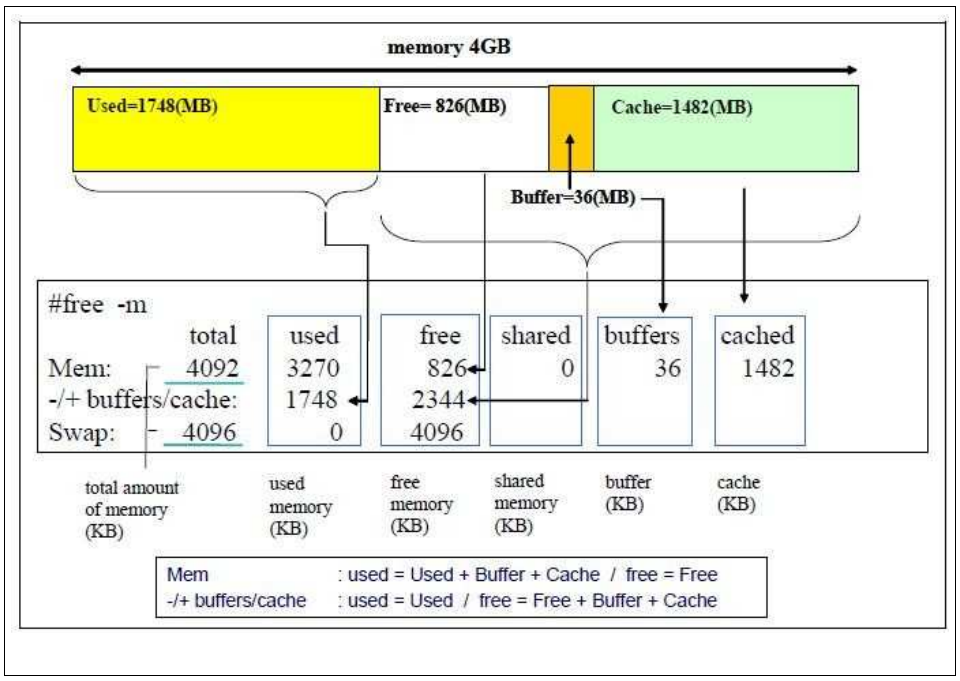

Figura 11: Detalhamento da saída do comando `free -m` [CILIENDO, 2007]

São parâmetros úteis para este comando [FREE, 1993] :

- `-b`, `-k`, `-m`, `-g` Exibe os valores em *bytes*, *kbytes*,

megabytes e *gigabytes*.

- `-l` Detalha a utilização em cada zona da memória.
- `-c <contador>` Exibe a saída do comando `uptime` o número de vezes definido em `<contador>`.

Este comando é mais apropriado para análises rápidas do consumo de memória, ou análises de problemas que estejam em andamento. Através da opção `-s`, o comando `free` pode exibir informações sobre a utilização de memória de forma contínua, possibilitando que se identifique em tempo real alterações no perfil do consumo de memória [RED HAT, 2009] .

3.2.6 O comando `iostat`

O comando `iostat` monitora as atividades de *I/O* em um ambiente, para que se possa saber através das taxas de transferência o quanto os discos estão sendo onerados [JOHNSON, 2004]. As informações fornecidas por este comando permitem ao administrador, caso necessário, balancear o *I/O* imposto aos discos físicos de um servidor.

Este tipo de intervenção geralmente faz-se necessária em servidores de banco de dados que, devido ao perfil das aplicações que o acessam, possuem intensas atividades de *I/O*.

```
[capacity@santaluz ~]$ iostat
Linux 2.6.9-42.ELsmp (santaluz)         03/12/2009

avg-cpu:  %user   %nice    %sys %iowait   %idle
          23.82    0.00    3.74    1.03   71.41

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
cciss/c0d0       23.14       938.64       332.99 2351158464  834091910
cciss/c0d0p1      0.00         0.03         0.00      83940        790
cciss/c0d0p2      3.14        13.99        11.16   35030944   27942680
cciss/c0d0p3     55.44       924.58       321.83 2315936044  806147280
dm-0              1.04         1.32         7.85    3313458   19663824
dm-1              0.10         4.15         0.44   10382706    1110344
dm-2              2.52        12.50        18.55   31301146   46477504
dm-3              1.01         0.57         7.68    1422666   19224992
dm-4             49.40       889.14       280.70 2227179690  703118096
dm-5              0.85        12.94         3.03   32420058    7594888
dm-6              0.51         3.96         3.58    9921634    8958792
```

Figura 12: Saída do comando `iostat`

Como pode ser observado na figura 12, o `iostat` reporta métricas de utilização de *CPU* de forma similar as exibidas pela ferramenta `top`. Estas informações vem seguidas de um relatório detalhado com estatísticas da utilização dos discos, onde cada linha representa um disco lógico configurado no ambiente.

Abaixo o significado das colunas referentes a utilização dos discos [IOSTAT, 2007] :

- *tps* – número de requisições de *I/O* feitas por segundo para o dispositivo.
- *Blk_read/s, Blk_wrtn/s* – número de blocos lidos e número de blocos escritos em um disco por segundo no disco. Os blocos podem possuir diferentes tamanhos, mas, normalmente os tamanhos estão entre 1024, 2048 e 4048 *bytes*, dependendo da

partição. Por exemplo, o tamanho de um bloco da partição `/dev/sda1` pode ser visto através do comando `dumpe2fs`, conforme figura 13.

```
$ dumpe2fs -h /dev/sda1 | grep -F "Block size"
```

Figura 13: Verificando o tamanho dos blocos em uma partição

- *Blk_read, Blk_wrtn* – total de blocos lidos e blocos escritos desde o último *boot*.

O comando `iostat` pode ser usado com diversos parâmetros, o parâmetro `-x` é ideal na análise de performance [IOSTAT, 2007]. A figura 14 mostra a saída do comando `iostat -x`.

```
[capacity@santaluz ~]$ iostat -x
Linux 2.6.9-42.ELsmp (santaluz)         03/12/2009

avg-cpu:  %user   %nice    %sys %iowait   %idle
          23.82    0.00    3.74    1.03   71.41

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
cciss/c0d0   2.12  33.32 14.85  8.30  938.62  332.98   469.31   166.49    54.94     0.55   23.70   2.55   5.90
dm-0         0.00   0.00  0.06  0.98    1.32    7.85     0.66     3.93     8.82     0.02   16.69   3.25   0.34
dm-1         0.00   0.00  0.05  0.06    4.14    0.44     2.07     0.22    45.04     0.01   63.05   4.40   0.04
dm-2         0.00   0.00  0.20  2.32   12.50   18.55     6.25     9.28    12.31     0.67  265.99   2.22   0.56
dm-3         0.00   0.00  0.05  0.96    0.57    7.67     0.28     3.84     8.13     0.79  782.42   1.13   0.11
dm-4         0.00   0.00 14.31 35.09  889.12  280.70   444.56   140.35    23.68     0.29    5.94   0.96   4.73
dm-5         0.00   0.00  0.47  0.38   12.94    3.03     6.47     1.52    18.76     0.02   23.17   3.27   0.28
dm-6         0.00   0.00  0.07  0.45    3.96    3.58     1.98     1.79    14.65     0.03   61.66   5.31   0.27
```

Figura 14: Saída do comando `iostat -x`

As colunas deste relatório são de extrema importância na análise da performance dos discos [IOSTAT, 2007]:

- *rrqm/s, wrqm/s* – número de requisições de leitura e escrita

executadas por segundo.

- *r/s*, *w/s* – número de requisições de leitura e escrita solicitadas por segundo.
- *rsec/s*, *wsec/s* – número de setores do disco lidos e escritos por segundo.
- *rkB/s*, *wkB/s* – número de *KB* lidos e escritos no dispositivo por segundo.
- *avgrq-sz* – média do tamanho das requisições enviadas ao dispositivo. Este valor é exibido em segundos.
- *avgqu-sz* – média do tamanho da fila de requisições enviadas ao dispositivo.
- *await* - média de tempo, em milissegundos, para requisições de *I/O* enviadas ao dispositivo.
- *svctm* – média do tempo de serviço, em milissegundos, para requisições de *I/O* enviadas ao dispositivo.
- *%util* – porcentagem do tempo de *CPU* durante requisições de *I/O* enviadas ao dispositivo. Segundo [JOHNSON, 2004], caso este valor se aproxime de 100% o disco apresenta-se saturado.

Para informações completas sobre outros parâmetros, e obtenção de maiores detalhes sobre a utilização deste comando, recomenda-se consulta a sua página de manual.

3.2.7 O comando `sar`

O `sar` está incluído no pacote de aplicativos *sysstat*, que inclui os comandos `iostat`, `mpstat` e `sar`. O comando `sar` consiste em três aplicações: o próprio `sar`, que exibe os dados coletados, o `sa1` e `sa2`, que são utilizados na coleta e armazenamento das informações [SAR, 2007].

Através do `sa1` e `sa2`, o sistema pode ser configurado para coletar informações e gerar *logs* para futuras análises.

Os arquivos de *log* são gerados em formato binário, geralmente na pasta `/var/log/sa`, onde cada arquivo representa um dia do respectivo mês. Para examinar os resultados de um dia específico, basta referenciar o arquivo de *log* no comando [JOHNSON, 2004]. Por exemplo, caso se deseje informações da utilização de memória referentes ao dia 15 do mês corrente, basta seguir o exemplo descrito na figura 15.

```
[capacity@martins sa]$ sar -r -f /var/log/sa/sa11
Linux 2.6.9-67.EL (martins)     03/11/2009

12:00:02 AM kbmemfree kbmemused  %memused kbbuffers  kbcached kbswpfree kbswpused  %swpused  kbswpcad
12:10:01 AM    112992  66580032     99.83   1321408  54660240 143527776      38368      0.03       224
12:20:01 AM    111552  66581472     99.83   1218944  54779216 143527776      38368      0.03       224
12:30:01 AM    216416  66476608     99.68   1210096  54519968 143527776      38368      0.03         0
12:40:01 AM    193504  66499520     99.71   1132576  54975200 143527776      38368      0.03         0
12:50:01 AM     93024  66600000     99.86   1080480  54979824 143527776      38368      0.03         0
01:00:01 AM    103312  66589712     99.85   1010288  55128448 143527776      38368      0.03         0
01:10:01 AM    104064  66588960     99.84   1075552  54495584 143527776      38368      0.03         0
01:20:01 AM    111744  66581280     99.83   1076784  54395280 143527776      38368      0.03         0
01:30:01 AM     97120  66595904     99.85   1135264  54072608 143527776      38368      0.03         0
(... a saida continua ...)
```

Figura 15: Usando o `sar` com arquivos de *log*

O `sar` coleta informações e gera relatórios sobre diversos perfis de atividade do sistema operacional, como operações de *I/O*, utilização de

CPU, taxa das trocas de contexto e interrupções, paginação em memória física e disco, o uso de memória compartilhada, *buffer* e rede [JOHNSON, 2004].

Baseado nos valores de contagem e nos intervalos dos parâmetros, o `sar` reporta informações determinado número de vezes e no intervalo em segundos fornecido através de parâmetro [JOHNSON, 2004]. Por exemplo, o comando `sar -u 5 10` deverá reportar estatísticas em tempo real sobre o consumo de *CPU* a cada 5 segundos para um total de amostragens, emitindo ao final uma média das informações fornecidas. Abaixo serão tratadas as principais características desta ferramenta.

O `sar` exibe informações a respeito das operações de *I/O* de maneira semelhante ao `iostat`. A figura 16 mostra a saída do comando `sar -b`, que exibe informações de *I/O* de uma máquina.

```
[capacity@martins ~]$ sar -b
Linux 2.6.9-67.EL (martins)     03/12/2009

12:00:01 AM       tps      rtps      wtps   bread/s   bwrtn/s
12:10:01 AM     81.28      8.57     72.71     69.35   1046.95
12:20:01 AM     56.00      0.01     55.99      0.08    800.53
12:30:01 AM     59.91      0.00     59.91      0.00    864.59
12:40:01 AM     62.79      0.03     62.75      1.89    935.62
12:50:01 AM     56.97      0.15     56.82     22.08    897.32
01:00:01 AM     66.92      0.00     66.92      0.00   1062.02
(... a saida continua ...)
```

Figura 16: Exibindo informações de *I/O* com o `sar`

Na coluna *tps* é informado o número total de operações de *I/O*, que são divididas em operações de leitura, exibido na coluna *rtps* e operações de escrita, coluna *wtps* [SAR, 2007]. As taxas para operações

de leitura e escrita são detalhadas nas colunas *bread/s* e *bwrtn/s* [SAR, 2007].

O `sar` também exibe informações a respeito do consumo de *CPU*, para todo o sistema ou individualmente por processador, característica útil para ambientes com mais de um processador. Esta funcionalidade permite que se identifique de forma clara problemas como falta de balanceamento na utilização dos processadores do ambiente, dentre outros. A figura 17 refere-se a saída da coleta de 5 amostragens a cada 2 segundos.

```
[capacity@Crato ~]$ sar -u 2 5
Linux 2.6.9-42.0.8.EL (Crato)   03/12/2009

06:02:39 PM       CPU     %user     %nice   %system   %iowait     %idle
06:02:41 PM       all     29.97      0.00     18.76     36.20     15.07
06:02:43 PM       all     27.33      0.00     15.31     40.86     16.50
06:02:45 PM       all     46.37      0.00     25.03     24.44      4.16
06:02:47 PM       all     63.04      0.00     36.96      0.00      0.00
06:02:49 PM       all     81.97      0.00     18.03      0.00      0.00
Average:          all     49.78      0.00     22.81     20.27      7.14
```

Figura 17: Exibindo informações de *CPU* com o `sar`

Para informações completas sobre outros parâmetros, e obtenção de maiores detalhes sobre a utilização deste comando, recomenda-se consulta a sua página de manual.

3.2.8 O comando `mpstat`

O comando `mpstat`, também faz parte do pacote *sysstat*. Ele reporta métricas de performance para cada *CPU* disponível em um ambiente com múltiplos processadores [MPSTAT, 2007]. No exemplo da

figura 18, são exibidas estatísticas a respeito da utilização de todos os processadores na primeira linha, e de forma individual nas linhas posteriores.

```
[capacity@Crato ~]$ mpstat -P ALL
Linux 2.6.9-42.0.8.EL (Crato)   03/12/2009

06:01:53 PM  CPU   %user   %nice %system %iowait    %irq   %soft   %idle    intr/s
06:01:53 PM  all   59.04    0.00   12.96    4.96    0.06    0.94   22.05   1612.36
06:01:53 PM    0   58.27    0.00   13.43    2.82    0.10    0.80   24.59    290.95
06:01:53 PM    1   59.00    0.00   13.68    7.35    0.07    1.40   18.50    547.42
06:01:53 PM    2   59.45    0.00   13.66    7.58    0.00    1.12   18.18    333.52
06:01:53 PM    3   59.42    0.00   11.08    2.07    0.07    0.43   26.93    440.47
```

Figura 18: Executando o `mpstat` em uma máquina com 4 processadores

O `mpstat`, assim como o `sar` e o `vmstat`, oferece recurso para exibição de informações de forma regular com intervalo em segundos definidos via linha de comando.

Para informações completas sobre outros parâmetros, e obtenção de maiores detalhes sobre a utilização deste comando, recomenda-se consulta a sua página de manual.

Capítulo 4

Análise dos gargalos de performance

4.1 Identificação de gargalos

Segundo [CILIENDO, 2007], os seguintes passos devem ser seguidos em uma estratégia de *tuning*:

1. Conhecer o servidor, sistema operacional e aplicação.
2. Efetuar uma cópia de segurança do ambiente.
3. Monitorar e analisar a performance do sistema.
4. Reduzir o gargalo, buscando identificar sua causa.
5. Corrigir o gargalo, tentando aplicar alterações de forma individual.
6. Voltar ao passo 3 até ficar satisfeito com a performance do sistema.

É fundamental a documentação de cada passo seguido, principalmente as alterações efetuadas e seu impacto na performance. A documentação do problema visa garantir que nas próximas vezes que ele ocorrer seja solucionado de forma rápida e eficaz, evitando também que ele volte a se repetir.

4.2 Gargalos em *CPU*

É importante que ao analisar um possível gargalo na utilização de *CPU* de um ambiente, o administrador veja o ambiente como um todo, e tenha o maior conhecimento possível conhecimento de todos os subsistemas que fazem parte da aplicação, bem como de suas interdependências [CILIENDO, 2007].

Em muitas situações onde acredita-se estar diante de um estouro da capacidade de processamento de um servidor, a causa raiz do problema pode não ser a *CPU*, e sim processos aguardando a resposta de sistemas que muitas vezes nem rodam no servidor em questão [CILIENDO, 2007].

Muitos acreditam que a *CPU* é a parte mais importante de um servidor, pensamento que motiva freqüentes erros no dimensionamento de servidores, onde o uso de *CPU* é superestimado o uso de outros recursos como memória, disco ou recursos de rede é subestimado [CILIENDO, 2007].

Alguns tipos de servidores de aplicação e a maioria dos servidores de banco de dados exigem mais capacidade de processamento de *CPU*. É de extrema importância considerar que o recurso a ser impactado no servidor vai depender diretamente do perfil da aplicação que nele executa ou executará. Como acontece com a maioria dos servidores *WEB* e servidores de aplicação *Java*, como *Weblogic* e *JBoss*, onde a memória física é um recurso mais extremamente impactado no processamento.

4.2.1 Encontrando gargalos em *CPU*

A identificação de gargalos em *CPU* pode ser feita de várias maneiras. Conforme abordado no capítulo 3, o Linux dispõe de uma infinidade de ferramentas para identificar este tipo de gargalo, sendo importante que se conheça o funcionamento de cada uma para que a melhor ferramenta seja escolhida para cada situação.

Além dos comandos `uptime` e `top`, que permite ao administrador identificar processos ofensores na utilização de *CPU*, os comandos "`mpstat -P ALL`" e "`sar -u`" são peças fundamentais no processo de monitoração da utilização de processadores em um servidor.

Estas duas ferramentas permitem que se tenha uma visão detalhada da utilização da *CPU*; em tempo real, através do `mpstat` e do `sar`, ou através de análise histórica com o `sar`.

É importante que o administrador não mantenha monitoração ativa para o mesmo recurso através de várias ferramentas. Esta prática pode resultar em um aumento desnecessário na carga do recurso monitorado [CILIENDO, 2007].

O primeiro passo é estar certo de que o problema de performance apresentado pelo sistema tem como origem a *CPU*, e não outros subsistemas.

Caso o problema de performance esteja realmente relacionado ao processador, pode-se tomar algumas providências afim de sanar o problema, conforme descrito abaixo [CILIENDO, 2007]:

- Deve-se ter certeza de que não há processos desnecessários rodando em *background*, esta informação pode ser obtida usando-se os comandos `ps` ou `top`. Caso sejam identificados processos com este perfil, eles devem ser parados, e se necessários ter sua execução agendada via *cron*, para que não fiquem executando o tempo todo.
- Identificar processos que não sejam críticos, mas que estejam consumindo *CPU* de forma desnecessária. Neste caso estes processos podem ter sua prioridade reduzida usando-se o `renice`.
- Com base no perfil da aplicação em execução, deve-se analisar configuração de servidor que será melhor aproveitada por ela. Por exemplo, aplicações *single-thread* não deverão tirar proveito de um servidor multiprocessado como aplicações *multi-thread. A* mesma situação serve para aplicações com suporte a 32 *bits* rodando em servidores com barramento de 64 *bits*, neste caso, a capacidade de processamento do servidor será subutilizada, resultando em investimento desnecessário.
- O administrador deve manter seu ambiente sempre com *drivers* e *firmwares* atualizados, afim de garantir sempre a melhor utilização de recursos do servidor.

4.3 Gargalos em memória

Em um servidor Linux, vários programas podem ser executados ao mesmo tempo, e alguns processos são mais executados que outros. Alguns processos executam em memória enquanto outros permanecem no

estado *sleeping*, aguardando que alguma situação se concretize para executarem [CILIENDO, 2007]. Enquanto uma aplicação faz acesso direto ao cache, sua performance é superior, pois esta porção da memória física permite acesso mais rápido às informações, evitando a necessidade de acesso a informações contidas em disco, o que torna a operação mais lenta [CILIENDO, 2005] .

O sistema operacional utiliza um algorítimo de escalonamento para controlar quando um programa utilizará espaço em memória física e quando deverá utilizar a área de paginação em disco. A área utilizada no processo de paginação nada mais é que arquivo criado pelo sistema operacional em uma parte do disco físico destinada a este fim, onde são armazenados programas que não estão ativos no momento, ou segundo critérios utilizados no processo de escalonamento, não têm prioridade para execução em memória física [CILIENDO, 2005] .

Assim como ocorre com todos os sistemas operacionais baseados em Unix, no Linux, existem diferenças entre o processo de paginação e *swapping* [CILIENDO, 2005] .

No processo de paginação, páginas são movidas individualmente para a área de *swap*; já o processo de *swapping* refere-se a uma operação de maior porte, onde todo o espaço endereçado a um processo é movido para a área de swap em uma única operação [CILIENDO, 2005] .

O processo de *swapping* pode ter uma ou duas causas [CILIENDO, 2005] :

- Um processo entra no modo *sleep*, aguardando a concretização de

alguma operação para que volte à atividade. Durante este período ele permanece inativo.

- Devido a processos mal desenvolvidos, ou simplesmente por excesso de atividade da aplicação o mecanismo de paginação não consegue páginas livres na memória física, o mecanismo de *swap* é chamado para liberar mais páginas para a aplicação. Este processo aumenta o *I/O* em disco de maneira significativa, degradando rapidamente a performance do servidor.

E em caso de gargalo neste recurso, existem alguns caminhos a serem seguidos, dentre eles estão [CILIENDO, 2005] :

- *Tuning* da área de *swap* através do uso de *bigpages*, *hugetlb* e *shared memory*.
- Aumentar ou reduzir o tamanho das páginas.
- Reduzir a taxa de *page-outs*.
- Limitar recursos destinados a usuários.
- Identificar e desabilitar *daemons* desnecessários.
- E por último, adicionar mais memória.

Conforme descrito em [HORMAN, 2009], existem dois métodos para alteração de parâmetros usados pelo sistema operacional no gerenciamento de memória virtual. O primeiro é através da interface *sysctl*, que permite a modificação de vários parâmetros de *tuning*

diretamente.

Este utilitário permite ao administrador especificar valores para cada parâmetro de *tuning* do gerenciador da memória virtual através da linha de comando, conforme figura 19.

```
$ systcl –w vm.max map count=65535
```

Figura 19: Tuning de parâmetros com o `systcl`

O `systcl` também suporta configurações através do arquivo de configurações `/etc/sysctl.conf`, através do qual podem ser feitas todas as alterações desejadas, que serão carregadas após *restart* do sistema.

O segundo método é através do *filesystem proc*. Os arquivos destinados a este tipo de *tuning* estão no diretório `/proc/sys/vm/` e podem ser lidos ou modificados através dos comandos *cat* e *echo*. Como exemplo, está o uso do comando `cat /proc/sys/vm/kswapd` para visualizar o valor corrente do arquivo *kswapd* [HORMAN, 2009]. O valor deve ser similar ao exibido na figura 20.

```
512 32 8
```

Figura 20: Visualizando o conteúdo do arquivo `/proc/sys/vm/kswapd`

Pode-se modificar este valor através do comando exemplificado na figura 21.

```
$ echo 511 31 7 > /proc/sys/vm/kswapd
```

Figura 21: Alterando o conteúdo do arquivo `/proc/sys/vm/kswapd`

A interface com o *filesystem proc* é um excelente meio para a parametrização da memória virtual quando se busca o máximo de desempenho para um ambiente [HORMAN, 2009].

4.4 Gargalos em discos

O subsistema de discos é muitas vezes causa de gargalos em ambientes, principalmente quando se trata de servidores de banco de dados. Normalmente, quando existe um problema de performance em discos, processos ficam consumindo *CPU* enquanto aguardam a finalização de operações de *I/O* [CILIENDO, 2005] .

A causa mais comum de gargalos em disco está na baixa performance dos dispositivos, quando na maioria das vezes é priorizado a capacidade do ambiente ao invés da performance [CILIENDO, 2005] .

Nestes casos, ao invés de montar um ambiente com discos menores e mais rápidos, o administrador, na maioria das vezes por razões de custos, opta por adquirir discos maiores e com menor performance, o que pode impactar diretamente no tempo de resposta às requisições de *I/O*.

Quando se trata de um ambiente com altas taxas de *I/O*, é importante que se entenda o tipo de carga que os discos receberão. O acesso aos discos será de forma randômica ou seqüencial? Qual o perfil

de *I/O* da aplicação? Mais leitura ou mais escrita? As operações de *I/O* serão longas ou rápidas? Resposta a questões como estas permitirão administrador estar certo de que seu ambiente será montado de maneira a obter a melhor performance [JOHNSON, 2004].

Outra causa também comum de gargalos refere-se a criação de vários discos lógicos no mesmo *array*. Esta prática pode aumentar o tempo de varredura no *array*, resultando também em uma queda na performance [CILIENDO, 2005] .

A atenção na disposição de arquivos pelos discos lógicos também é muito importante para evitar problemas de performance. Uma vez que arquivos muito acessados, tanto para leitura quanto para escrita, devem localizar-se de preferência em *filesystems* que estejam montados em discos diferentes, afim de não sobrecarregar os dispositivos [CILIENDO, 2005] .

Após identificar-se um potencial gargalo em discos, as seguintes providências podem ser tomadas [CILIENDO, 2005] :

- Se os discos recebem uma carga de trabalho de forma seqüencial, é possível que esta sobrecarregue as controladoras, sendo importante analisar a possibilidade de substituição por controladoras mais velozes. Contudo, se os discos recebem carga de forma randômica, é possível que o gargalo esteja na sobrecarga dos discos, sendo conveniente analisar a possibilidade de adição de mais discos.
- Adição de mais discos em um ambiente *RAID*. A distribuição dos dados sobre uma quantidade maior de discos pode trazer ganhos de

performance para operações de leitura e escrita, aumentando também o número de *I/Os* por segundo.

- Utilizar volumes lógicos com *striping* ao invés de utilizar discos físicos ou discos lógicos sem *striping*.
- O aumento na memória física aumenta também a área destinada ao cache de disco, melhorando os tempos de resposta.

4.5 Gargalos na rede

Além dos comandos `sar` e `netstat` que são de extrema utilidade em análises de performance em uma rede, pode-se através do comando *ifconfig* identificar de forma rápida a presença de gargalos na rede [CILIENDO, 2005] . A figura 22 mostra a saída do comando *ifconfig*.

```
[capacity@santaluz ~]$ ifconfig
eth2      Link encap:Ethernet  HWaddr 00:13:21:FB:AF:4E
          inet addr:10.128.242.213  Bcast:10.128.242.255  Mask:255.255.255.0
          inet6 addr: fe80::213:21ff:fefb:af4e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:293383834 errors:0 dropped:0 overruns:0 frame:0
          TX packets:574022771 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20758049675 (19.3 GiB)  TX bytes:866195950305 (806.7 GiB)
          Interrupt:217

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:27355578 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27355578 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6209073800 (5.7 GiB)  TX bytes:6209073800 (5.7 GiB)
```

Figura 22: Saída do comando *ifconfig*

É importante que se saiba que RX representa a transmissão de dados e TX representa a recepção.

Caso sejam identificados valores diferentes de 0 em *errors*, *dropped*, *overruns* ou *collisions*, pode-se estar diante de um gargalo de rede [CILIENDO, 2005] . Neste caso, o primeiro passo seria a verificação de conexões físicas de rede, bem como a configuração em dispositivos como hubs e switches.

Os passos descritos abaixo, podem ser seguidos quando se deseja buscar solução para gargalos em rede [CILIENDO, 2005] :

- Checar se as configurações da placa de rede estão corretas e de acordo com o perfil do ambiente.
- Alterar a organização das *subnets*.
- Utilizar placas de rede mais rápidas.
- Se possível, substituir placas de rede e observar o desempenho.
- Acrescentar mais placas de rede ao ambiente afim de aumentar o *throughput* de dados.

Capítulo 5

Estudo de caso: Análise de capacidade em um servidor de banco de dados

Visando realizar um experimento utilizando as métricas e ferramentas apresentadas, foi escolhido um servidor de banco de dados.

A análise foi efetuada em atendimento a um chamado da área usuária, onde foi reportado queda na performance do ambiente.

Configuração do servidor analisado:

- Sistema Operacional: Red Hat Enterprise Linux AS 3
- Banco de Dados: Oracle 10g
- Processador: 2 x Dual-Core Intel Itanium 2 9000 Series 1.6Ghz
- Memória: 32GB de memória DDR2-533

Nesta análise foram observadas as principais métricas referentes ao consumo de *CPU*, memória, *throughput* de dados via *SAN* e processos, afim de identificar o principal ofensor no consumo de *CPU* do servidor de banco de dados.

5.1 Consumo de *CPU*

Para analisar as métricas de consumo de *CPU* foi utilizado um histórico de 7 dias do comando `sar`, cujos binários referentes ao histórico

de consumo do servidor localizavam-se em `/var/log/sa`.

A figura 23 mostra o comando executado para gerar um arquivo texto com as informações a serem analisadas:

```
$ for i in `ls /var/adm/sa/sa??`
> do
> sar -u -f $i >> sar_historico.txt
> done
```

Figura 23: Script para geração de arquivo com dados históricos do `sar`

A partir das informações contidas no arquivo `sar_historico.txt` foram gerados quatro gráficos através do editor de planilhas *Calc*.

Cada um dos gráficos das figuras abaixo foram gerados a partir da coluna do `sar` referente a respectiva métrica.

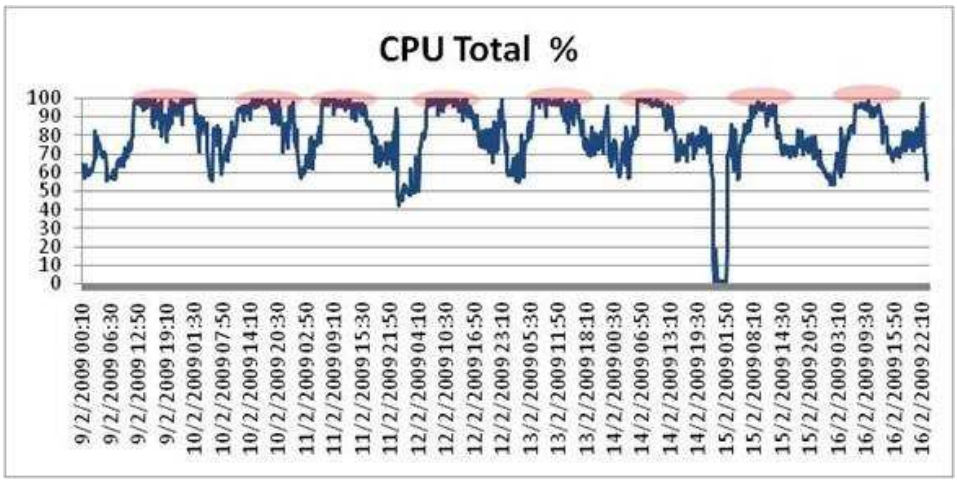

Figura 24: Consumo de *CPU* – Total % (User time + System time + Waiting)

Observa-se através da figura 24 que o servidor apresenta constantes picos de 100% na utilização de *CPU*. Mantendo a média de consumo acima dos 70% que seria uma margem segura de consumo para um servidor de banco de dados.

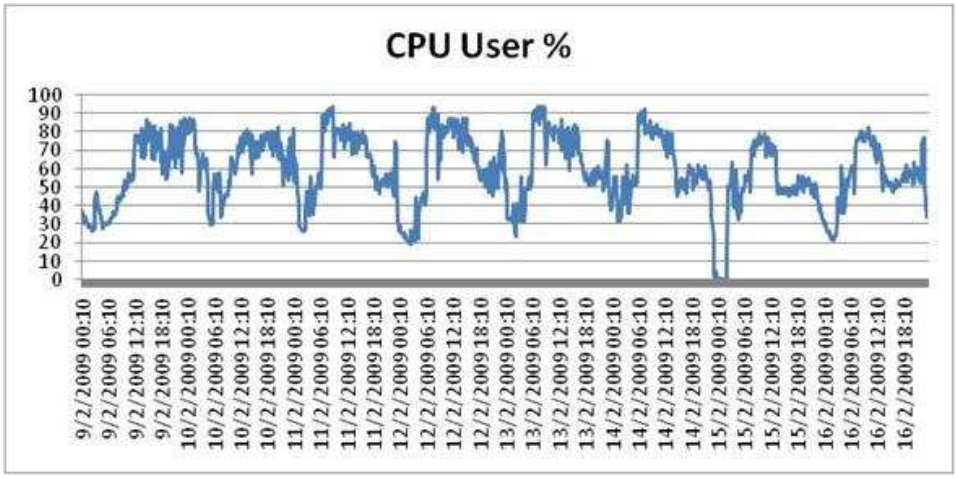

Figura 25: Consumo de *CPU* - *User* %

A figura 25 mostra o alto consumo de *CPU* em nível de usuário, que como o nome diz é nível de execução direcionado pelo *Kernel* a execução de aplicações do usuário [JOHNSON, 2004]. E por tratar-se de um servidor de banco de dados, entende-se que o Oracle é o principal responsável por este consumo, devendo esta informação ser comprovada através da utilização de ferramentas para o mapeamento de processos, como o `ps`, `pstree` ou o `top`.

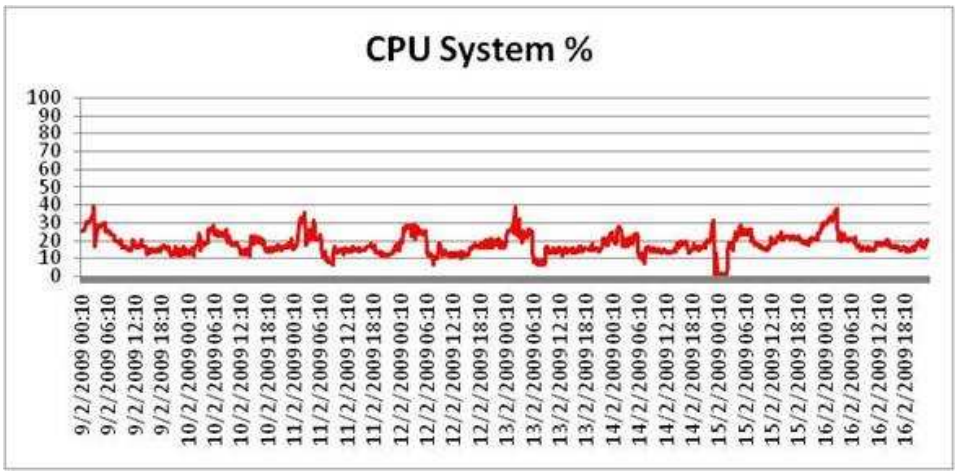

Figura 26: Consumo de *CPU - System* %

Através da figura 26, pode ser visto o consumo de *CPU* em nível de sistema, que refere-se a operações do próprio sistema operacional [JOHNSON, 2004]. Apesar de apresentar eventuais picos de 40%, em média apresenta-se aceitável, não sendo um indicador de gargalos no ambiente.

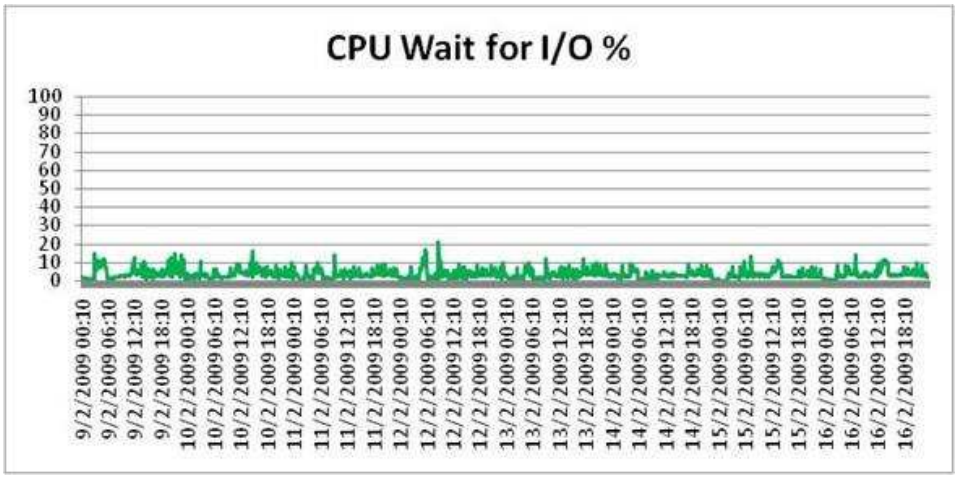

Figura 27: Wait for *I/O* %

Na figura 27 é indicado uma média aproximada de 10% de *Wait for I/O*, significando que aproximadamente 10% do tempo de *CPU* está sendo utilizado por processos aguardando a conclusão de algum tipo de operação de *I/O* [CILIENDO, 2007].

Em ambientes com baixos índices de operações de *I/O* em disco, como servidores *WEB* e a maioria dos servidores de aplicação, espera-se que esta métrica esteja o mais próximo possível de zero.

Neste caso, por se tratar de um servidor de banco de dados com altas taxas de operações de leitura e escrita em disco, considera-se aceitável ocorrências de *Wait for I/O* na média entre 10% e 15% [JOHNSON, 2004]. Valores acima destes podem indicar problemas de contenção de *I/O* no ambiente.

O baixo índice de *Wait for I/O* torna desnecessária, neste caso, a análise dos tempos de resposta de cada disco, análise que pode ser feita através do comando `iostat -x`.

É importante destacar que outras ferramentas para *CLI* podem ser utilizadas em análises de *CPU*, como comando `vmstat`, `mpstat`, e outros abordados no capítulo 3. Cabendo ao administrador escolher aquela que no momento melhor atenda as suas necessidades.

5.2 Consumo de memória

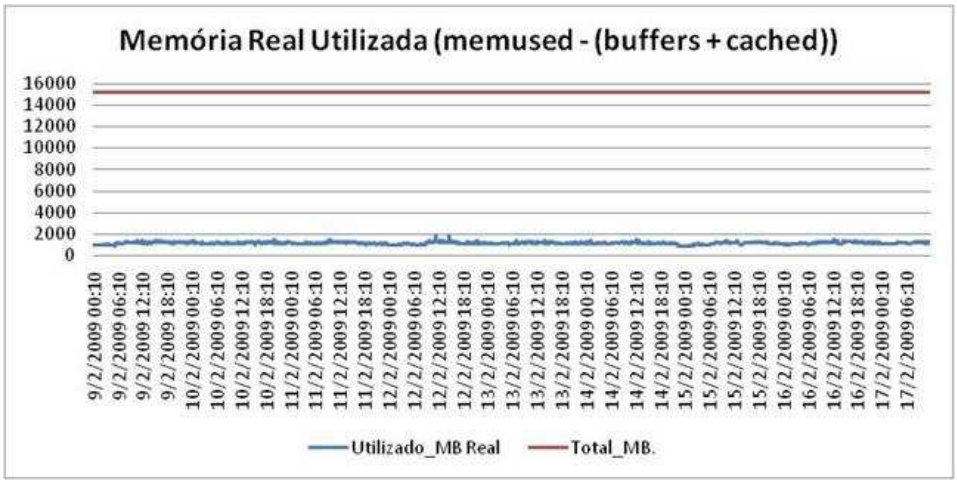

Figura 28: Consumo de memória, em *MB*.

Para medir o consumo de memória foi utilizado o comando `sar` com o parâmetro `-r` que gera um relatório detalhado sobre o consumo de memória. O comando `vmstat` também pode ser utilizado neste tipo de análise, porém o mesmo não possui a funcionalidade de armazenar informações históricas como o `sar`, podendo este recurso ser improvisado através de *scripts*.

Como pode ser visto na figura 28, dos 15*GB* de memória física disponível, o servidor consome aproximadamente 2*GB*, dispondo de 13*GB* livres, deixando claro que a memória física não é um gargalo na performance do ambiente.

Conforme informado no gráfico, chegou-se a memória física real utilizada subtraindo-se a memória alocada para *buffer* e *cache* do total de

memória utilizada.

5.3 *Throughput* de dados

Por se tratar de um servidor conectado a um *array* de discos, é imprescindível a análise do tráfego de dados entre o servidor e os discos externos a ele conectados.

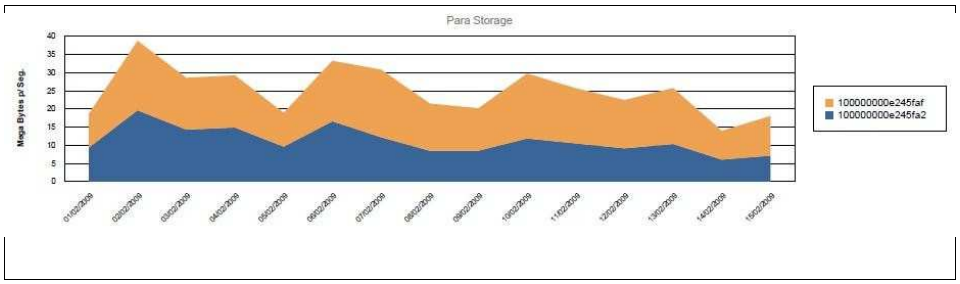

Figura 29: *Troughput* Médio – *MB/s*

Através da figura 29, observa-se que cada dispositivo *Fiber Channel* mais conhecido como *HBA* apresenta tráfego máximo de 20*MB/s* em cada dispositivo. Considerando que, neste caso, a capacidade das portas da *HBA*, do *Switch* e *Storage* é de 2*GB/s*, entende-se que o tráfego de dados entre o servidor e o Storage está normal, não apresentando qualquer tipo de gargalo.

5.4 Comportamento dos Processos

Através do utilitário `top`, observou-se o comportamento dos

processos durante um período de pico no consumo de *CPU*.

```
top - 17:40:12 up 187 days, 15:34,  6 users,  load average: 7.82, 11.95, 12.28
Tasks: 278 total,  10 running, 262 sleeping,   0 stopped,   6 zombie
Cpu(s): 75.7% us,  8.8% sy,  0.0% ni,  0.7% id, 13.8% wa,  0.0% hi,  1.0% si
Mem:  15585296k total, 15575984k used,     9312k free,   100032k buffers
Swap: 20971488k total,   453872k used, 20517616k free, 14220000k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
31513 oracle    25   0 1947m 1.8g 1.7g R   99 11.8   0:52.82 oracle
29223 oracle    25   0 1935m 1.7g 1.7g R   85 11.8   1:17.66 oracle
32546 oracle    18   0 1935m 1.7g 1.7g R   59 11.7   0:03.96 oracle
10091 oracle    15   0 1934m 1.7g 1.7g S   20 11.7   0:06.78 oracle
32412 oracle    18   0 1937m 1.7g 1.7g R   15 11.7   0:01.59 oracle
11059 oracle    16   0 1936m 1.7g 1.7g D   14 11.7 443:46.81 oracle
15447 root      15   0  8928  892  440 S   12  0.0   0:22.59 bpbkar
11011 oracle    15   0 1954m 1.7g 1.7g S    6 11.7  76:09.97 oracle
11007 oracle    15   0 1934m 1.7g 1.7g S    6 11.6  59:42.30 oracle
11003 oracle    15   0 1934m 1.7g 1.7g S    5 11.6  56:16.54 oracle
11005 oracle    15   0 1934m 1.7g 1.7g S    3 11.6  52:14.53 oracle
11009 oracle    15   0 1934m 1.7g 1.7g S    3 11.6  52:02.76 oracle
 4197 root      34  19     0    0    0 R    3  0.0   3325:30 kipmi0
11019 oracle    16   0 1934m 1.7g 1.7g R    2 11.7 442:41.53 oracle
11021 oracle    16   0 1934m 1.7g 1.7g R    2 11.7 441:44.35 oracle
11023 oracle    16   0 1933m 1.7g 1.7g S    2 11.7 442:39.17 oracle
```

Figura 30: Fotografia do `top` em execução

Verifica-se pela figura 30 que no momento da análise a *CPU* do servidor apresentava apenas 0,7% de ociosidade, sendo que 75,7% estava sendo consumido por aplicações em execução no servidor, 8,83% pelo sistema operacional e 13,8% do total de *CPU* sendo utilizado por processos aguardando a conclusão de operações de *I/O*.

Nas primeiras linhas observa-se processos *oracle* como os principais ofensores no consumo de recursos da máquina, além da presença do processo *bpbkar*, que refere-se a aplicação Netbackup que no momento estava em execução e consumindo recursos de CPU e memória.

É importante destacar que, mesmo sendo os processos oracle os principais ofensores no consumo de recursos de *CPU* neste servidor, não

é boa prática a execução de processos de *backup* durante o horário comercial em servidores de banco de dados, pois, o processo de *backup* poderá concorrer diretamente com as operações de I/O, sendo mais um ofensor na degradação da performance do ambiente [JOHNSON, 2004].

Na análise efetuada, o correto uso e interpretação das saídas das ferramentas `sar` e `top`, foi essencial e suficiente para identificar os principais causadores do alto consumo de *CPU* no servidor analisado.

Conforme descrito no capítulo 3, com o uso do `sar` é possível identificar gargalos em *CPU*, memória, *I/O* e rede, podendo outras ferramentas serem utilizadas no detalhamento da análise, caso seja necessário.

Neste caso em específico, o conhecimento das métricas para monitoração de *CPU* e memória permitiram a identificação do principal gargalo no consumo de recursos do servidor analisado.

Como resultado desta análise, foi identificado a existência de dois principais ofensores, sendo que para cada um há recomendações específicas, que, caso sejam atendidas com sucesso deverão reduzir o consumo de *CPU* no servidor e como conseqüência melhorar a performance do ambiente.

Para a otimização no consumo de recursos por parte dos processos Oracle, recomenda-se um trabalho conjunto entre as equipes de suporte a banco de dados e desenvolvimento, com o objetivo de identificar as principais *queries* responsáveis pelo consumo de *CPU*, e buscar formas de otimizar sua execução, através do tratamento de índices ou mesmo da utilização de comandos que garantam melhor performance nos processos

de leitura e gravação no banco de dados.

Referente ao *backup* em execução durante o horário comercial, recomenda-se que seja analisado os tempos totais decorridos nos processos de *backup* incremental e *full*. Sendo constatado que o *backup* está invadindo o horário comercial por não conseguir finalizar no período da madrugada, recomenda-se a revisão da política de *backup* utilizada, bem como de possíveis gargalos na infra estrutura de rede destinada a este fim, garantindo que o *backup* seja iniciado e concluído fora do horário comercial.

Capítulo 6

Conclusão

O conhecimento dos conceitos e ferramentas apresentados neste documento é fundamental em processos de análise de capacidade e performance de ambientes Linux.

A familiarização com as principais métricas para consumo de recursos físicos apresentadas no capítulo 2, e com os melhores caminhos a serem seguidos na identificação de gargalos apresentados no capítulo 4, permitirá que administradores de ambientes Linux sejam capazes de obter os melhores resultados na utilização de todas as ferramentas apresentadas no capítulo 3.

Na análise de capacidade descrita no capítulo 5, a utilização deste conhecimento foi vital na identificação do principal ofensor para a queda de performance do ambiente, abrindo caminho para a regularização da situação, garantindo assim a postergação do investimento para aquisição de novo servidor com maior capacidade.

6.1 Trabalhos Futuros

Esta monografia abre várias possibilidades de trabalhos futuros. Uma proposta de tuning em parâmetros do *Kernel* para diferentes perfis

de utilização de servidores é uma potencial proposta de trabalho futuro, podendo esta ser embasada nesta monografia como ponto inicial, seu título seria *Proposta de tuning em Linux para servidores de Internet e Banco*. Neste contexto é fortemente recomendado o uso da interface `sysctl`.

Outra proposta seria um estudo detalhando de ferramentas gráficas open source destinadas a análise de capacidade e performance, seu título seria *Estudo de ferramentas gráficas para análise de capacidade e performance em servidores Linux*.

8 REFERÊNCIAS BIBLIOGRÁFICAS

[JOHNSON, 2004] JOHNSON, Sandra K. ; HUIZENGA, Gerrit; PULAVARTY Badari; Performance Tuning for Linux Servers; IBM Press; 2004.

[CILIENDO, 2005] CILIENDO, Eduardo; Tuning Red Hat Enterprise Linux on IBM e-server xSeries Servers; ibm.com/redbooks; Jul 2005

[CILIENDO, 2007] CILIENDO, Eduardo; KUNIMASA Takechika; Linux Performance and Tuning Guidelines; ibm.com/redbooks; Jul 2007.

[COMPUTERWORLD, 2008] Redação da Computerworld; Como as empresas dão a partida em projetos de adoção das melhores práticas de TI estabelecidas pelo padrão de governança de ITIL; Jornal Computerworld; Jan 2008; Ano XIII; N° 488.

[OGC, 2001] Office of Government Commerce; Service Delivery Book; 12th impression; Apr 2001.

[TOP, 2002] Warner, James C.; Man Page TOP(1); Sep 2002.

[VMSTAT, 1994] Ware, Henry; Fréderick, Fabian; Man Page VMSTAT(8); Throatwobbler Ginkgo Labs; Jul 1994.

[PS, 2004] Lankester, Branko; Johnson, Michael K.; Man Page PS(1); Jul 2004.

[FREE, 1993] Edmonds, Brian; Man Page FREE(1); Cohesive Systems; Mar 1993.

[IOSTAT, 2007] Godard, Sebastien; Man Page IOSTAT(1); Jul 2007.

[SAR, 2007] Godard, Sebastien; Man Page SAR(1); Oct 2007.

[MPSTAT, 2007] Godard, Sebastien; Man Page MPSTAT(1); Jul 2007.

[ANDRADE, 2009] Cid Rodrigues de Andrade. Blog Tecnologia e Educação. Disponível na Internet via www. url:
http://blog.cidandrade.pro.br/tecnologia/
Informação capturada em 15 de janeiro de 2009.

[HORMAN, 2009] Horman, Neil; Murray, Norm; Understending Virtual Memory. Red Hat Magazine. Disponível na Internet via www. url:
http://www.redhat.com/magazine/001nov04/features/vm/
Informação capturada em 22 de janeiro de 2009.

[PRITCHARD, 2007] Pritchard, Steven; Pessanha, Bruno Gomes; Langfeldt, Nicolai; Stanger, James; Dean, Jeff; Certificação Linux LPI Rápido e Prático; O'Reilly, 2007.

[RED HAT, 2009] Red Hat Linux Manuals; Red Hat Linux System Administration Primer; Disponível na Internet via www. Url: http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/admin-primer/s1-resource-capacity.html
Informação capturada em 25 de janeiro de 2009.