UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE SISTEMAS DE INFORMAÇÃO

MARCELO TOMIELLO ZORZI

TUNING DE SISTEMA OPERACIONAL EM BANCO DE DADOS ORACLE

Monografia apresentada como requisito para a obtenção do Grau de Bacharel em Sistema de Informação da Universidade de Caxias do Sul

Orientadora: Prof.ª Dra. Helena Grazziotin Ribeiro

Caxias do Sul

2015

APROVAÇÃO

MARCELO TOMIELLO ZORZI

TUNING DE SISTEMA OPERACIONAL EM BANCO DE DADOS ORACLE

Monografia apresentada como requisito para a obtenção do Grau de Bacharel em Sistemas de Informação da Universidade de Caxias do Sul.

Banca examinadora:

Presidente/orientador Prof.ª Dra. Helena Grazziotin Ribeiro

Examinadores

Trabalho apresentado e aprovado pela banca examinadora em ___/___/_____

DEDICATÓRIA

Dedico este trabalho à minha família, pelo apoio e compreensão que foram fundamentais para o desenvolvimento deste trabalho.

AGRADECIMENTOS

Meus sinceros agradecimentos; primeiramente, à minha orientadora, Prof.ª Dra. Helena Grazziotin Ribeiro, pela competência em suas orientações; também agradeço à minha família, pelo apoio no desenvolvimento do trabalho.

RESUMO

Atualmente, com a crescente demanda por informações, os Sistemas de Gerenciamento de Banco de Dados (SGBD) devem utilizar os recursos computacionais da melhor maneira possível. Dessa forma, é fundamental conhecer o processo de *tuning.* Para tanto, o presente trabalho aborda o processo de *tuning* de sistema operacional para banco de dados Oracle.

Para compreender o processo de *tuning*, precisa-se entender o funcionamento do banco de dados, mas também é importante compreender como o sistema operacional dispõe os recursos para o banco de dados. Após a compreensão da arquitetura, é possível identificar os gargalos de desempenho em ambos os sistemas.

Para a correção dos gargalos de desempenho, é necessário aplicar as técnicas de *tuning* no sistema operacional e banco de dados, de modo que ambos utilizem os recursos de *hardware* da melhor maneira possível.

Assim sendo, o presente trabalho tem como objetivo realizar o processo de *tuning* utilizando as melhores práticas recomendadas pela Red Hat para o sistema operacional Linux, servindo como base de execução o banco de dados Oracle. Além dos parâmetros de configuração do Linux, o trabalho aponta quais parâmetros e configurações podem ser realizados no banco de dados Oracle para obtenção do melhor desempenho. Por fim, aborda-se os testes de *benchmark* para geração da linha base no processo de *tuning*, bem como validar se técnicas foram aplicadas corretamente.

Palavras-chave: *Tuning*. Banco de Dados. Sistema Operacional. Gargalos. Linux. Oracle.

SUMÁRIO

SUMÁRIO DE FIGURAS

SUMÁRIO DE TABELAS

SUMÁRIO DE GRÁFICOS

SIGLAS

SGBD Sistema de Gerenciamento de Banco de dados

CFS *Complete Fair Scheduler*

CFQ *Complete Fair Queuing*

VFS *Virtual File System*

SGA *Shared Global Area*

PGA *Program Global Area*

RBO *Rule-Based Optimizer*

CBO *Cost-Based Optimizer*

1 INTRODUÇÃO

Com a crescente demanda por informações, os Sistemas de Gerenciamento de Banco de Dados (SGBD) devem estar preparados para atender a carga de trabalho. O SGBD é um sistema de *software* de uso geral que facilita o processo de definição, construção, manipulação e compartilhamento de banco de dados entre diversos usuários e aplicações. Para que o SGBD consiga atender a demanda gerada por usuários e aplicações que o utilizam, o sistema operacional deverá fornecer os recursos de *hardware* necessários para o SGBD (ELMASRI E NAVATHE, 2011).

O procedimento de *tuning* ou sintonia é realizado no sistema operacional e SGBD para que o sistema operacional utilize da forma mais eficiente os recursos de *hardware* disponível, proporcionando o desempenho necessário para que o SGBD consiga atender as aplicações.

O *tuning* de sistema operacional está diretamente ligado ao *hardware* onde o mesmo está instalado, envolvendo recursos tais como acesso à leitura e gravação em disco, desempenho de rede, quantidade de memória e processador. O processo de *tuning* do SGBD consiste em três pilares, sendo eles otimização de consultas e objetos do banco de dados, *tuning* do sistema operacional e *tuning* na arquitetura e memória do SGBD (MACEDO *et al*. ,2013).

A otimização de consultas e de objetos do banco de dados consiste na análise de consultas executadas no banco de dados, bem como a utilização de índices. O *tuning* do sistema operacional baseia-se em ajustes nos parâmetros de sistema que podem ser alterados, visando atingir o melhor desempenho possível com o *hardware* utilizado. O *tuning* de arquitetura e memória do SGBD compõe-se em ajustar parâmetros do SGBD, fazendo com que ele utilize da melhor forma possível os recursos que o sistema operacional está oferecendo.

O trabalho proposto envolve somente dois pilares, o *tuning* do sistema operacional e *tuning* na arquitetura e memória do SGBD, não envolvendo otimização de consultas e objetos no banco de dados, não havendo assim necessidade de alteração nas aplicações que utilizam o SGBD.

O sistema operacional abordado do trabalho é baseado em GNU/Linux, que surgiu em 1991 como projeto pessoal de Linus Torvalds. O Linux é um sistema operacional de código fonte aberto distribuído gratuitamente. A distribuição

GNU/Linux utilizada será *a Red Hat Enterprise Linux 6.5*, sendo a distribuição com o foco no mercado corporativo, sendo ela homologada pelos maiores fabricantes *hardware* e *software* no mercado (NEMETH *et al.*, 2007). O *Red Hat Enterprise Linux* 6.5 será utilizado como base para o SGBD *Oracle Database* versão 11.2.0.4 em sua versão *Standard Edition*.

Para análise e avaliação de resultados obtidos com o procedimento de *tuning* no banco de dados existem ferramentas de teste de *benchmark* TPC (*Transaction Processing Performance Council*), consórcio de vários fabricantes de *software* e *hardware* sem fins lucrativos. Este consórcio define *benchmarks* para vários fins, como processamento de transações *on-line*, comércio eletrônico, apoio à decisão etc. (NOVAIS, 2006)

Existem testes de *benchmark* tais como o TPC-C, que objetiva simular um sistema de OLTP genérico, baseando-se para tal num sistema de gestão de inventário de um fornecedor de materiais (NOVAIS, 2006).

A análise do comportamento do banco de dados é fundamental para a validação do processo de *tuning*. Pelo fato da base de dados possuir a licença *Standard Edition* e para não ocorrer problemas de licenças junto à Oracle, é recomendado pela Oracle a utilização da ferramenta *StatsPack* que acompanha a instalação da versão 11.2.0.4 (ORACLE, 2002). O *StatsPack* é uma ferramenta que cria *snapshots* (fotografia) de estatísticas de *performance* do banco de dados e gera relatórios comparando os *snapshots* criados. Com a geração de relatórios, a ferramenta auxilia o administrador do banco de dados a analisar o estado do banco de dados e identificar problemas de desempenho ou problemas de configuração em determinados parâmetros (DIAS *et al*., 2004).

1.1 QUESTÃO DE PESQUISA E MOTIVAÇÃO

Ambos os *softwares* citados anteriormente, Red Hat Enterprise Linux 6.5 e Oracle Database, são amplamente utilizados na Universidade de Caxias do Sul, onde o trabalho proposto será realizado. A motivação para o desenvolvimento do trabalho se deu com a aquisição de novos *hardwares* (servidores) para sustentar os novos servidores de banco de dados.

A primeira instalação e configuração do banco de dados Oracle foi realizado por uma empresa terceirizada, quando adicionado os servidores de desenvolvimento e homologação para testes nos servidores novos, sendo que o desempenho obtido no Oracle 11.2.0.4 é praticamente o mesmo do Oracle em produção, com a versão 10.2.0.5, e servidores com capacidade significativamente inferior aos adquiridos. As consultas executadas no ambiente novo deveriam possuir um tempo de resposta inferior somente pelo fato da capacidade de processamento ser superior ao ambiente antigo.

Com base nesse resultado obtido através de testes com as aplicações, é necessário avaliar quais modificações podem ser realizadas no sistema operacional e banco de dados Oracle para que o mesmo utilize todo o processamento que os servidores possuem.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Realizar as configurações de *tuning* no sistema operacional e banco de dados para que o mesmo consiga executar as transações da forma mais eficiente, sem a necessidade de alteração nos sistemas que utilizam o banco de dados.

1.2.2 Objetivos específicos

- Identificar os principais gargalos do sistema operacional e banco de dados;
- Identificar as configurações necessárias utilizando como base os guias de melhores práticas para realização do processo de *tuning*;
- Validar o processo de *tuning* utilizando ferramentas de *benchmark*.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está dividido em dez capítulos para uma compreensão deste estudo.

O primeiro capítulo apresenta uma breve introdução sobre o tema abordado e a motivação que levou ao desenvolvimento do trabalho proposto. Além disso, nele constam os objetivos e a estrutura do trabalho.

No segundo capítulo, tem-se o processo de *tuning* como um todo. Após isso, é abordado o processo de *tuning* em sistema operacional Linux e banco de dados Oracle, com base no objetivo do trabalho.

O terceiro capítulo analisa a arquitetura do *Red Hat Enterprise Linux 6*, bem como o funcionamento de cada subsistema presente no sistema operacional. Este capítulo também aborda as ferramentas de monitoramento do *Red Hat Enterprise Linux 6*.

No quarto capítulo é apresentada a arquitetura do banco de dados *Oracle*, sendo analisado o funcionamento da instância e banco de dados, como também os principais processos do banco de dados.

O quinto capítulo considera a ferramenta *StatsPack* para monitoramento do banco de dados Oracle e a interpretação do relatório gerado pela mesma.

No sexto capítulo, tem-se o processo de *tuning* do sistema operacional, sendo utilizadas as melhores práticas para sustentar o banco de dados Oracle.

O sétimo capítulo mostra o processo de *tuning* no banco de dados *Oracle*, quando são utilizadas as práticas para melhora de tempo de resposta e número de transações.

O oitavo capítulo demonstra os conceitos dos testes de *benchmark,* bem como sua utilização em banco de dados e constata-se a proposta de solução com base nos objetivos gerais e específicos.

No nono e penúltimo capítulo, é apresentado os resultados dos testes de *benchmark* bem como a comparação entre os mesmos.

Por fim, são apresentadas as conclusões do presente trabalho.

2 TUNING

Na área da computação, o *tuning* ou sintonia é o processo de realização ajustes com o objetivo de aumentar o desempenho de uma aplicação, sistema operacional ou SGBD. O processo de *tuning* é abordado em três etapas (PRADO, 2014), sendo elas:

- Entender o problema;
- Elaborar o diagnóstico; e
- Aplicar as dicas e técnicas de *tuning.*

2.1 ENTENDER O PROBLEMA

A primeira etapa no processo de *tuning* é o entendimento do problema, sendo isso fundamental para conhecer todos os componentes envolvidos no processo. Nessa etapa são levantados todos os componentes que fazem parte do problema em questão e da forma que eles estão ligados entre si (BENEDETTI, 2004).

Por exemplo, se uma aplicação *web* está apresentando um problema de lentidão de acesso, é necessário avaliar toda a estrutura que sustenta a sua aplicação, pois não necessariamente o problema está na aplicação ou servidor de aplicação, e sim em algum outro componente como a comunicação de rede ou no banco de dados (PRADO, 2014).

2.2 ELABORAR DIAGNÓSTICO

Com o entendimento dos componentes envolvidos no processo de *tuning*, parte-se para a segunda etapa, que é a elaboração do diagnóstico, quando é necessário identificar nessa etapa quais componentes estão apresentando problema de desempenho, sendo eles chamados de *bottlenecks* ou gargalos.

A localização de gargalos é uma tarefa difícil de ser realizada, dependendo da complexidade da aplicação, pois os mesmos podem estar escondidos dentro de um componente. Por esse motivo, a etapa anterior, de entender o problema, é fundamental para a elaboração do diagnóstico e localização dos possíveis gargalos na aplicação.

Utilizando o mesmo exemplo da etapa de entendimento do problema, a elaboração do diagnóstico da aplicação *web* inicia com a análise de todos os componentes que fazem parte da aplicação, como por exemplo, servidor de aplicação, rede ou banco de dados. Para identificação dos gargalos, faz-se necessário a utilização de ferramentas de diagnóstico que todos os componentes disponibilizam.

2.3 APLICAR AS DICAS E TÉCNICAS DE *TUNING*

Como parte final do processo de *tuning*, tem-se a aplicação das técnicas com base no diagnóstico elaborado na etapa anterior. Nessa etapa são aplicados os ajustes em cada componente que está apresentando gargalos, a fim de atingir o desempenho esperado.

A aplicação de técnicas de *tuning* consiste em utilizar as melhores práticas fornecidas pela documentação de cada componente no qual foi detectado o gargalo.

Continuando com o exemplo da aplicação *web* com problemas de acesso, identifica-se a etapa de elaboração de diagnóstico, a qual o servidor não está conseguindo atender as requisições devido a um problema no *buffer* nas conexões de rede. Na etapa de aplicação das técnicas de *tuning*, deve-se ajustar o sistema operacional e servidor de aplicação *web* para que eles consigam atender o número de requisições.

2.4 *TUNING* DE SISTEMA OPERACIONAL

O processo de *tuning* de sistema operacional consiste em fazer com que o sistema utilize de maneira mais eficiente os recursos de *hardware*. Além de estar diretamente ligado ao *hardware*, o *tuning* de sistema operacional é realizado conforme a aplicação e carga de trabalho. Por exemplo, o processo de *tuning* de sistema operacional não será o mesmo caso o servidor seja um *firewall*, serviço responsável pela entrada e saídas de pacotes na rede, que necessita de mais processamento dos recursos de rede do que um servidor com banco de dados que necessita de ajuste de memória, acesso a disco e processamento.

O processo de *tuning* de sistema operacional se baseia em quatro etapas (CILIENDO, 2005), sendo elas:

- Entender os fatores que afetam o desempenho;
- Estabelecer uma linha base com a configuração atual e comparar com desempenho futuro;
- Identificar os gargalos no sistema; e
- Realizar o *tuning* do sistema.

2.4.1 Entender os fatores que afetam o desempenho

Esse processo consiste em analisar quais subsistemas do Linux podem afetar o desempenho do sistema operacional. Os subsistemas são divididos da seguinte forma:

- Escalonador de Processos.
- Arquitetura de memória.
- Escalonador de acesso a Disco.
- Subsistema de rede.

Cada subsistema está abordado no Capítulo 3, bem como a arquitetura do sistema como um todo.

2.4.2 Estabelecer Linha Base

O estabelecimento da linha base é verificar o estado do sistema como ele está e, a partir desse ponto, gerar métricas e traçar os objetivos para a futura configuração. Cada subsistema possui sua métrica distinta, sendo algumas delas (CILIENDO E KUNIMASA, 2007):

- Métrica de Processos: utilização do processador, processos de usuário, processos do sistema, tempo de espera, tempo ocioso, carga de trabalho etc.
- Métrica de Memória: memória livre, *swap* usada, *cache* e *buffer* etc.
- Métrica de Disco: espera de entrada/saída, tempo de espera, média de espera, transferência por segundo etc.
- Métricas de Rede: pacotes recebidos e enviados, quantidades de colisões, pacotes rejeitados etc.

O trabalho proposto não aprofunda a utilização das métricas de sistema operacional citadas anteriormente.

2.4.3 Identificar os Gargalos no Sistema

Esse processo consiste em utilizar as ferramentas de monitoramento, a fim de identificar qual subsistema do sistema operacional possui algum *bottlenecks* ou gargalo. Gargalo é algum problema ou situação que impede desempenho satisfatório do sistema. A identificação dos gargalos é um trabalho difícil que exige profundo conhecimento do ambiente (CILIENDO E KUNIMASA, 2007).

Os gargalos são divididos conforme cada subsistema e cada um exige uma abordagem distinta de processador, de memória, de disco e de rede.

2.4.3.1 Gargalos de Processador

Segundo Ciliendo e Kunimasa (2007), o processador é o recurso principal para servidores de aplicação e banco de dados, e muitas vezes pode ser a origem de gargalos de desempenho. Quando o processador permanece com alta utilização, não quer dizer que o mesmo está sempre trabalhando; ele pode estar esperando o retorno de algum outro subsistema como, por exemplo, o disco.

Para a identificação dos gargalos de processador, é importante a utilização de ferramentas de monitoramento presentes no sistema operacional. As ferramentas de monitoramento são abordadas no Capítulo 3.

2.4.3.2 Gargalos de Memória

O sistema operacional é capaz de rodar várias aplicações ao mesmo tempo. Conforme Ciliendo e Kunimasa (2007), as aplicações realizam as mais variadas requisições de espaço na memória principal, e, em alguns casos, a utilização da memória *swap*, sendo que a mesma está mais detalhada no Capítulo 3.

A identificação dos gargalos de memória se dá realizando o levantamento das aplicações que rodam no servidor e através de ferramentas de monitoramento para verificar como o sistema está alocando os recursos necessários. As ferramentas de monitoramento são abordadas no Capítulo 3.

2.4.3.3 Gargalos de Disco

De acordo com Ciliendo e Kunimasa (2007), o subsistema de disco é, em muitas vezes, o aspecto mais importante relacionado ao desempenho do servidor, tornando-se, talvez, a origem dos gargalos.

O gargalo de disco pode estar escondido. Por exemplo, o alto uso do processador pode estar atrelado à espera do mesmo para operações de entrada/saída no disco. O gargalo relacionado ao subsistema de disco está diretamente realizado ao *hardware* utilizado no servidor, sendo esse subsistema mais sensível à tecnologia utilizada que os demais.

No entendimento de Ciliendo e Kunimasa (2007), o gargalo de disco ocorre devido à preocupação somente na quantidade de dados que podem ser armazenados, mas não em desempenho. Para a correta identificação de gargalos no subsistema de disco, utiliza-se ferramentas de monitoramento abordadas no Capítulo 3.

2.4.3.4 Gargalos de Rede

Gargalos no subsistema de rede podem ser causados por vários fatores, como por exemplo, o subsistema de memória e disco (CILIENDO E KUNIMASA, 2007). Além dos subsistemas do próprio sistema operacional, esse subsistema pode ser afetado por outros fatores como cabeamento de rede ou *switches*. As ferramentas de monitoramento de rede são basicamente analisadores de tráfego, os quais são detalhados no Capítulo 3.

2.4.4 Tuning do Sistema Operacional

Consoante Ciliendo e Kunimasa (2007), o Red Hat Linux fornece uma grande variedade de parâmetros e configurações para maximizar o desempenho do sistema. Os ajustes de parâmetros e configurações são focados para aumento de desempenho em banco de dados, sendo o enfoque do trabalho o banco de dados Oracle, mas podem ser utilizados como base para os demais bancos de dados instalados em sistemas Red Hat. Essas configurações são detalhadas no Capítulo 6.

2.5 TUNING DE BANCO DE DADOS ORACLE

O processo de *tuning* do banco de dados *Oracle*, segundo Macedo *et al.* (2013), consiste nos seguintes objetivos:

- Melhora de desempenho das aplicações e
- Diminuição o tempo de resposta de consultas e operações no banco de dados.

Na interpretação de Macedo et al. (2013), para atender a esses objetivos, o processo de *tuning* de banco de dados é dividido em três etapas:

- Otimização de consultas e objetos do banco de dados;
- *Tuning* de sistema operacional; e
- Ajuste na arquitetura de memória do banco de dados.

2.5.1 Otimização de consultas e objetos do banco de dados

Na análise de Macedo *et al.* (2013), a otimização de consultas e objetos do SGBD consiste em avaliar a estrutura lógica do banco de dados, tais como consultas realizadas, criação de índices e ajuste de consultas, sendo necessário um trabalho em conjunto com o fornecedor de *software* específico como, por exemplo, um sistema de gestão hospitalar. O trabalho proposto não contempla essa etapa no processo de *tuning*.

2.5.2 Tuning de sistema operacional

Na visão de Macedo *et al.* (2013), o *tuning* de sistema operacional é realizado através dos ajustes de parâmetros do sistema para que o banco de dados utilize de forma mais eficiente os recursos de *hardware*. *Tuning* de sistema operacional é abordado com mais detalhes no Capítulo 6.

2.5.3 Ajuste na arquitetura no banco de dados

Essa etapa consiste em ajustes nos parâmetros do banco de dados, a fim de fazer com que o mesmo opere da maneira desejada (MACEDO *et al*. ,2013). Os

ajustes na arquitetura do banco de dados Oracle são analisados de forma mais concisa no Capítulo 4.

2.5.4 Processo de Tuning

Conforme o *Oracle Database Performance Tuning Guide* (2014), o processo de *tuning* no banco de dados pode ter duas abordagens:

- Monitoramento Proativo.
- Eliminação dos gargalos.

2.5.4.1 Monitoramento Proativo

De acordo com o *Oracle Database Performance Tuning Guide* (2014), o monitoramento proativo também pode ser chamado de *tuning* proativo. Normalmente, ele é realizado em períodos agendados, sendo feita análise nas estatísticas de desempenho, a fim de verificar alguma mudança no comportamento do ambiente.

O monitoramento não necessariamente resulta em alteração nas configurações do banco de dados. Essas alterações são realizadas somente se for detectado algum problema de desempenho (*Oracle Database Performance Tuning Guide*, 2014). É comum, com o passar o tempo, ocorrer degradação de desempenho, sendo possível, através de monitoramento proativo, acompanhar a degradação e tomar as medidas necessárias para controlar a situação.

A recomendação do *Oracle Database Performance Tuning Guide* (2014) é somente realizar ações de *tuning* nas situações que realmente se faz necessário, caso contrário, poderá ocorrer perda de desempenho. No Capítulo 5, comenta-se sobre a ferramenta *StatsPack* para monitoramento de estatísticas do banco de dados.

2.5.4.2 Eliminação dos Gargalos

Para o *Oracle Database Performance Tuning Guide* (2014), a eliminação dos gargalos consiste em sua identificação e em sua eliminação. O processo de *tuning*

nesse modelo melhora a utilização dos recursos que estão com gargalos, eliminando os mesmos.

Para realização dessa tarefa, o *Oracle Database Performance Tuning Guide* (2014) recomenda as seguintes ações:

- Realizar alteração na aplicação ou na forma como ela opera;
- Realizar ajustes no banco de dados *Oracle*; e
- Realizar ajustes no *hardware*.

No trabalho proposto, o foco é realizar ajustes no banco de dados *Oracle*, sendo que, de acordo com o *Oracle Database Performance Tuning Guide* (2014), o método mais efetivo para eliminar gargalos é alterar a aplicação.

2.6 TUNING DOS COMPONENTES

O processo de *tuning* é uma tarefa difícil e exige profundos conhecimentos de *hardware*, sistema operacional e aplicação (BENEDETTI, 2004). O objetivo do trabalho é a realização do processo de *tuning* no banco de dados Oracle, sendo ele executado no sistema operacional o *Red Hat Enterprise Linux*.

Com base no modelo proposto por Prado (2014), no qual a primeira etapa consiste no entendimento do problema, implica em conhecer os componentes relacionados, sendo que todas as aplicações têm relação direta com o sistema operacional onde ela está instalada. A primeira abordagem é o entendimento de como o sistema operacional faz a comunicação entre o *hardware* e as aplicações instaladas nele.

Após o entendimento o sistema operacional, é necessário a análise no banco de dados *Oracle*, a fim de compreender a relação entre ele e o sistema operacional.

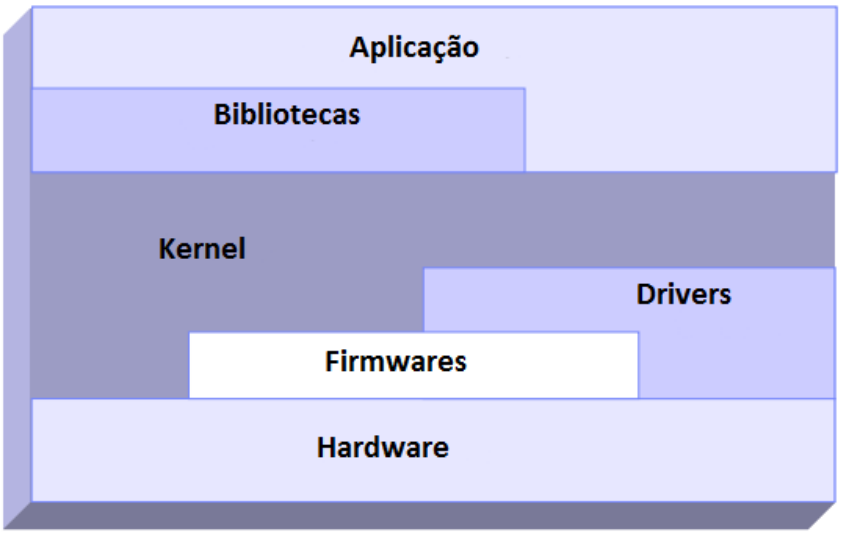
3 SISTEMA OPERACIONAL RED HAT

Para a realização do processo de *tuning* no sistema operacional, é necessário o entendimento da arquitetura do mesmo. A partir da compreensão de como o *Linux* realiza a comunicação entre o *hardware* e as aplicações como, por exemplo, o banco de dados Oracle, é possível entender o problema e elaborar o diagnóstico, sendo que essas atividades são base do processo de *tuning* (PRADO, 2014).

3.1 ARQUITETURA LINUX

A arquitetura do sistema *Linux* é divido em aplicação, biblioteca, *kernel*, *drivers*, *firmware* e *hardware*, de modo que cada uma delas pode impactar no desempenho do sistema operacional (CILIENDO E KUNIMASA, 2007). A Figura 1 ilustra as camadas que podem gerar um problema de desempenho.

Figura 1 – Camadas do Linux



Fonte: Traduzido de (CILIENDO E KUNIMASA, 2007, p.15).

O processo de *tuning* pode ser realizado em uma determinada camada do sistema operacional como, por exemplo, na aplicação, mas uma biblioteca antiga instalada no sistema pode influenciar na perda de desempenho da aplicação. Para a realização do processo de *tuning*, é necessário o entendimento de como o sistema

operacional gerencia os recursos disponíveis e cada subsistema poderá ser ajustado, dependendo o cenário proposto.

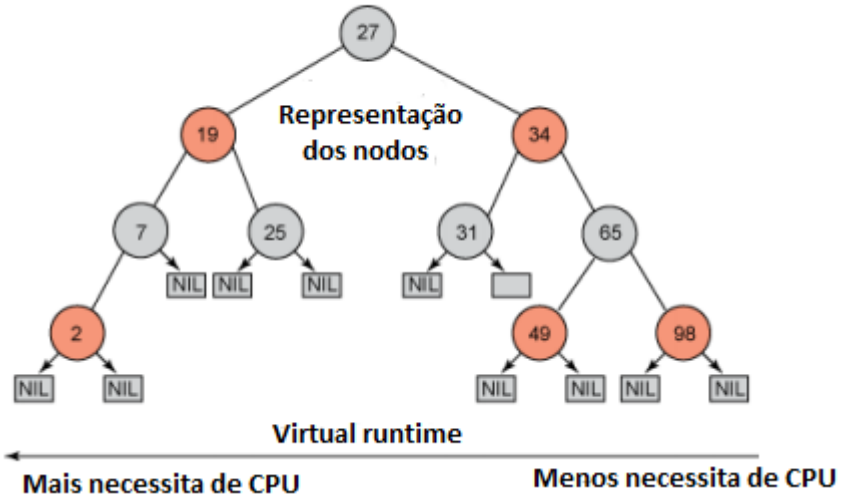
3.2 ESCALONADOR DO PROCESSADOR

O escalonador do processador é componente responsável por manter o processador sempre ocupado, atendendo a processos e *threads* em execução. O *Red Hat 6* utiliza o escalonador de processos *Completely Fair Scheduler* (CFS ou Escalonador Completamente Justo), sendo sua proposta principal manter o balanceamento justo entre os processos com base no tempo de execução de cada tarefa (JONES, 2009). Dessa forma, o CFS faz a distribuição de uma determinada quantidade de tempo para cada processo realizar sua tarefa.

O CFS utiliza o *virtual runtime* para manter o balanceamento de quanto tempo as tarefas necessitam de acesso ao processador. O *Virtual runtime* do processo com valores baixos necessita de mais tempo de acesso ao processador, e tarefas com o *virtual runtime* com valores mais altos necessitam de menos tempo do processador (JONES, 2009). Para garantir que os processos que estão esperando, por exemplo, algum recurso de rede ou disco, não recebam a mesma quantidade de tempo do que eles normalmente precisam, usa-se o processo chamado *sleeper fairness* (justiça para dorminhoco) (JONES, 2009).

Para manter o balanceamento dos processos que estão na fila de execução, o CFS utiliza a *red-black tree* (árvore rubro-negra), que é uma árvore binária de pesquisa contendo um *bit* extra em cada nó para as cores vermelha e preta, tendo por objetivo garantir que as operações demoram O(log n), sendo *n* o número de nós na árvore, nos piores casos. A figura 2 ilustra como o CFS realiza o balanceamento da árvore utilizando o *virtual runtime* como índice.

Figura 2 – Representação do CFQ



Fonte: Traduzido de (JONES, 2009, p.3)

Cada nó da árvore corresponde a uma tarefa requisitando acesso ao processador, sendo que as tarefas inseridas no lado esquerdo da árvore são as que necessitam de mais tempo para serem realizadas, e as tarefas inseridas no lado direito da árvore necessitam de menos tempo de acesso ao processador.

Para manter o tempo de execução justo entre todas as tarefas, o escalonador verifica o nó mais à esquerda da árvore e o coloca na fila de execução. Adiciona-se o tempo de execução no *virtual runtime* e o nó é inserido novamente à árvore no seu lado direito, somente se o processo está apto para execução (JONES, 2009). Assim, o escalonador garante o balanceamento do tempo para as tarefas com os mais variados tempos de execução, fazendo com que o processador consiga atender todas de forma mais eficiente.

O CFS não utiliza priorização propriamente dita, pois para essa situação ele utiliza o *group scheduling* (grupo de escalonamento). Essa funcionalidade faz com que os tempos de um determinado grupo de tarefas sejam divididos entre os membros do grupo de escalonamento com base em uma hierarquia (JONES, 2009), fazendo com que as tarefas do grupo utilizem o mesmo padrão de tempo.

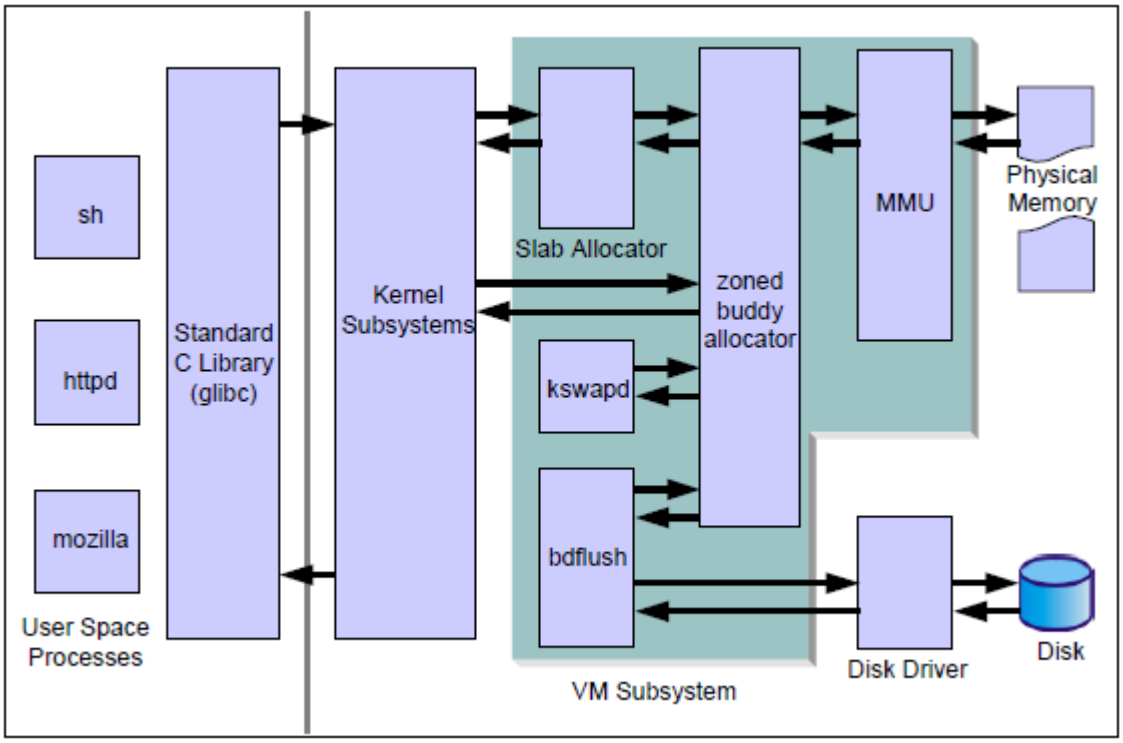
3.3 ARQUITETURA DE MEMÓRIA

Todo o gerenciamento de memória realizado pelo Linux é realizado pelo *virtual memory manager* ou gerenciador de memória virtual, ou seja, as aplicações

não possuem acesso direto à memória principal. É o subsistema responsável por receber as requisições vindas das aplicações que necessitam de acesso à memória e entregar um mapeamento no *virtual memory* ou memória virtual (CILIENDO E KUNIMASA, 2007). O *virtual memory manager,* dependendo da quantidade de memória que a aplicação necessita, pode não necessariamente mapear a memória principal, e sim um espaço no disco chamado de *swap* ou troca. *Swap* é o espaço utilizado para que o sistema operacional consiga mapear através do *virtual memory manager* mais memória do que ele realmente possui.

Outra característica do *virtual memory manager* é fazer com que as aplicações não gravem diretamente no subsistema de disco, mas sim no *cache* ou *buffer* (CILIENDO E KUNIMASA, 2007). Esse processo só não ocorre quando o arquivo a ser gravado for maior que o *cache* disponível. Quando a aplicação grava no *cachê*, existe um processo do sistema operacional que se chama *bdflush* que faz com que os dados que estão no *cache* sejam gravados no disco (CILIENDO E KUNIMASA, 2007). A Figura 3 ilustra o funcionamento do subsistema de memória.

Figura 3 – Arquitetura de Memória



Fonte: Traduzido de (CILIENDO E KUNIMASA, 2007, p.27).

O gerenciamento de memória do Linux é muito eficiente, sendo que a configuração padrão do *virtual memory* utiliza quase toda a memória livre para o *cache* e disco, mantendo livre uma média de 20 *megabytes*, não importando o tamanho da memória. Essa técnica é utilizada para acelerar as operações de entrada/saída, gravando primeiro da memória principal que é mais rápida e, quando o processo terminar, o *bdflush* realiza a gravação em disco (CILIENDO E KUNIMASA, 2007).

3.4 ESCALONADORES DE ACESSO A DISCO

O *Linux* utiliza quatro escalonadores de acesso a disco: CFQ, *Deadline*, *Anticipatory* e *Noop*. Essa característica se dá pelas inúmeras aplicações e trabalhos que o sistema pode realizar, sendo que cada escalonador pode apresentar um melhor desempenho, dependendo do ambiente ou carga de trabalho (CILIENDO E KUNIMASA, 2007).

3.4.1 Complete Fair Queuing (CFQ) ou Filas Completamente Justas

O CFQ é o escalonador padrão utilizado no *Red Hat Linux*, pois implementa QoS (Qualidade de Serviço), mantendo uma fila de operações de entrada/saída para os processos (CILIENDO E KUNIMASA, 2007). Esse é o escalonador utilizado em grandes ambientes, possuindo como característica principal atender a uma grande quantidade de requisições, evitando ao máximo a latência de acesso a disco. Isso ocorre porque o sistema atende e divide o acesso entre os processos, utilizando uma técnica semelhante a que o CFS faz com o processador.

3.4.2 Deadline

O *deadline* é o escalonador que mais se aproxima da velocidade real do subsistema de disco (CILIENDO E KUNIMASA, 2007). Ele utiliza um método cíclico (*round robin*), provendo um bom pedido de latência e mantendo o bom rendimento de acesso a disco. Este é o escalonador recomendado para ambientes que utilizam grande carga de acesso a disco como um banco de dados.

3.4.3 Anticipatory ou Antecipatória

O *Anticipatory* utiliza o mesmo princípio apresentado no *deadline* com o acréscimo do mecanismo de “antecipação”, significando que o escalonador tenta se antecipar escrevendo no disco em um único fluxo de dados ao invés de vários acessos pequenos (CILIENDO E KUNIMASA, 2007). Esse escolanador é recomendado para utilização em computadores pessoais.

3.4.4 Noop

O *Noop* não utiliza um algoritmo complexo no escalonador; ele simplesmente emprega o FIFO (primeiro que entra é o primeiro que sai), não utilizando ordenação dos dados e priorização de acesso como ocorre no CFQ. Esse escalonador é recomendado para o uso em disco de estado sólido que possui grande capacidade de gravação.

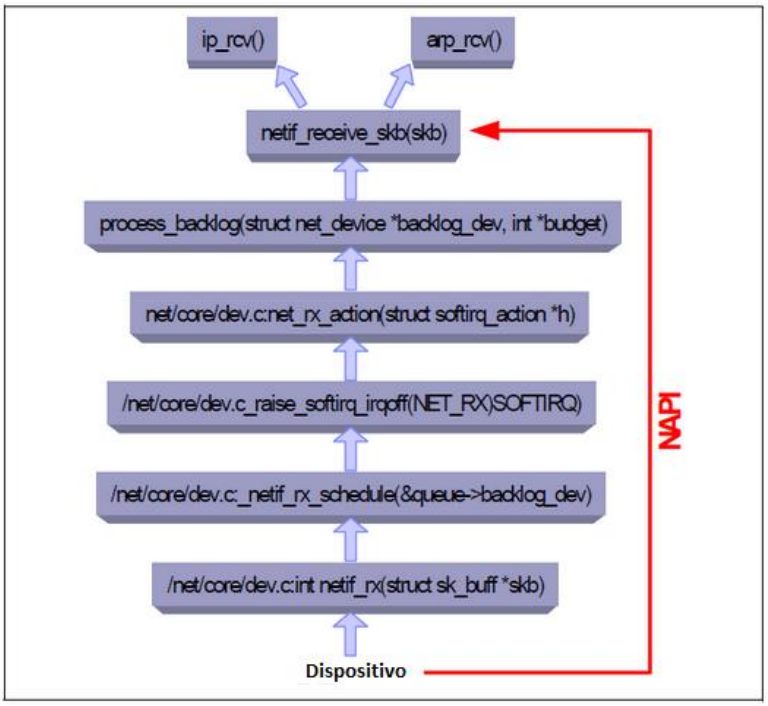
3.5 SUBSISTEMA DE REDE

O subsistema de rede sofreu uma grande mudança com a utilização de uma nova API para serviços de rede: o NAPI (CILIENDO E KUNIMASA, 2007). A configuração padrão do subsistema de rede é focada na confiabilidade e na baixa latência e alto desempenho, sendo ideal para utilização como *firewall,* que é serviço que gerencia a entrada e saída de pacotes em uma rede (CILIENDO, 2004). Com essas características, algumas aplicações como banco de dados acabam tendo um desempenho inferior ao esperado nesse cenário.

De acordo com Ciliendo e Kunimasa (2007), o subsistema de rede é dos componentes mais importantes com relação ao desempenho do sistema operacional, sendo que esse subsistema recebe influência de equipamentos que estão fora do controle do sistema operacional, tais como roteadores e *switches*.

A abordagem padrão acontece quando um pacote chega até a placa de rede e ele é movido para o *buffer* da placa de rede. Na sequência, o pacote é movido do para a *buffer* do sistema operacional e emite uma chamada de interrupção para o processador (CILIENDO, 2004). Por fim, o processamento é realizado e o pacote é enviado para a pilha de rede e entregue à aplicação. (Figura 4).

Figura 4 – Subsistema de rede e NAPI



Fonte: Traduzido de (CILIENDO E KUNIMASA, 2007, p.42).

Com a utilização da implementação do NAPI, o processamento de rede foi ajustado para que somente no primeiro pacote sejam realizados todos os processos convencionais, de modo que os demais pacotes da mesma origem sejam encaminhados para o *buffer* do sistema operacional sem a necessidade da carga de processamento tradicional.

3.6 SISTEMAS DE ARQUIVOS

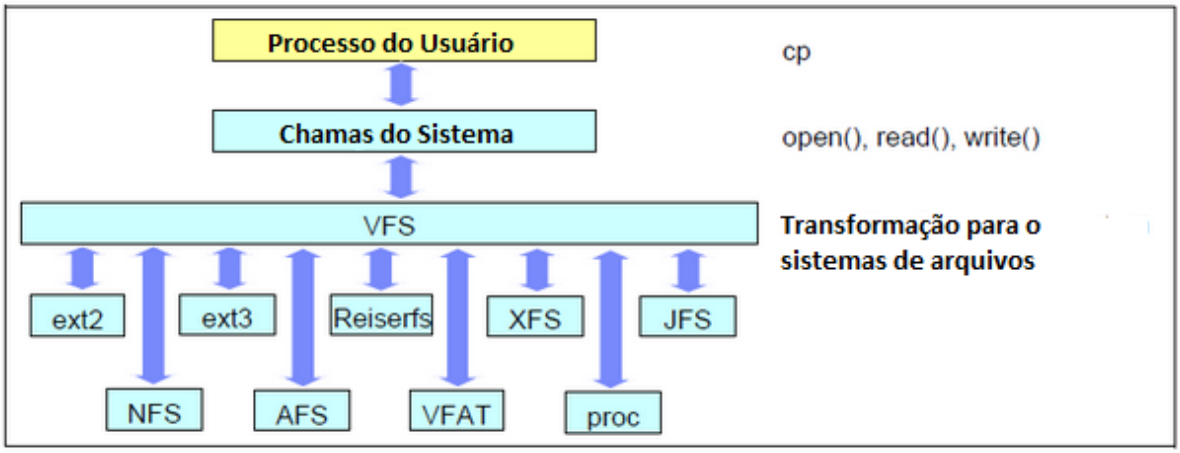
O *Red Hat Linux* suporta uma grande quantidade de sistema de arquivos, sendo cada um deles indicado para determinada carga de trabalho. Nas seções seguintes, aborda-se os principais sistemas de arquivos suportados no *Red Hat Linux*, bem como o funcionamento da camada de abstração entre os processos de usuário e o sistema de arquivos.

3.6.1 Virtual File System (VFS)

O *Virtual File System* ou Sistema de Arquivos Virtual (VFS), segundo Ciliendo e Kunimasa (2007), é a camada de abstração entre a camada de aplicação e os

vários sistemas de arquivos suportados pelo *Linux*. O VFS é responsável por prover modelos de objetos comuns, entre eles, objeto de arquivo, *cache* etc, bem como os métodos necessários para acessar os objetos descritos. Dessa forma, o VFS esconde o método de implementação de cada sistema de arquivos. A Figura 5 ilustra a camada de VFS.

Figura 5 – *Virtual File System*



Fonte: Traduzido de (CILIENDO E KUNIMASA, 2007, p.30).

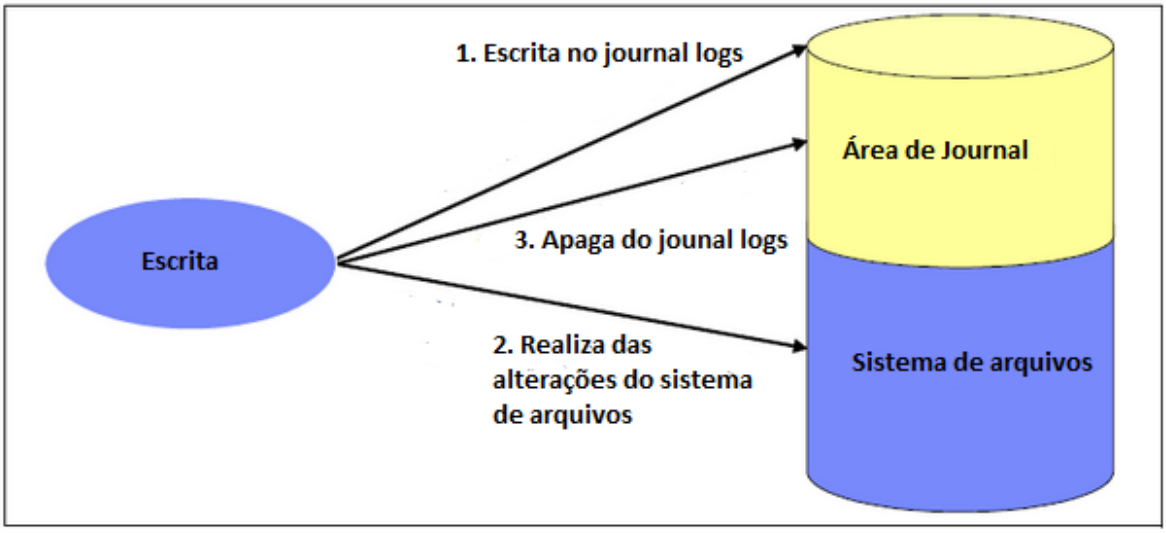
3.6.2 Ext4

Ext4 é o sistema de arquivos que substitui seu antecessor, o ext3, o qual é suportado pela maioria das distribuições Linux do mercado, incluindo o *Red Hat Linux*, sendo ele o sistema de arquivos padrão (DJORDJEVIC E TIMCENKO, 2012). O ext4 suporta o tamanho máximo de sistemas de arquivos em 16 TB, de modo que o tamanho máximo de um arquivo é de 16 TB (Red Hat Enterprise Linux 6 Performance Tuning Guide, 2013). O ext4 mantém as compatibilidades com o ext3 e ext2, sendo uma tarefa simples a migração para o novo sistema de arquivos.

A grande diferença entre o ext4 e o ext3 é o fato do ext4 ser um sistema de arquivos de 64 bits, possuindo uma maior escalabilidade e eficiência (DJORDJEVIC E TIMCENKO, 2012). Ele possui suporte *journaling*, que consiste em realizar as gravações em uma área que se chama *journal* antes dos arquivos serem gravados no sistema de arquivos (CILIENDO E KUNIMASA, 2007). O *journaling* é utilizado para manter a integridade dos dados caso ocorra alguma falha, pois as alterações

gravadas no *journal* podem ser recuperadas através da leitura dos *logs* de gravação. A Figura 6 ilustra o funcionamento do *journaling.*

Figura 6 – Funcionamento do *Journal*



Fonte: Traduzido de (CILIENDO E KUNIMASA, 2007, p.30).

O ext4 é recomendado para a utilização nos mais variados ambientes e aplicações. A Oracle recomenda a utilização de Ext4 em ambientes de produção.

3.6.3 XFS

Pelas informações do Red Hat Enterprise Linux 6 Performance Tuning Guide (2013), o XFS é um robusto sistema de arquivos em arquitetura 64 bits com suporte a *journaling*. Ele suporta sistemas de arquivos com tamanho de até 500 TB. Assim, é um sistema de arquivos recomendado para ambientes com grande volume de dados.

O XFS apresenta excelente escalabilidade de entrada/saída, utilizando árvores binárias para indexação dos arquivos do usuário e metadados do sistema de arquivos.

3.6.4 Proc

O *proc* não é um sistema de arquivos real, e sim diretório com um conjunto de subdiretórios que representa o estado atual do sistema operacional (Red Hat

Enterprise Linux 6 Performance Tuning Guide, 2013). Os subdiretórios do *proc* possuem as informações de *hardware* e as informações dos processos que estão sendo executados no sistema operacional. As ferramentas de monitoramento abordadas nas próximas seções utilizam esse diretório para coletar as informações.

3.7 FERRAMENTAS DE MONITORAMENTO

O Linux possui uma grande variedade de ferramentas de monitoramento do estado do sistema. Nas próximas seções serão abordadas algumas ferramentas de linha de comando para monitoramento.

3.7.1 Comando uptime

Na posição de Ciliendo e Kunimasa (2007), o comando *uptime* mostra a quantidade de tempo que o servidor está no ar, bem como a informação de quantos usuários estão atualmente logados no mesmo. Outra informação importante consiste no *system load average* ou carga média do sistema. Na consideração de Ciliendo e Kunimasa (2007), a carga média do sistema é divido em três métricas: a primeira média no último minuto; a segunda é a média nos últimos cinco minutos; e a terceira é a média dos últimos trinta minutos.

Quando os valores forem baixos na carga média do sistema, isso representa que o sistema está com pouca carga de trabalho. Para avaliar se o sistema está com a carga de utilização elevada, é necessário avaliar quantos processadores estão ativos no sistema. Por exemplo, se o servidor possuir quatro processadores e a média do sistema nas três métricas está em cinco, quer dizer que o servidor está com carga mais alta que sua capacidade.

As informações sobre o comando *uptime* estão contidas neste trabalho na página de documentação através do comando *man uptime*.

3.7.2 Comando top

Conforme o *Red Hat Enterprise Linux 6 Performance Tuning Guide* (2013), o comando *top* é uma ferramenta dinâmica de tempo real, que mostra os processos que estão em execução no sistema e, além disso, informações detalhadas de cada

processo como, por exemplo, a quantidade de memória compartilhada utilizada, memória residente utilizada etc. O comando *top* também mostra em tempo real a carga média do sistema, informações de memória do sistema desde a sua inicialização. Outra informação importante que a ferramenta mostra é a utilização do tempo de *wait for I/O*, ou seja, a porcentagem de tempo que o processador aguarda no subsistema disco para a realização de operações de entrada e saída.

As informações sobre o comando *top* estão na página de documentação através do comando *man top* do presente trabalho.

3.7.3 Comando ps

Ao contrário do comando *top*, o *ps* mostra somente o estado atual dos processos, de maneira que suas informações mais detalhadas podem ser encontradas em Red Hat Enterprise Linux 6 Performance Tuning Guide (2013).

Para uma melhor compreensão sobre o assunto, o presente estudo tem informações sobre o comando *os* na página de documentação através do comando *man ps*.

3.7.4 Comando vmstat

O *vmstat (Virtual Memory Statistics* ou Estatísticas de Memória Virtual) prove um relatório sobre os processos do sistema, utilização de memória, paginação, entrada/saída e interrupções (Red Hat Enterprise Linux 6 Performance Tuning Guide, 2013). Essa ferramenta tem a mesma característica do *ps*, porém não em tempo real.

Sobre o comando *vmstat,* a página de documentação através do comando *man vmstat*, deste trabalho, mostra mais informações.

3.7.5 Comando sar

No ponto de vista de Ciliendo e Kunimasa (2007), o comando *sar* é utilizado para coletar, reportar e armazenar estatísticas do sistema. O *sar* exibe informações semelhantes ao *top*, mas com a vantagem de gerar relatórios de estatística. Além

das informações sobre processo, memória e disco, o *sar* também coleta informações sobre a utilização de rede.

Ciliendo e Kunimasa (2007) explicam que é recomendado utilizar a ferramenta *sar* para monitoramento de todos os sistemas *Linux* instalados.

O comando *sar* está demonstrado na página de documentação através do comando *man sar*, neste trabalho.

3.7.6 Comando free

O comando *free* é utilizado para coletar informações referentes a utilização da memória do sistema, tais como memória livre e quantidade de *swap* que o sistema está utilizando. O comando é semelhante ao *ps*, que coleta informação do momento em que o comando foi executado.

O comando *free* apresenta uma melhor análise na página de documentação através do comando *man free* deste trabalho.

3.7.7 Comando iostat

O comando *iostat* apresenta informações similares ao *uptime,* mas com a adição de informações detalhadas do subsistema de disco. Conforme Ciliendo e Kunimasa (2007), o *iostat* cria relatórios detalhados, sendo a ferramenta ideal para identificação de gargalos no subsistema de disco.

3.7.8 Comando dmesg

O comando *dmesg* mostra as mensagens do *kernel* do Linux, sendo assim útil para verificar problemas de *hardware* no servidor (CILIENDO E KUNIMASA, 2007). Através desse comando podem ser encontrados problemas de alocação de memória, subsistema de disco e até mesmo módulos de dispositivos com problema.

3.7.9 Tabela Comparativa

Abaixo, tem-se uma tabela comparativa entre as ferramentas apresentadas, mostrando as diferenças entre elas e para cada situação que a mesma pode ser utilizada.

Tabela 1 – Comparação entre as ferramentas

Comando	Carga de Trabalho	Processador	Disco	Memória	Tempo Real	Relatórios
uptime	X					
Ps		X				
Top	X	X	X	X	X	
vmstat				X		
Sar	X	X	X	X	X	X
Free						
iostat	X		X		X	X

4 BANCO DE DADOS ORACLE

A primeira parte deste capítulo consiste no estudo sobre a arquitetura e funcionamento do banco de dados *Oracle*. Já na segunda etapa, tem-se a elaboração do diagnóstico utilizando a ferramenta *StatsPack* que faz parte do banco de dados *Oracle*. O banco de dados *Oracle* suporta transações e respeita as propriedades ACID (atomicidade, consistência, independência e durabilidade) em um banco de dados, ou seja, quando uma transação é executada, ela é realizada na íntegra ou é totalmente desfeita em caso de problemas na execução (BRAGHETTO, 2006).

No pensamento de Elmasri e Navathe (2011), transação é uma unidade atômica de trabalho que atua sobre um banco de dados, sendo assim todas as transações devem ser bem-sucedidas, caso contrário, toda ela deve ser desfeita. Conforme Elmasri e Navathe (2011), uma transação é bem-sucedida quando ela termina com o comando *COMMIT*, fazendo com que as operações sejam gravadas no banco de dados. Quando uma transação não é bem-sucedida, ou seja, ocorre uma falha antes do processo de *COMMIT,* é gerado o processo de *ROLLBACK,* que, segundo Elmasri e Navathe (2011), consiste em desfazer todas as alterações realizadas pela transação.

Para Watson (2010), o banco de dados *Oracle* é divido em duas partes, sendo elas:

- Instância; e
- Banco de dados

A instância consiste na estrutura de memória e processos, já a estrutura de banco de dados são os arquivos armazenados no disco (WATSON, 2010). No processo de *tuning* do banco de dados é fundamental o conhecimento de como a instância interage com o banco de dados. Nas próximas seções, aprofunda-se o funcionamento da instância e do banco de dados.

4.1 ARQUITETURA DA INSTÂNCIA

Watson (2010) comenta que a instância é composta de estruturas de memória e processos, sendo sua existência temporária na memória principal. Os processos

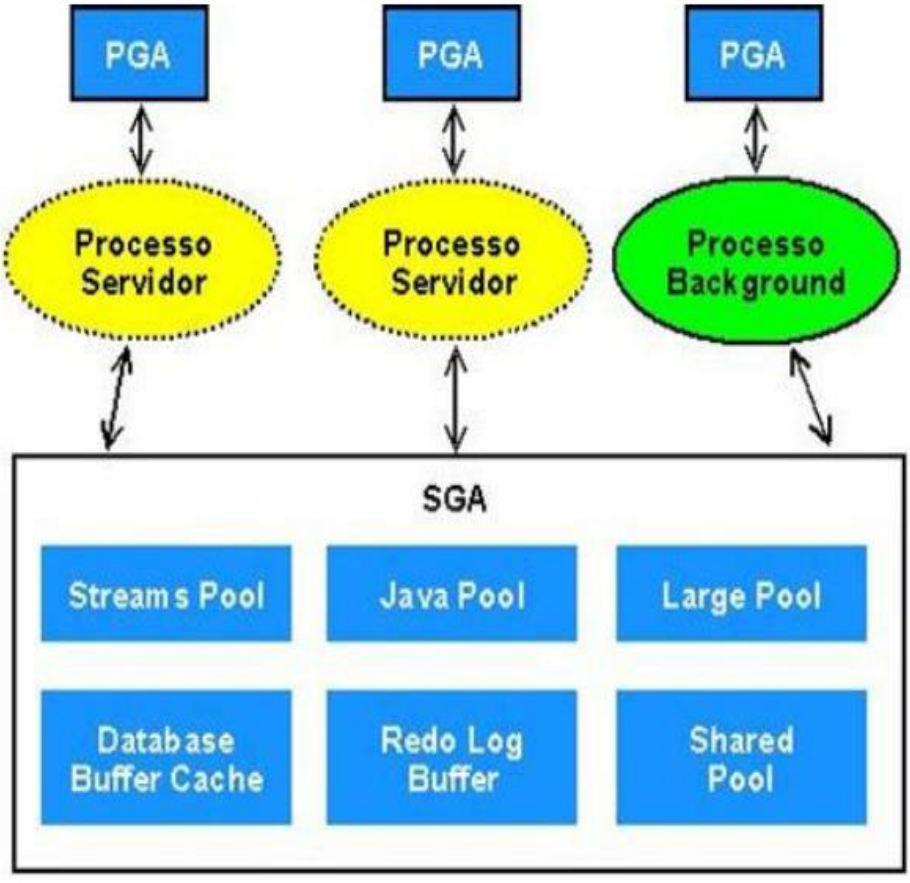
que fazem parte da instância são conhecidos como processos *background* (ou processos de segundo plano), sendo eles iniciados quando a instância está ativa. Nesta seção, são abordados os principais componentes da estrutura de memória e processos da instância.

O gerenciamento de memória da instância é divido em duas áreas:

- Área de Sistema Global; e
- Área Global de Programa.

Cada uma das áreas está analisada com mais detalhes nas próximas seções. A Figura 7 ilustra a estrutura de memória do banco de dados *Oracle*.

Figura 7 – Estrutura de memória do banco de dados Oracle



Fonte: Retirado de (MACEDO *et al*. ,2013, p.114).

4.1.1 Área de Sistema Global

Área de Sistema Global ou SGA (*System Global Area*) são segmentos de memória fornecidos pelo sistema na inicialização da instância (WATSON, 2010). A

estrutura da SGA contém alguns componentes e processos de segundo plano. Os principais componentes da SGA são: *Cache Buffer*, *Buffer Log*, *Shared Pool*, *Java Pool* e PGA, os quais são detalhados nas próximas seções.

4.1.1.1 Cache Buffer

Na opinião de Watson (2010), o *cache buffer* é a área de trabalho do banco de dados *Oracle* para a execução de Linguagem de Consulta Estruturada (SQL). Quando é realizada uma atualização de dados, ela é feita no *cache buffer* em primeiro momento. Ou seja, ela não é realizada na área de banco de dados, que é armazenada no disco.

Os blocos que contém dados são copiados para o *cache buffer* e as alterações são aplicadas a essas cópias dos blocos de dados no *cache buffer* (WATSON, 2010). Os dados são mantidos no *buffer* até que o espaço que está sendo ocupado por eles no *cache buffer* seja solicitado por outros processos. As consultas executadas no banco também se beneficiam do *cache buffer*, pois os blocos recém-utilizados ficam disponíveis no mesmo.

Na visão de Watson (2010), todos os blocos que contêm dados acessados com frequência estarão no *cache buffer*, sendo assim as operações mais frequentes são realizados em memória e não no disco, no qual as operações de entrada/saída são mais lentas.

De acordo com Watson (2010), o tamanho do *cache buffer* do banco de dados é crucial para o desempenho, pois caso os blocos que o banco de dados utiliza com frequência não estão presentes no *cache buffer*, é necessário que seja realizada a leitura de disco, que é mais lenta que o acesso à memória principal.

4.1.1.2 *Buffer* Log

Watson (2010) orienta que o *buffer log* é uma área de preparação pequena e de curto prazo para os vetores de alterações antes que eles sejam gravados no *redo log* no disco. Os vetores de alteração são as modificações de transações validadas realizadas nos blocos de dados. Já o *redo log*, segundo Watson (2010), é a garantia do banco de dados que os dados nunca serão perdidos. O banco de dados grava os

vetores de alterações no *redo log,* assim sendo possível extrair as alterações realizadas no banco de dados até o ponto de falha.

Seguindo com a mesma técnica do *cache buffer,* nada é gravado diretamente no disco, ou seja, os vetores de alteração são gravados na memória principal que é mais rápida que no disco. Uma gravação do *buffer log* no disco pode ser um lote de muitos vetores de alteração provenientes de muitas transações (WATSON, 2010). Para garantir a integridade do banco de dados, o *redo log* é gravado no disco praticamente em tempo real, e o processo de segundo plano responsável pela gravação do *redo log* é o LGWR, o qual está apresentado com detalhes na seção *Log Writer* (LGWR).

Consoante Watson (2010), o *buffer log* é circular, isto é, à medida que as sessões de usuário gravam os vetores de alteração, o processo LGWR grava as informações do *redo log*, liberando espaço no *buffer log*. Caso o processo LGWR seja mais lento na gravação do *redo log* do que a criação dos vetores de alteração, todas as atividades de modificação de dados serão suspensas enquanto o *buffer log* é limpo pelo LGWR. Dessa forma, esse processo é um dos maiores gargalos do banco de dados Oracle, pois não é possível realizar alterações nos dados mais rápido que o LGWR, sendo que o mesmo consegue gravar as informações do *redo log* (WATSON, 2010).

4.1.1.3 *Shared Pool*

Na consideração de Watson (2010), o *shared pool* é a mais complexa estrutura da SGA. Ela é divida em várias estruturas que são gerenciadas internamente pelo banco de dados Oracle.

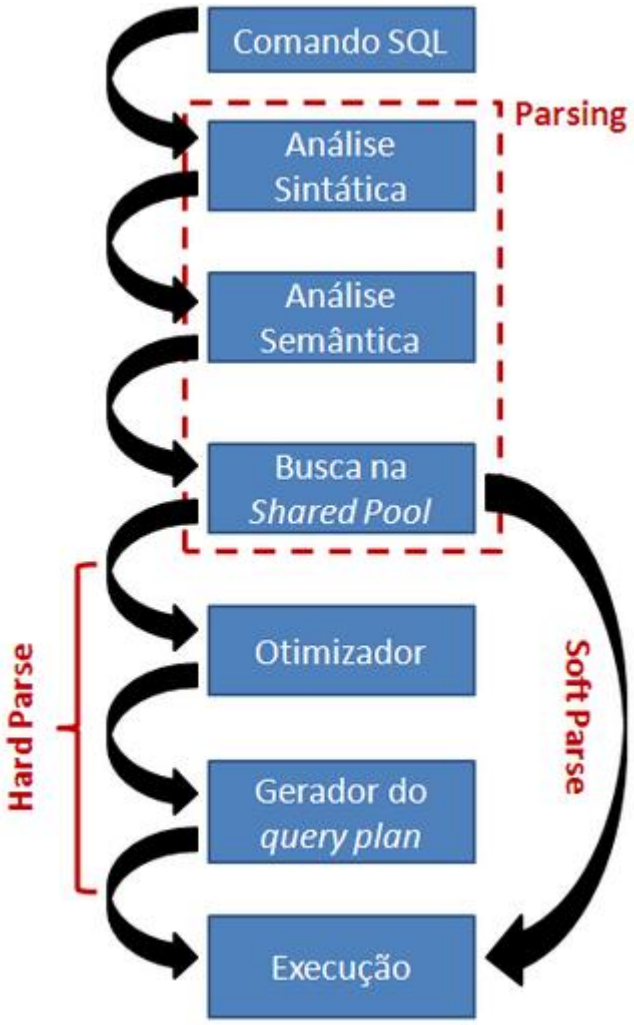
As principais estruturas do *shared pool* estão descritas nas próximas seções.

4.1.1.3.1 Libary Cache

O *libary cache* ou *cache* de biblioteca é uma área de memória usada para armazenar o código executado recentemente, na sua forma analisada por *parse* (WATSON, 2010). A análise *parse* é o processo que converte do código SQL escrito pelo programador na linguagem interna no Oracle, sendo também conhecido como *hard parse*. O armazenamento do código *parse* é realizado para que o Oracle não

necessite realizar o processo novamente para mesma instrução, ou seja, esse processo também conhecido como *soft parse*. O processo de *parse* do banco de dados é muito custoso, sendo em alguns casos mais lento que a própria execução da instrução. A Figura 8 apresenta o processo de *parse*, composto pelas etapas de análise sintática, análise semântica e busca no *shared pool*.

Figura 8 – Estrutura de parse do banco de dados Oracle



Fonte: Retirado de (SILVA, P. H).

O *cache* de biblioteca faz justamente com que as instruções já executadas não passem novamente pelo processo de *parse* e sejam executadas diretamente aumentando assim o desempenho da instrução.

4.1.1.3.2 Cache de dicionário de dados

Segundo Watson (2010), o *cache* de dicionário de dados armazena as definições de objetos usados recentemente, tais como descrições de tabela, índices, usuários etc. A ideia do *cache* de dicionário de dados é fazer com que os dados mais acessados estejam na memória para que a operação de *parse* não precise consultar o dicionário de dados no disco para realização da análise.

4.1.1.3.3 Área PL/SQL

Os objetos como *procedures*, *triggers*, funções ou pacotes estão armazenados no dicionário de dados como código-fonte e na sua forma compilada (Watson, 2010). A área de PL/SQL do *shared pool* é responsável por armazenar esses objetos na memória principal para que eles não precisem ser lidos no dicionário de dados após a primeira execução.

4.1.1.3.4 Cache de resultados de funções PL/SQL e consultas SQL

Conforme Watson (2010), o *cache* de resultado permite que o banco de dados armazene os resultados no *cache*, fazendo com que na próxima vez que uma consulta seja executada o resultado seja obtido diretamente do *cache*. Caso uma tabela sofra alguma alteração, o otimizador do banco de dados *Oracle* irá executar a instrução por completo, atualizando assim o *cache* de resultado, não correndo o risco de obter informações desatualizadas.

4.1.1.4 Java Pool

É utilizado para execução de *procedures Java* armazenadas no banco de dados, sendo empregado como um espaço *heap* que é necessário para instanciar os objetos *java* (WATSON, 2010).

4.2 PGA

Program Global Area ou PGA é associada à memória não compartilhada, sendo a área privada da sessão do usuário (WATSON, 2010). As informações alteradas na sessão que não sofreram o processo de *COMMIT* ficam registradas

somente na sessão e, somente após o *COMMIT*, as informações ficam disponíveis para as demais sessões.

4.3 PROCESSOS DA INSTÂNCIA

Os processos de segundo plano são iniciados juntamente com a instância, e permanecem ativos até que a instância seja encerrada. Nesta seção, são abordados os principais processos de segundo plano (Watson, 2010), sendo eles:

- *System Monitor* (SMON)
- *Process Monitor* (PMON)
- *Database Writer* (DBWn)
- *Log Writer* (LGWR)
- *Manageability Monitor* (MMON)
- *Memory Manager* (MMAN)

4.3.1 System Monitor (SMON)

No entendimento de Watson (2010), o processo SMON é responsável por montar o banco de dados, localizando e validando os arquivos de controle. Com o término dessa tarefa, o processo é responsável por localizar e validar os arquivos de dados e os arquivos de *log on-line*, assim realizando o processo de abertura do banco de dados.

Após a abertura do banco de dados, o SMON é responsável por realizar o monitoramento de espaço livre nos arquivos de dados e consistência dos arquivos de *redo log*.

4.3.2 Process Monitor (PMON)

O processo PMON é responsável por monitorar os processos de sessão de usuário e detectar problemas com os mesmos (WATSON, 2010). Caso uma sessão seja encerrada da forma incorreta, o processo PMON é encarregado de encerrar o processo de sessão do usuário com problema, efetuando a liberação da memória utilizada e executando um *rollback* de todas as transações incompletas.

4.3.3 Database Writer (DBWn)

As sessões de usuário realizam grande parte de suas transações no *buffer cache,* na memória principal, não gravando as informações no disco. O processo responsável por realizar a gravação dos blocos que estão no *buffer cache* no disco é o *database writer* ou gravador de banco de dados. Cada instância pode possuir vários *database writers*, sendo esses denominados DBW0, DBW etc (WATSON, 2010).

O processo DBWn grava a menor quantidade de dados possível no disco. O processo realiza a gravação dos blocos no disco somente nas seguintes situações:

- Ausência de *buffer* livre;
- Excesso de *buffer* sujo;
- Tempo limite de três segundos; e
- Quando há um *checkpoint* (mecanismo que salva o estado atual do banco de dados mesmo sem a execução do *COMMIT*).

(WATSON, 2010)

4.3.4 Log Writer (LGWR)

Watson (2010) explica que o LGWR é responsável por gravar o conteúdo do *buffer log* nos arquivos de *redo log on-line*. Essa operação é realizada para garantir a integridade do banco de dados caso ocorra alguma falha.

O LGWR realiza a gravação nos arquivos de *redo log* nas seguintes circunstâncias:

- Quando uma sessão emite uma instrução *COMMIT*;
- Quando o *buffer* log estiver um terço completo; e
- Se DBWn estiver pronto para gravar blocos sujos.

(WATSON, 2010)

4.3.5 Manageability Monitor (MMON)

Conforme descrito por Watson (2010), o MMON é responsável por reunir as estatísticas sobre a atividade de desempenho. Todas as estatísticas coletadas no

processo MMON são armazenados na SGA para realização de análise de desempenho e emissão de histórico. Além da coleta de estatísticas de desempenho, o MMON também é responsável por monitorar a instância para a emissão de alertas definidos.

4.4 ARQUITETURA DO BANCO DE DADOS

A estrutura física do banco de dados Oracle é composta por três arquivos, sendo eles:

- Arquivo de Controle;
- Arquivos de *redo log*; e
- Arquivos de Dados.

  Cada um desses arquivos é abordado com mais detalhes nas seções a seguir.

4.4.1 Arquivo de Controle

O arquivo de controle é o principal arquivo da estrutura física do banco de dados Oracle. Nele estão contidas as referências para os demais arquivos físicos com o *redo log* e arquivos de dados. Além dos ponteiros, os arquivos de controle armazenam as informações para manter a integridade dos dados (WATSON, 2010).

4.4.2 Arquivos redo log

Watson (2010) especifica que o *redo log* armazena uma cadeia contínua em ordem cronológica de cada vetor de alteração válida aplicada no banco de dados. Através desse arquivo é possível “rebobinar” o banco de dados até o momento desejado.

O banco de dados Oracle é composto de dois conjuntos de arquivos de *redo log*, sendo que um dos conjuntos é utilizado para manter as alterações correntes e no outro conjunto é realizada uma cópia de segurança.

4.4.3 Arquivos de Dados

Na definição de Watson (2010), os arquivos de dados são repositórios de dados, nos quais estão armazenados os segmentos lógicos tais como tabelas, índices, dicionário de dados etc. Nesses arquivos de dados são gravados os blocos sujos através do processo DBWn.

5 MONITORAMENTO DO BANCO DE DADOS ORACLE

O banco de dados Oracle possui uma variada coleção de ferramenta para monitoramento do banco de dados, além de suportar ferramentas de terceiros para o mesmo propósito. As ferramentas que são disponibilizadas junto com o banco de dados Oracle são: o AWR (*Automatic Workload Repository*) e o *StatsPack*.

Como o objetivo do trabalho proposto é a utilização do banco de dados Oracle em sua versão *Standard*, não é possível utilizar o AWR, pois o mesmo necessita da utilização da versão *Enterprise* com o pacote *Tuning Pack* do banco de dados Oracle licenciado. Muitas ferramentas de terceiros também utilizam do pacote *Tuning Pack* para coletar as informações necessárias.

Para não ocorrer problemas de licenciamento, a ferramenta utilizada para monitoramento do banco de dados será o Statspack, que não necessita de licenças adicionais.

Segundo o *Oracle9i Database Performance Tuning Guide and Reference Release 2* (2002), o *Statspack* é um pacote de *scripts* SQL que coleta, automatiza, armazena e permite a visualização das informações de desempenho do banco de dados, de modo que informações coletadas são armazenadas para análise futura. Conforme o *Oracle9i Database Performance Tuning Guide and Reference Release 2* (2002), os dados coletados podem ser analisados utilizando as ferramentas de relatórios que o *StatsPack* provê, sendo que as principais informações são:

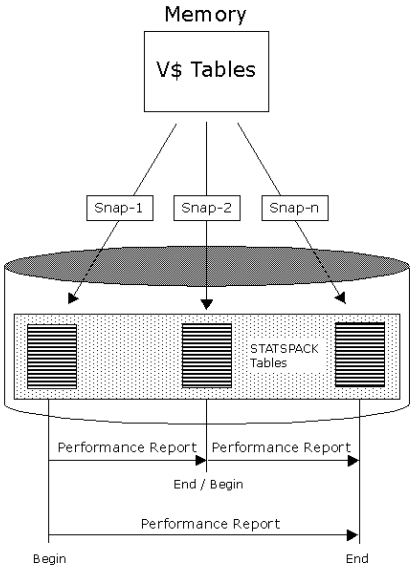
- Saúde da instância.
- Sumário de carga.
- Informações de *cache.*
- Eficiência da Instância.
- Eventos em espera.
- Estatísticas de Processador e Memória.
- Estatísticas de sistema de arquivos.

Para tanto, cada uma dessas informações está abordada nas próximas seções.

5.1 FUNCIONAMENTO *STATSPACK*

A ferramenta *StatsPack* coleta as informações da memória do banco de dados e as armazena nas tabelas criadas no processo de instalação do *StatsPack*. A coleta de informações, segundo o *Oracle9i Database Performance Tuning Guide and Reference Release 2* (2002), isso se dá através de um *snapshot (*fotografia), que consiste em tirar uma "fotografia" do banco de dados e realizar uma comparação com as *snapshots* antigas presentes nas tabelas do *StatsPack*. A Figura 9 ilustra o processo de geração de relatório de comparação entre as *snapshots* geradas.

Figura 9 – Estrutura de funcionamento do StatsPack



Fonte: http://www.akadia.com/services/ora_statspack_survival_guide.html

Segundo o *Oracle9i Database Performance Tuning Guide and Reference Release 2* (2002), é possível alterar a quantidade de informações que a *snapshot* coleta, sendo as informações divididas nos seguintes níveis:

- Nível 0 : nível que coleta informações gerais de estatísticas do banco de dados, tais como: estatística de desempenho, estatísticas do sistema, informações da SGA etc;
- Nível 5: coleta informações de quais comandos SQL que mais utilizam recursos, além das informações coletadas dos níveis anteriores;
- Nível 6: captura o plano de execução dos comandos SQL, e é utilizado para encontrar problemas nas consultas SQL, além das informações coletadas pelos níveis anteriores;

- Nível 7: captura informações no nível de segmento de dados, tais como: leitura de blocos, espera de *buffer* e todas as informações dos níveis anteriores. Esse é nível recomendado na grande maioria dos casos;
- Nível 10: captura informações de Latch (bloqueio na memória do banco de dados) e todas as informações dos níveis anteriores.

Quanto mais alto o nível utilizado na captura da fotografia do banco de dados mais recursos são despendidos para a realização das operações de captura.

5.2 RELATÓRIOS STATSPACK

A ferramenta *StatsPack* é capaz de gerar relatórios com uma grande quantidade de informações e detalhes. Saber realizar a interpretação do relatório é fundamental para avaliação do correto funcionamento do banco de dados e para a identificação de problemas nos parâmetros de configuração do banco de dados. Nas próximas subseções são apresentadas as informações mais relevantes do relatório.

5.2.1 Seção Sumário

O sumário é a seção que ilustra informações gerais, tais como:

- Quantidade de processadores;
- Versão do banco de dados Oracle;
- Se o banco de dados faz parte de um ambiente RAC (*Real Application Cluster*); e
- Tempo de atividade do banco de dados.

As informações apresentadas são apenas com fins informativos para situar o DBA no ambiente em que foi executado o relatório.

5.2.2 Seção Informação de Cache e Load Profile (Perfil de Trabalho)

Nesta seção do relatório, tem-se o tamanho do *Cache Buffer*, *Log Buffer e Shared Pool Size*. Estes são os principais componentes da SGA do banco de dados Oracle. Na seção *Load Profile* seção do relatório, mostra uma grande quantidade de

informações referente ao estado do banco de dados Oracle. As principais informações que essa seção aborda são:

- *Hard Parses*: ilustra a quantidade de *hard parses* que o banco de dados realizou desde o início até o fim do *snapshot*. Quanto maior o valor desse campo do relatório, mais recursos do banco de dados são despendidos para realização da operação de *parse*, e menos consultas estarão presentes no *cache* de biblioteca;
- *Transactions*: ilustra a quantidade de transações executadas por segundo. Essa informação é útil para a criação de uma linha base para o processo de *tuning* visto no Capítulo 2;
- *Executes*: ilustra a quantidade de execuções que são realizadas por segundo em cada transação. Essa informação é útil para a criação de uma linha base para o processo de *tuning* visto no Capítulo 2.
- *Physical Reads*: ilustra a quantidade de leituras executadas no disco a procura de blocos, realizando operações de entrada/saída. Quando esse valor está muito alto pode ser uma indicação que o *cache buffer* não está com o tamanho adequado;
- *Physical Writes*: ilustra a quantidade de gravações realizadas no disco.

5.2.3 Eficiência da Instância

Esta seção do relatório é a mais importante para identificação de problemas de configuração e dimensionamento da estrutura de memória do banco de dados Oracle. As principais informações trazidas pela seção do relatório são:

- *Buffer Hit*: ilustra a taxa de acertos na procura de blocos no *cache buffer*. Esse indicador significa que, por exemplo, 95% os blocos solicitados estão no *cache buffer* no período entre o início e o fim do *snapshot*;
- *Library Hit*: ilustra a taxa de acertos na procura por instruções SQL já executadas pelo banco de dados. Esse indicador mostra que, por exemplo, 97% das instruções estão contidas no *cache* de biblioteca, não ocorrendo assim o processo de *hard parse*;
- *Parse CPU to Parse Elapsed*: ilustra a quantidade de tempo do processador teve para a realização do processo do *parse*;

- *Soft Parse*: ilustra a taxa de vezes que as sessões do banco de dados utilizam as instruções que já estão presentes no *shared pool*.

5.2.4 Top 5 Eventos Temporários

Esta seção do relatório é o foco do processo de *tuning*, pois é ela que irá mostrar os eventos do banco de dados que mais estão consumindo tempo. A grande maioria dos eventos listados são eventos de espera, ou seja, eventos em que o banco de dados está esperando por algum recurso para a conclusão de uma operação. Alguns eventos comuns que podem ser listados nessa seção:

- *CPU Time*: esse evento não é necessariamente um evento de espera, ilustrando a quantidade de processador que é utilizado no intervalo entre início e fim do *snapshot* do banco de dados;
- *Db file sequential read*: ilustra a quantidade de espera para que sejam realizadas gravações na área temporária do banco de dados. Esse evento com valores altos indica problema de dimensionamento na estrutura de PGA;
- *Db file scattered read*: ilustra a quantidade de espera para leitura de blocos que estão presentes no *cache buffer*, mas que necessitam também de blocos presentes no disco. Esse evento com valores altos indicam problema de dimensionamento da cache *buffer*;
- *Library Cache Load Lock*: ilustra a quantidade de tempo que a sessão está aguardando para carregar os objetos no *cache* de biblioteca. Esse evento com valores altos indicam problema de dimensionamento na área de *shared pool* do banco de dados;
- *Log File Parallel Write*: ilustra a quantidade de tempo que o processo está esperando para realizar a gravação no *redo log*. Esse evento com valores elevados pode indicar gargalo no subsistema de disco;
- *Control File Parallel Write*: ilustra a quantidade de tempo que o processo está esperando para realizar a gravação em todos os arquivos de controle do banco de dados. Esse evento com valores elevados pode indicar gargalo no subsistema de disco;

- *DB File Single Write*: ilustra a quantidade de tempo que o processo está esperando para gravar os cabeçalhos nos arquivos de dados. Esse evento com valores elevados pode indicar gargalo no subsistema de disco;
- *Buffer Busy Waits*: ilustra a quantidade de tempo que o processo está esperando para realizar o acesso a um determinado bloco que se está bloqueado no *cache buffer*. Esse evento com valores elevados pode indicar problema no dimensionamento da SGA.

5.2.5 Estatística do Processador e Memória e Seção File IO Stats

A seção Estatística do processador e memória do relatório mostra as informações referentes à utilização e consumo do processador pela instância, bem como a utilização da estrutura de memória SGA e PGA pelo banco de dados Oracle.

A seção *File IO Stats* do relatório mostra as informações referentes às estatísticas de operações de entrada/saída nos arquivos de dados do banco de dados. Essas informações são utilizadas para determinar se o subsistema de disco está causando algum gargalo nas operações do banco de dados.

6 TUNING DO SISTEMA OPERACIONAL RED HAT

Como dito no Capítulo 3, o Linux possui uma grande variedade de parâmetros que podem ser ajustados a fim de atingir o desempenho esperado. Para Ciliendo e Kunimasa (2007), o processo de *tuning* deve ser direcionado para a aplicação que se deseja melhorar o desempenho. Como o objetivo do trabalho é melhorar o desempenho do banco de dados Oracle, os parâmetros e configurações abordadas neste capítulo serão para esse objetivo. Sendo assim, esses parâmetros podem não ser aplicados ou não ter o efeito esperado em outras aplicações.

Este capítulo será dividido nos processos de *tuning* para cada subsistema do sistema operacional Red Hat Linux, verificando-se as configurações e parâmetros que podem ser ajustados, a fim de atingir o desempenho desejado.

6.1 TUNING SUBSISTEMA DE MEMÓRIA

Na interpretação de Ciliendo e Kunimasa (2007), o subsistema de memória é um dos fatores mais importantes para o desempenho do banco de dados, pois conforme a arquitetura descrita do banco de dados Oracle, a grande maioria das operações realizadas no banco de dados são em memória. O bom dimensionamento dos recursos de memória, conforme Ciliendo e Kunimasa (2007), consiste em garantir memória suficiente para o banco de dados. Além disso, é necessário reservar certa quantidade de memória para as operações do próprio sistema operacional.

6.1.1 Memória Compartilhada

De acordo com o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o Red Hat Linux provê um amplo suporte à memória compartilhada, mas para o correto funcionamento do banco de dados Oracle alguns ajustes são necessários.

O dimensionamento correto da memória compartilhada está diretamente ligado à quantidade de memória física disponível e o quanto de memória será configurado na SGA do banco de dados Oracle. Conforme o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), para garantir o processo de *tuning*

correto para a utilização da memória SGA do banco de dados Oracle, é necessário calcular os limites que o sistema operacional irá reservar para a memória compartilhada.

Os parâmetros de configuração de memória compartilhada necessários o correto funcionamento do banco de dados Oracle são:

- kernel.shmall
- kernel.shmmax
- kernel.shmmni

Os parâmetros descritos nessa seção são configurados no arquivo de configuração */etc/sysctl.conf.*

6.1.1.1 Parâmetro kernel.shmall

O parâmetro **kernel.shmall** corresponde à quantidade máxima de memória compartilhada que o sistema operacional irá disponibilizar para as aplicações. Para o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), é necessário aplicar a seguinte equação para definir o **kernel.shmall**:

x*(1024*1024*1024)/4096;

Onde **x** corresponde à quantidade de memória disponível no servidor.

Por exemplo, se o servidor possui 128Gb de memória, o cálculo se dá da seguinte forma: 128*(1024*1024*1024)/4096 = 33554432

O valor obtido como, por exemplo, 33554432, deverá ser adicionado no parâmetro de configuração **kernel.shmall** do sistema operacional para 128Gb de memória.

6.1.1.2 Parâmetro kernel.shmmax

O parâmetro **kernel.shmmax** corresponde ao tamanho máximo do segmento de memória compartilhada. Para isso, na análise do Oracle Database 11g Release 2

on Red Hat Enterprise Linux 6 (2012), é necessário aplicar a seguinte equação para definir o **kernel.shmmax**:

x*(1024*1024*1024)/2;

Onde **x** corresponde à quantidade de memória disponível no servidor.

Utilizando a mesma quantidade de memória do exemplo anterior, o cálculo se dá da seguinte forma: 128*(1024*1024*1024)/2 = 68719476736.

O valor obtido como, por exemplo, 68719476736, deverá ser adicionado no parâmetro de configuração **kernel.shmmax** do sistema operacional para 128Gb de memória.

6.1.1.3 Parâmetro kernel.shmmni

O parâmetro **kernel.shmmni** corresponde ao número máximo de segmentos de memória compartilhada. Conforme o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o valor recomendado é 4096 para esse parâmetro, independente da quantidade de memória física disponível.

Dessa forma, abaixo um exemplo do arquivo de configuração /etc/sysctl.conf para 128Gb de memória.

```
kernel.shmall  = 33554432
kernel.shmmax = 68719476736
kernel.shmmni  = 4096
```

6.1.2 Memória Virtual

Nesta seção são abordados os parâmetros de configuração recomendados para melhor utilização da memória virtual para o banco de dados Oracle. Com base no *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), os

parâmetros recomendados para melhor utilização da memória virtual configurados no arquivo de configuração */etc/sysctl.conf*, são:

- vm.swappiness
- vm.dirty_background_ratio
- vm.dirty_ratio
- vm.dirty_expire_centisecs
- vm.dirty_writeback_centisecs

6.1.2.1 Parâmetro vm.swappiness

De acordo com o *Red Hat Enterprise Linux 6 Performance Tuning Guide* (2013), o parâmetro **vm.swappiness** controla o comportamento do sistema operacional para a realização das operações de troca entre a memória física e a memória *swap* no disco. Valores altos priorizam o desempenho do sistema, realizando um grande volume de trocas quando a memória não está ativa. Valores mais baixos priorizam a interatividade e evitam ao máximo o processo de troca, ou seja, somente em último caso o sistema utilizará a memória *swap*.

Consoante o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o valor recomendado para esse parâmetro é 0, fazendo assim com que o sistema operacional utilize a memória *swap* como último recurso.

6.1.2.2 Parâmetro vm.dirty_background_ratio

Para o *Red Hat Enterprise Linux 6 Performance Tuning Guide* (2013), o parâmetro **vm.dirty_background_ratio** inicia a gravação em segundo plano dos blocos sujos quando o percentual estipulado no parâmetro é alcançado. Por exemplo, se o servidor possuir 4Gb de memória e o parâmetro **vm.dirty_background_ratio** estiver em 10, o processo de gravação é iniciado quando atingir os 10% de blocos sujos dos 4Gb de memória.

O valor recomendado para esse parâmetro, para o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), é 3. Assim, quando o sistema atingir os 3% de blocos sujos sobre o total da memória, os blocos são gravados e a área de *cache* é liberada.

6.1.2.3 Parâmetro vm.dirty_ratio

O *Red Hat Enterprise Linux 6 Performance Tuning Guide* (2013) explica que o parâmetro **vm.dirty_ratio** inicia a gravação dos blocos sujos quando eles correspondem ao percentual estipulado no parâmetro. Por exemplo, se o servidor possuir 4Gb de memória e o parâmetro **vm.dirty_ratio** estiver com o valor de 20, sendo assim, quando o sistema possuir um total de 20% de memória alocada para blocos sujos, é iniciado o processo de gravação em disco.

O valor recomendado para esse parâmetro, segundo o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), é 15. Assim, quando atingir 15% do total da memória com blocos sujos, o processo de gravação dos blocos em disco é iniciado.

6.1.2.4 Parâmetro vm.dirty_expire_centisecs

Morreale (2008) sustenta que o parâmetro **vm.dirty_expire_centisecs** define quanto tempo é necessário para determinar quando um bloco sujo se torna ilegível, e quando um bloco precisa ser atualizado em memória para gravação em disco. Por exemplo, se o parâmetro está configurado para 500, quer dizer que em 500 centésimos de segundo os blocos são considerados ilegíveis.

O valor recomendado para esse parâmetro, de acordo com *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6*, (2012), é 500, sendo o valor-padrão do sistema operacional.

6.1.2.5 Parâmetro vm.dirty_writeback_centisecs

Morreale (2008) comenta que o parâmetro **vm.dirty_writeback_centisecs** define o tempo que o processo de gravação dos blocos sujos é iniciado. Por exemplo, se o valor para o foi configurado para 100, significa que o processo será iniciado a cada 100 centésimos de segundo.

O valor recomendado para esse parâmetro, na análise do *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), é 100, sendo o valor-padrão do sistema operacional.

No quadro abaixo, tem-se um exemplo do arquivo de configuração /etc/sysctl.conf com os parâmetros indicados para memória virtual.

```
vm.swappiness = 0
vm.dirty_background_ratio = 3
vm.dirty_ratio = 15
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100
```

Com base nos parâmetros definidos por *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), pode-se concluir que essa configuração evita ao máximo a utilização da memória *swap* pelo sistema operacional, fazendo com que o mesmo sempre utilize a memória disponível.

6.1.3 Semáforos

De acordo com o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o Linux provê semáforos para compartilhamento de informação entre os processos. O banco de dados Oracle necessita de informações específicas para o correto funcionamento do banco de dados.

Conforme o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), os valores recomendados para o parâmetro **kernel.sem** são 250 32000 100 128. Cada valor corresponde a:

- 250: número máximo de semáforos por conjunto;
- 32000: número máximo de semáforos para todo o sistema,
- 100: número máximo de operações por semáforos;
- 128: número máximo de indicadores de semáforos.

No quadro a seguir, tem-se um exemplo do arquivo de configuração */etc/sysctl.conf* para os semáforos.

```
kernel.sem = 250 32000 100 128
```

6.2 TUNING SUBSISTEMA DE DISCO

O subsistema de disco é muito importante para as operações do banco dados. Mesmo que o banco de dados realize a maioria das operações em memória utilizando os recursos de *buffer*, em algum momento esses dados que estão na memória deverão ser gravados no disco e o sistema operacional deverá conseguir atender a essas requisições o mais rápido possível.

6.2.1 Escalonador de disco

O *Red Hat Enterprise Linux 5 Tuning and Optimizing Red Hat Enterprise Linux for Oracle 9i and 10g Databases* (2008) explica que o escalonador de disco recomendado para utilização em ambiente com banco de dados Oracle é o *Deadline.* Conforme dito na Seção 1.4.2, o *deadline* possui a característica de ser o escalonador mais próximo de leitura/gravação em tempo real.

Ao realizar o Red Hat Linux para utilizar o escalonador *deadline*, é necessário realizar o comando ***echo deadline > /sys/block/DISPOSITIVO/queue/scheduler.***

6.2.2 Parâmetro fs.file-max

Segundo o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o parâmetro **fs.file-max** é responsável por determinar a quantidade de arquivos que o sistema operacional poderá abrir. A recomendação para o parâmetro fs.file-max, conforme o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), é 6815744, para garantir que o sistema operacional consiga abrir um grande número de arquivos conforme a carga de trabalho.

No quadro abaixo, tem-se um exemplo do arquivo de configuração */etc/sysctl.conf* para o máximo de arquivos abertos.

```
fs.file-max = 6815744
```

6.2.3 Parâmetro fs.aio-max-nr

De acordo com o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o parâmetro **fs.aio-max-nr** define a quantidade máxima de operações de entrada/saída não sincronizadas. Operações não sincronizadas são aquelas que podem iniciar outras tarefas sem a necessidade de esperar que a mesma seja completada.

Para o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o valor recomendado para o parâmetro **fs.aio-max-nr** é de 1048576, fazendo com que o sistema operacional suporte um maior número de operações não sincronizadas.

No quadro a seguir, tem-se um exemplo do arquivo de configuração */etc/sysctl.conf* para a quantidade máxima de operações não sincronizadas.

```
fs.aio-max-nr = 1048576
```

6.3 TUNING SUBSISTEMA DE REDE

O processo de *tuning* envolve uma grande quantidade de parâmetros e configurações para os mais variados tipos de aplicação. Conforme o objetivo do trabalho nesta seção, somente os parâmetros relacionados ao desempenho em banco de dados Oracle serão analisados.

6.3.1 Parâmetro net.core.rmem_default

Ciliendo (2004) orienta que o parâmetro **net.core.rmem_default** define o tamanho-padrão que é recebido no *buffer* presente no subsistema de rede. Esse parâmetro faz com que o sistema consiga receber maior quantidade de *bytes* no *buffer*.

Consoante o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o valor recomendado para esse parâmetro é 262144, que corresponde a um *buffer* de 256Kb.

No quadro a seguir, tem-se um exemplo do arquivo de configuração */etc/sysctl.conf* para o tamanho padrão de recebimento no buffer.

```
net.core.rmem_default = 262144
```

6.3.2 Parâmetro net.core.rmem_max

No posicionamento de Ciliendo (2004), o parâmetro **net.core.rmem_max** define o tamanho máximo de *bytes* que é recebido no *buffer.* Esse parâmetro faz com que o sistema operacional consiga lidar de forma eficiente quando pacotes de rede grandes chegam ao subsistema de rede.

Conforme o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o valor recomendado para esse parâmetro é 4194304, permitindo que sejam armazenado no *buffer* pacotes com no máximo de 4096Kb.

Abaixo, tem-se um exemplo do arquivo de configuração /etc/sysctl.conf para o tamanho máximo de pacotes recebidos no *buffer*.

```
net.core.rmem_max = 4194304
```

6.3.3 Parâmetro net.core.wmem_default

Ciliendo (2004) cita que o parâmetro **net.core.wmem_default** define o tamanho padrão de *bytes* que é enviado pelo *buffer*. Esse parâmetro faz com que o sistema consiga enviar mais pacotes conforme as requisições das aplicações.

O *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012) define qual o valor recomendado para esse parâmetro é 262144, fazendo com que seja enviado pelo *buffer* pacotes de 256Kb.

A seguir, tem-se um exemplo do arquivo de configuração */etc/sysctl.conf* para o tamanho padrão de envio pelo buffer.

```
net.core.wmem_default = 262144
```

6.3.4 Parâmetro net.core.wmem_max

Na perspectiva de Ciliendo (2004), o parâmetro **net.core.wmem_max** define o tamanho máximo de *bytes* que é enviado pelo *buffer.* Esse parâmetro faz com que o sistema operacional consiga enviar pacotes grandes para o destino quando for necessário.

Pelo *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o valor recomendado para esse parâmetro é 1048576, fazendo com que seja enviado pelo *buffer* pacotes com no máximo de 1024Kb.

Verifica-se, no quadro a seguir, um exemplo do arquivo de configuração /etc/sysctl.conf para o tamanho máximo de pacotes enviados pelo *buffer*.

```
net.core.wmem_max = 1048576
```

6.3.5 Parâmetro net.ipv4.ip_local_port_range

Segundo o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o parâmetro define o intervalo de portas que será realizado a conexão cliente-servidor. A recomendação para esse parâmetro são valores de 9000 65000. Isso faz com que o servidor Oracle realize a abertura das portas no intervalo de 9000 até 65000 para retorno das conexões dos clientes.

No quadro abaixo, tem-se um exemplo do arquivo de configuração /etc/sysctl.conf para o intervalo de portas.

```
net.ipv4.ip_local_port_range = 9000 65500
```

6.4 UTILIZANDO O TUNED E KTUNE

Para o *Red Hat Enterprise Linux 6 Performance Tuning Guide* (2013), o *tuned* é o processo que monitora e coleta dados de vários componentes do sistema, e dinamicamente realiza os ajustes necessários. Ou seja, faz com que os

componentes que são mais requisitados tenham os ajustes realizados automaticamente para obter o maior desempenho.

O *ktune* vem acompanhado com a ferramenta *tuned-adm,* que, pelo *Red Hat Enterprise Linux 6 Performance Tuning Guide* (2013), contém perfis de *tuning* pré-configurados para aumentar desempenho ou reduzir o consumo de energia conforme a necessidade.

O *tuned-adm* contém os seguintes perfis de *tuning*:

- **Default**: utilizado para redução do consumo de energia, sendo considerado o perfil básico onde o *tuned* e *ktune* estão desabilitados (Red Hat Enterprise Linux 6 Performance Tuning Guide, 2013);
- **Latency-Performance**: utilizado para desempenho de latência típica. Esse perfil desabilita os processos de *tuning* dinâmico e *transparent hugepages*.*Transparent Hugepages*. Segundo o *Red Hat Enterprise Linux 6 Performance Tuning Guide* (2013), é a técnica que faz com que o sistema trabalhe com páginas de memória muito maiores, pois o banco de dados Oracle suporta essa configuração, mas não recomenda sua utilização. Além disso, ele configura o escalonador de CPU para a *performance*, desabilitando os controles de energia e ajusta o escalonador de disco para *deadline* (Red Hat Enterprise Linux 6 Performance Tuning Guide, 2013);
- **Throughput-Performance**: utilizado em sistema com altas cargas de trabalho. Esse perfil desabilita todos os mecanismos de controle de energia e habilita os processos de *tuning* dinâmico. Além disso, ele configura o escalonador do processador para a *performance* e o escalonador de disco para *deadline.* Outro parâmetro que o perfil habilita é o *transparent hugepages* (Red Hat Enterprise Linux 6 Performance Tuning Guide, 2013);
- **Enterprise-Storage**: utilizado para altas cargas de trabalho em operações de entrada/saída. Esse perfil realiza as mesmas configurações que o *throughput-performance***.** Como diferencial, desabilita as barreiras do sistema de arquivos, fazendo que o mesmo não utilize os mecanismos de segurança com falhas, tais como desligamento incorreto ou falha de energia (Red Hat Enterprise Linux 6 Performance Tuning Guide, 2013);
- **Virtual-Guest**: utilizado para máquinas virtuais, emprega os mesmos parâmetros que o *enterprise-storage.* Sua principal diferença é a forma em

que o sistema é configurado para realizar *swap,* sendo ele voltado para desempenho das aplicações (Red Hat Enterprise Linux 6 Performance Tuning Guide, 2013);

- **Virtual-Host**: utilizado para servidores que hospedam máquinas virtuais, usa os mesmo parâmetros que o *enterprise-storage*, sendo a diferença os parâmetros relacionados à virtualização (Red Hat Enterprise Linux 6 Performance Tuning Guide, 2013).

Segundo o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), o perfil recomendado para utilização em banco de dados Oracle é *enterprise-storage*, que configura alta capacidade de recurso no subsistema de rede e disco.

Para isso, faz-se necessário a execução do seguinte comando: ***tuned-adm profile enterprise-storage***.

7 TUNING BANCO DE DADOS ORACLE

O banco de dados Oracle possui uma grande quantidade de parâmetros, que afetam a forma com que o banco de dados desempenha as tarefas. Uma vez que o objetivo do trabalho é obter melhor tempo de resposta do banco de dados Oracle, os parâmetros abordados neste capítulo são para esse objetivo.

7.1 TUNING NO GERENCIAMENTO DE MEMÓRIA

Watson (2010) considera que o uso da memória em uma instância do Oracle é crítico para o desempenho. A quantidade de memória alocada para a instância não deve ser nem muito pequena, que o banco de dados não consiga carregar em memória todos os objetos que ele utiliza com frequência, e nem muito grande, que possa fazer com que o sistema operacional não tenha memória suficiente para ele, causando *swap*.

Como visto no Capítulo 4, o banco de dados Oracle possui uma estrutura de memória complexa com vários parâmetros a serem verificados. Segundo o *Performance Tuning Guide 11g Release 2* (2012), a Oracle recomenda fortemente a utilização do recurso *Automatic Memory Management* ou Gerenciamento Automático de Memória, que faz com que a instância gerencie e aloque dinamicamente os recursos de memória necessários.

No entender de Watson (2010), além de facilitar a administração do banco de dados, essa configuração também proporciona grandes benefícios de desempenho. Conforme estudo realizado por Macedo *et al*. (2013), utilizando o gerenciamento automático de memória, o banco de dados conseguiu obter um melhor desempenho quando mais recursos foram exigidos.

Segundo o *Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6* (2012), a quantidade de memória para que pode ser alocado é correspondente ao parâmetro **kernel.shmmax**, visto no Capítulo 6.

O gerenciamento automático de memória realiza a gerência de memória de SGA e PGA da instância de banco de dados. Isso faz com que o administrador de banco de dados não necessite configurar manualmente todos os *buffers* e *caches* vistos no Capítulo 4.

Para realizar a configuração do gerenciamento automático de memória, por exemplo, para o um servidor com 128Gb de memória com somente uma instância de banco de dados rodando, é necessário executar os comandos que são apresentados no quadro abaixo.

```
alter system set db_cache_size=0 scope=spfile;
alter system set shared_pool=0 scope=spfile;
alter system set large_pool=0 scope=spfile;
alter system set java_pool=0 scope=spfile;
alter system set sga_taget=0 scope=spfile;
alter system set pga_aggregate_target=0 scope=spfile;
alter system set memory_target=64G scope=spfile;
```

Esses comandos zeram todas as configurações manuais aplicadas no banco de dados e ativaram 64G de memória para ser utilizado pelo gerenciamento automático de memória.

7.1.1 Monitorando o gerenciamento automático de memória

Mesmo que o administrador do banco de dados configure o gerenciamento automático de memória, é possível monitorar a forma com que ele está operando. Para isso, segundo o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), é possível coletar as informações necessárias através das seguintes *views* do banco de dados Oracle:

- *V$MEMORY_DYNAMIC_COMPONENTS*: essa *view* mostra a quantidade de memória alocada para cada recurso de *tuning* dinâmico do banco de dados;
- V$MEMORY_RESIZE_OPS: essa *view* mostra as informações das últimas oitocentas operações de realocação de memória realizada;
- *V$MEMORY_TARGET_ADVICE*: essa *view* mostra informações de conselhos de *tuning* a ser realizado no parâmetro *memory_target*;
- *V$SGA_DYNAMIC_FREE_MEMORY*: essa *view* mostra as informações da quantidade de memória livre dos componentes para futuras operações de realocação.

7.2 TUNING DO OTIMIZADOR

O banco de dados Oracle possui dois otimizadores de consulta: o *Rule-Based Optimizer* ou Otimizador Baseado em regras de negócio (RBO) e o *Cost-Based Optimizer* ou Otimizador Baseado no custo (CBO).

O otimizador-padrão do Oracle 11g é o CBO, sendo que o RBO somente é mantido para fins de compatibilidade. O CBO, na visão de Nunes (2009), utiliza estatísticas e histogramas para conseguir executar a consulta de uma maneira que utilize o mínimo possível de processamento.

O otimizador CBO trabalha com base nas estatísticas gravadas no dicionário de dados do banco de dados Oracle. Sendo que essa coleta de estatísticas pode ser realizada de forma manual ou automática. A coleta automática de estatística é realizada no período em que o banco de dados está ocioso. Em banco de dados que possuem em grande número de acessos, esse período de ociosidade pode não existir, sendo necessário realizar o processo de coleta de estatísticas de forma manual. Para isso é necessário a execução dos seguintes comandos:

```
EXEC DBMS_STATS.GATHER_DATABASE_STATS;
EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;
```

O comando EXEC DBMS_STATS.GATHER_DATABASE_STATS coleta estatísticas de todo o banco de dados. Já o comando EXEC DBMS_STATS.GATHER_DICTIONARY_STATS coleta estatística do dicionário de dados. Estatísticas do banco de dados desatualizadas podem comprometer o desempenho das consultas realizadas.

Nunes (2009) argumenta que as estatísticas geradas incluem as seguintes informações:

- **Tabelas**: número de linhas, número de blocos, média de comprimento da fila;
- **Colunas**: Número de valores distintos, número de valores nulos, distribuição de dados (histograma), estatísticas de extensão;
- **Índices**: número de blocos folha, níveis, fator de clusterização;
- **Sistema**: consumo de entrada/saída, consumo do processador.

7.2.1 Parâmetro OPTIMIZER_MODE

Conforme o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), o parâmetro *OPTIMIZER_MODE* configura o comportamento-padrão do otimizador. Esse parâmetro pode ser configurado das seguintes maneiras:

- **ALL_ROWS**: o otimizador usa uma abordagem baseada em custos para todas as instruções SQL na sessão, independentemente da presença de estatísticas, e otimiza com o objetivo de melhor rendimento (uso mínimo de recursos para completar a instrução inteira);
- **FIRST_ROWS_n**: o otimizador usa uma abordagem baseada em custos, independentemente da presença de estatísticas, e otimiza com o objetivo de melhor tempo de resposta para retornar o primeiro *n* número de linhas, onde *n* é igual a 1, 10, 100 ou 1000;
- **FIRST_ROWS**: o otimizador usa uma mistura de custos e heurística para encontrar um melhor plano para a entrega rápida das primeiras linhas.

O padrão para esse parâmetro é *ALL_ROWS*, sendo esse o recomendado.

7.2.2 Parâmetro OPTIMIZER_INDEX_CACHING

Pelas informações do *Oracle Database Performance Tuning Guide 11g Release 2* (2014), o parâmetro *OPTIMIZER_INDEX_CACHING* controla o custo de operações em laços de repetição aninhado. O valor para esse parâmetro pode variar de 0 até 100, que indica a porcentagem de blocos de índices que podem estar contidos no *buffer*. Se o valor do parâmetro está configurado como 100, quer dizer que 100% dos blocos estão contidos no *buffe*r. O otimizador irá ajustar o custo conforme o valor do parâmetro.

O valor-padrão para esse parâmetro é 0, fazendo com que o custo das operações seja mais alto, pois é passado para o otimizador que 0% dos blocos estão presentes no *buffer*. Segundo recomendação do *Oracle Database Performance Tuning Guide 11g Release 2* (2014), o valor recomendado para esse parâmetro é 0.

Para alterar esse parâmetro, é importante a utilização do seguinte comando: **alter system set OPTIMIZER_INDEX_CACHING=0 scope=spfile.**

7.2.3 Parâmetro OPTIMIZER_INDEX_COST_ADJ

Conforme o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), o parâmetro OPTIMIZER_INDEX_COST_ADJ controla o custo de utilização de índices ou na realização de *full-scan (*o otimizador percorre toda a tabela para encontrar o resultado desejado). Os valores para esse parâmetro podem variar de 1 até 10000, sendo o valor-padrão 100. Por exemplo, se o parâmetro está configurado para o valor 10, o custo na utilização de índice é um décimo do custo do acesso normal.

Segundo o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), o valor recomendado para utilização é 100. Para alterar esse parâmetro, deve-se fazer a utilização do seguinte comando: **alter system set OPTIMIZER_INDEX_COST_ADJ=100 scope=spfile.**

7.2.4 Parâmetro OPTIMIZER_FEATURES_ENABLE

O *Oracle Database Performance Tuning Guide 11g Release 2* (2014) diz que o parâmetro OPTIMIZER_FEATURES_ENABLE controla as funcionalidades do otimizador com base na versão do banco de dados Oracle instalado. A recomendação para esse parâmetro é sempre utilizar conforme a versão instalada e, assim, utilizar todas as funcionalidades e melhorias realizadas. O valor-padrão desse parâmetro é a versão do banco de dados instalado.

Para verificar qual é o valor configurado para esse parâmetro, executa-se o seguinte comando: **show parameter OPTIMIZER_FEATURES_ENABLE.**

7.3 PARÂMETRO PROCESS

Na compreensão do *Oracle Database Performance Tuning Guide 11g Release 2* (2014), esse parâmetro controla a quantidade de processos que a instância pode iniciar. O seu valor-padrão é 100. Esse parâmetro não possui uma recomendação, sendo de valor a ser configurado, pois está diretamente ligado à carga de trabalho que o banco de dados Oracle será submetido. O importante desse parâmetro é garantir que a instância consiga criar a quantidade necessária de processos para atender os usuários.

Para alterar esse parâmetro, faz-se necessário a utilização do seguinte comando: **alter system set process=1024 scope=spfile.**

7.4 PARÂMETRO SESSIONS

Conforme o Oracle *Database Performance Tuning Guide 11g Release 2* (2014), esse parâmetro controla a quantidade de sessões que a instância pode iniciar. Seu valor-padrão é o mesmo valor configurado para o parâmetro PROCESS. A recomendação, segundo Database Reference (p.226), o valor para esse parâmetro se dá através da equação:

(1.1* PROCESS) + 5

PROCESS é o valor configurado no parâmetro no banco de dados Oracle. Para configurar esse parâmetro, precisa-se da execução do seguinte comando: **alter system set sessions=1024 scope=spfile.**

7.5 PARÂMETRO OPEN_CURSORS

O *Database Reference* (2014) considera que esse parâmetro controla a quantidade de cursores, que são áreas de armazenamento temporárias de consultas, que a instância de banco de dados pode iniciar. Os valores desse parâmetro podem variar de 0 até 65535, sendo o valor-padrão 50. Segundo o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), o valor recomendado para esse parâmetro é 500. Assim, é possível garantir que a instância consiga criar os cursores necessários, mas também inibe a criação de muitos cursores para não degradar o desempenho do banco de dados.

Para configurar esse parâmetro, necessita-se da execução do seguinte comando: **alter system set open_cursors=500 scope=spfile.**

7.6 PARÂMETRO SESSION_CACHED_CURSORS

Consoante o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), esse parâmetro controla a quantidade de cursores de sessão que são armazenados em *cache*. Para o *Database Reference* (2014), as instruções de SQL, que rodam várias vezes, fazem com que o cursor seja movido para a área de *cache*, sendo assim instruções subsequentes utilizarão o cursor em *cachê*, não havendo a necessidade de abrir o mesmo novamente.

O valor-padrão para esse parâmetro é 50, sendo que para esse parâmetro não existe um valor recomendado. Segundo o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), é possível determinar se o valor-padrão está atendendo de forma eficaz. Para isso, deve-se executar os seguintes comandos:

```
SELECT a.value curr_cached, p.value max_cached,
s.username, s.sid, s.serial#
FROM v$sesstat a, v$statname b, v$session s, v$parameter2 p
WHERE a.statistic# = b.statistic# and s.sid=a.sid and a.sid=&sid
AND p.name='session_cached_cursors'
AND b.name = 'session cursor cache count';
```

Comando retorna a quantidade de cursores que estão contidos no *cache* da sessão passada por parâmetro.

```
SELECT cach.value cache_hits, prs.value all_parses,
round((cach.value/prs.value)*100,2) as "% found in cache"
FROM v$sesstat cach, v$sesstat prs, v$statname nm1, v$statname nm2
WHERE cach.statistic# = nm1.statistic#
AND nm1.name = 'session cursor cache hits'
AND prs.statistic#=nm2.statistic#
AND nm2.name= 'parse count (total)'
AND cach.sid= &sid and prs.sid= cach.sid;
```

Comando que retorna a quantidade de cursores que foram encontrados no *cache*.

Com base nos resultados obtidos nas consultas, é possível determinar se o parâmetro foi configurado da forma correta.

Para configurar esse parâmetro, faz-se necessário a execução do seguinte comando: **alter system set session_cached_cursors=100 scope=spfile.**

7.7 PARÂMETRO DB_WRITER_PROCESSES

O *Oracle Database Performance Tuning Guide 11g Release 2* (2014) entende que esse parâmetro controla a quantidade de processos de escrita (DVWn), os quais são iniciados com a instância. O valor-padrão para esse parâmetro é 1. Com a configuração de múltiplos processos de escrita, o trabalho de identificação de *buffers* a serem gravados e as operações de entrada/saída são divididos entre os processos.

Segundo o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), a utilização de múltiplos processos de escrita é altamente recomendado para ambientes com grande volumes de operações de entrada e saída.

A recomendação do *Oracle Database Performance Tuning Guide 11g Release 2* (2014) para esse parâmetro, é a utilização de um processo de escrita para cada 8 processadores presentes.

Para configurar esse parâmetro, observa-se a execução do seguinte comando: **alter system set DB_WRITER_PROCESSES=4 scope=spfile**

7.8 PARÂMETRO FILESYSTEMIO_OPTIONS

De acordo com o *Oracle Database Performance Tuning Guide 11g Release 2* (2014), esse parâmetro habilita ou desabilita operações de entrada/saída assíncrona ou operações de entrada/saída direta. Tal parâmetro suporta os seguintes valores:

- **ASYNCH**: habilita operações de entrada/saída assíncrona no sistema de arquivos, que não possuem nenhuma exigência de tempo para transmissão;
- **DIRECTIO**: habilita operações de entrada/saída diretamente no sistema de arquivos, ignorando o *buffer cachê*;
- **SETALL**: habilita as operações assíncronas e diretas no sistema de arquivos;
- **NONE**: desabilita as operações assíncronas e diretas no sistema de arquivos.

Conforme recomendação do *Oracle Database Performance Tuning Guide 11g Release 2* (2014), o valor recomendado para esse parâmetro é *SETALL*, habilitando ambas as formas de acesso ao sistema arquivos, e delegando ao banco de dados Oracle quando houver a necessidade de uso de cada um dos métodos.

Para configurar esse parâmetro, faz-se necessário a execução do seguinte comando: **alter system set filesystemio_options=setall scope=spfile.**

8 PROPOSTA DE SOLUÇÃO

A proposta de solução do presente estudo consiste em realizar as configurações necessárias para que o sistema operacional *Red Hat Linux* disponibilize todos os recursos necessários para que o banco de dados Oracle atenda as requisições da forma mais eficiente.

Para isso, a proposta de solução é dívida nas seguintes partes:

- Preparação ao ambiente.
- Aplicação do processo de *tuning* no sistema operacional.
- Aplicação do processo de *tuning* no banco de dados Oracle.
- Submissão do banco de dados a testes de *benchmark* utilizando a ferramenta *Benchmark Factory for Database*.
- Análise dos resultados.

8.1 BENCHMARK

Para validação do processo de *tuning* abordado no trabalho proposto e comparação dos resultados obtidos através do processo, o banco de dados será submetido a testes de *benchmark*.

Na análise de Novais (2006), *benchmark* é um programa utilizado para testar *hardware*, *software* ou um sistema. Quando utilizado em banco de dados, é um conjunto de instruções utilizadas para medir e comparar o desempenho de dois ou mais bancos de dados. Além de comparações entre bancos de dados, o *benchmark* pode ser utilizado para validação de configurações efetuadas e comparação de resultados no processo de *tuning*.

Novais (2006) explica que o processo de *benchmark* envolve três etapas:

- Definição do sistema a ser testado;
- Definição da carga de trabalho submetida; e
- Estabelecimento de métricas de resultado como tempo de resposta.

Para bancos de dados relacionais, na visão de Novais (2006), os testes de *benchmark* são definidos pela *Transaction Processing Performance Council* (TPC), que é um consórcio de empresas fabricantes de *hardware* e *software* sem fins

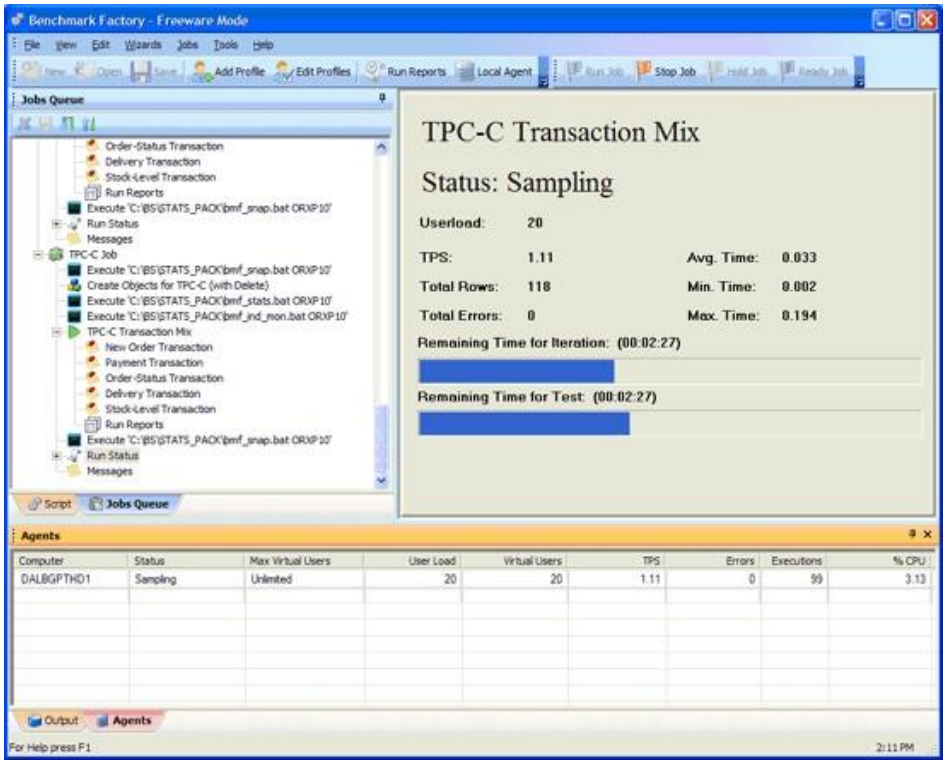
lucrativos, que define testes de *benchmark* para vários fins, sendo um deles os bancos de dados.

Para banco de dados, o consórcio definiu o *benchmark* TPC-C que, segundo Novais (2006), é utilizado para processamento de transações *on-line* (OLTP). Esse *benchmark* simula um sistema genérico de OLTP, que consiste em um sistema de gerenciamento de estoque.

A ferramenta para a realização dos testes de *benchmark* é a *Benchmark Factory for Database* (http://software.dell.com/products/benchmark-factory/). Na compreensão de *Benchmark Factory™ for Databases* (2014), é a ferramenta de teste que simula usuários e transações banco de dados. A ferramenta atende a todas as etapas no processo de *benchmark*, pois nela é possível ajustar a carga de trabalho no teste e realizar comparações entre os testes realizados assim, também servindo com uma linha base no processo de *tuning*.

A ferramenta, conforme *Benchmark Factory™ for Databases* (2014), utiliza os testes-padrão de mercado para realização dos testes. A Figura 10 ilustra a execução de um teste de *benchmark* sendo executado pela ferramenta.

Figura 10 – Exemplo de execução do *Benchmark Factory for Database*



Fonte: http://www.toadworld.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-00-00-57/BertBlog021308_2D00_3.gif

8.2 PREPARAÇÃO DO AMBIENTE

Nesta etapa, prepara-se o ambiente para a realização das demais etapas da proposta, a qual consiste na instalação do ambiente com as configurações indicadas na Tabela 2.

Tabela 2 – Características do ambiente

Servidor	Dell R710
Processador	4 Intel(R) Xeon(R) CPU X5690 @ 3.47GHz
Memória	24Gb
Rede	1Gbps
Disco	1,2Tb
Sistema Operacional	Red Hat Linux 6.5 64 Bits
Banco de Dados	Oracle Database 11.2.0.4

Os pacotes instalados no sistema operacional são somente os necessários para que o Oracle Database possa ser instalado, sendo que nenhum outro serviço adicional será instalado. No Oracle Database serão instalados todos os últimos pacotes de correção. Após a instalação do banco de dados Oracle, o mesmo será submetido ao teste de *benchmark* TPC-C através da ferramenta *Benchmark Factory for Database.* O teste de *benchmark* será realizado com as seguintes características:

Tabela 3 – Característica do teste de *benchmark*

Número de Usuários	700
Quantidade de Dados	64Gb

8.3 APLICAÇÃO DO PROCESSO DE TUNING NO SISTEMA OPERACIONAL

Nesta etapa serão aplicadas as configurações de *tuning* vistas no Capítulo 6, utilizando-se como base o *hardware* disponível. Para isso são necessárias as seguintes configurações no arquivo */etc/sysctl.conf*:

```
kernel.shmmax = 12884901888
kernel.shmmni = 4096
kernel.shmall  = 3145728
vm.swappiness = 0
vm.dirty_background_ratio = 3
vm.dirty_ratio = 15
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100
kernel.sem = 250 32000 100 128
fs.file-max = 6815744
fs.aio-max-nr = 1048576
net.core.rmem_default = 262144
net.core.wmem_default = 262144
net.core.wmem_default = 1048576
net.ipv4.ip_local_port_range = 9000 65500
```

Neste caso serão utilizados todos os parâmetros recomendados pelo levantamento bibliográfico. Outra recomendação é a utilização do *Tuned* e *Ktune* com o perfil de configuração *enterprise-storage*.

8.4 APLICAÇÃO DO PROCESSO DE TUNING NO BANCO DE DADOS ORACLE

Nesta etapa serão realizadas as configurações de *tuning* no banco de dados Oracle apresentadas no Capítulo 7. Para isso é necessário a execução dos comandos apresentados no quadro abaixo.

```
alter system set db_cache_size=0 scope=spfile;
alter system set shared_pool=0 scope=spfile;
alter system set large_pool=0 scope=spfile;
alter system set java_pool=0 scope=spfile;
alter system set sga_taget=0 scope=spfile;
alter system set pga_aggregate_target=0 scope=spfile;
alter system set memory_target=12G scope=spfile;
alter system set optimizer_index_caching=100 scope=spfile;
alter system set optimizer_index_cost_adj=0 scope=spfile;
alter system set process=1024 scope=spfile;
alter system set sessions=1024 scope=spfile;
alter system set open_cursors=500 scope=spfile;
alter system set filesystemio_options=setall scope=spfile;
```

Após a configuração dos comandos, é necessário reiniciar o banco de dados para que o mesmo inicie com os valores configurados.

8.5 APLICAÇÃO DOS TESTES DE BENCHMARK

Nesta etapa, o banco de dados será submetido ao teste de *benchmark* TPC-C através da ferramenta *Benchmark Factory for Database*, sendo utilizada como base a mesma carga da Seção 8.2 da Proposta de Solução. A cada início e fim do teste de *benchmark* será criado um *snapshot* através da ferramenta *StatsPack* para a avaliação da parametrização do banco de dados.

8.6 ANÁLISE DOS RESULTADOS

Com base nos resultados obtidos nos testes de *benchmark* e na ferramenta *StatsPack,* será avaliado se o processo de *tuning* melhorou o tempo de resposta do banco de dados e se o mesmo conseguiu atender a mais requisições, comparando com a configuração-padrão de sistema operacional e banco de dados.

O que ocorre nesta etapa é um processo cíclico que abrange o processo de *tuning* de sistema operacional, *tuning* de banco de dados e teste de *benchmark*,

sendo repetido até que seja possível reduzir em 20% do tempo de resposta e que se possa realizar 25% a mais de transações.

9 TESTES DE BENCHMARK

Como fase inicial para a realização da proposta de solução, foi feita a instalação e configuração do *Red Hat Linux 6.5* e *Oracle Database 11.0.4*, conforme descrito no Capítulo 9. Nessa etapa, não foi realizada nenhuma configuração adicional no sistema operacional e banco de dados.

Além da configuração do sistema operacional e banco de dados, fez-se necessário a instalação do *Benchmark Factory for Database* em sua versão *trial* (funcionamento por 30 dias), *software* responsável pela realização dos testes.

9.1 ESTRUTURA DOS TESTES

Após a instalação do *Benchmark Factory for Database*, foi constatado que não seria possível utilizar uma carga de 700 usuários, devido a uma restrição de licenciamento do *software* que só permite em sua versão *trial* 100 usuários, conforme a proposta de solução descrita no Capítulo 8. Para contornar essa situação, os testes serão realizados conforme a tabela a seguir.

Tabela 4 – Estrutura de carga de trabalho

Carga de Usuário	Tempo de Iteração
20	5 minutos
40	5 minutos
60	5 minutos
80	5 minutos
100	5 minutos

Na primeira rodada de testes, utiliza-se uma carga de 20 usuários, realizando execuções do banco de dados em um período de 5 minutos. Na próxima iteração, a carga de usuários sobe para 40, utilizando o mesmo período de 5 minutos, assim até chegar ao máximo de 100 usuários.

A quantidade de dados inserida no banco de dados permaneceu a mesma descrita no Capítulo 8, bem como o tipo de teste a ser realizado.

9.2 REALIZAÇÃO DOS TESTES E COLETA DOS RESULTADOS

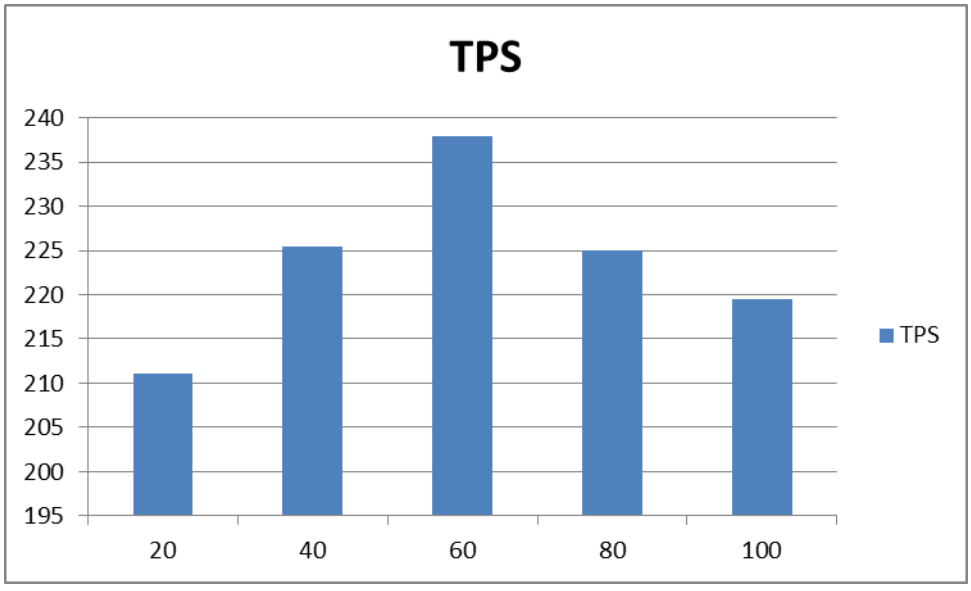
Como abordado no Capítulo 2.4, o primeiro passo no processo de *tuning* é o estabelecimento da linha base. Para a criação da linha base, utilizou-se as configurações-padrões de sistema operacional e banco de dados, sendo que os resultados obtidos para cada iteração estão descritos na tabela a seguir.

Tabela 5 – Resultados obtidos na linha base

Usuários	TPS	Execuções	Linhas	Tempo de Resposta Médio
20	211,04	37975	2414416	0,002
40	225,39	40488	2565334	0,005
60	237,88	42632	2719428	0,007
80	224,99	40328	2631396	0,010
100	219,54	39172	2537350	0,013

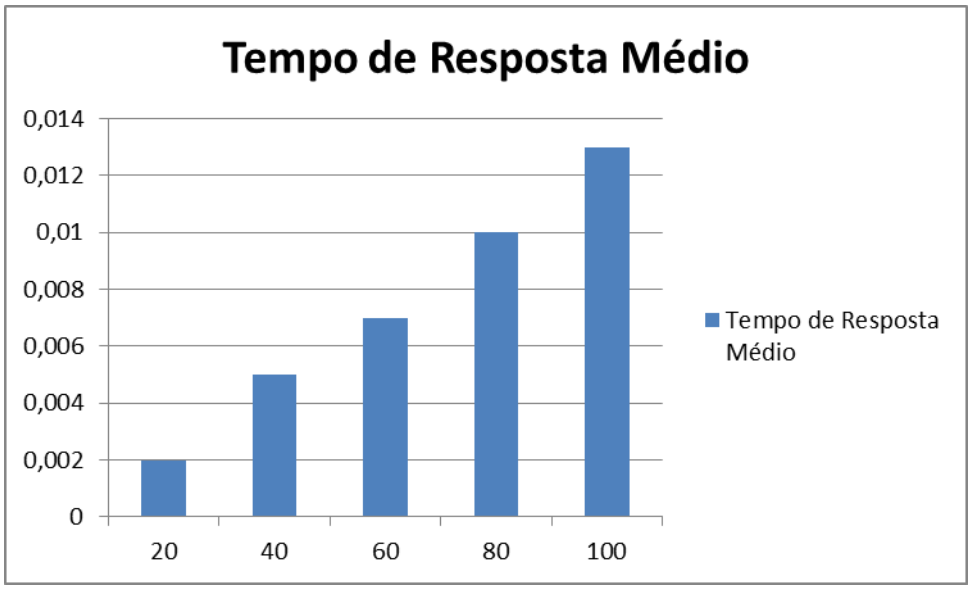
O Gráfico 1 ilustra a taxa de transações por segundo entre cada iteração realizada no banco de dados.

Gráfico 1 – Taxa de transações por segundo para cada iteração



Já o Gráfico 2 ilustra o tempo médio de resposta obtido em cada iteração com o banco de dados.

Gráfico 2 – Tempo de resposta médio para cada iteração



Com base nos resultados obtidos utilizando a configuração-padrão do sistema operacional e banco de dados, pode-se verificar que com essa configuração a taxa máxima de transferência que o servidor atingiu foi na iteração com 60 usuários.

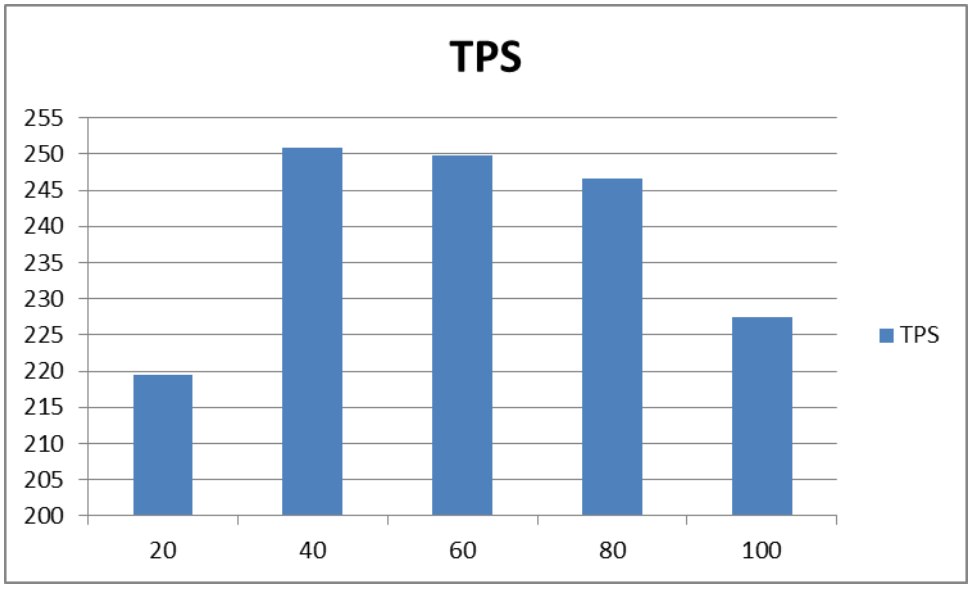
Após a obtenção da linha base, foi realizado o processo de *tuning* no sistema operacional *Red Hat Linux,* conforme descrito no Capítulo 8.3. Os resultados obtidos para cada iteração estão descritos na tabela a seguir.

Tabela 6 – Resultados obtidos com *tuning* de sistema operacional

Usuários	TPS	Execuções	Linhas	Tempo de Resposta Médio
20	219,46	39476	2517292	0,002
40	250,91	44449	2823210	0,004
60	249,84	44717	2871350	0,007
80	246,64	43964	2842281	0,009
100	227,40	40882	2662658	0,013

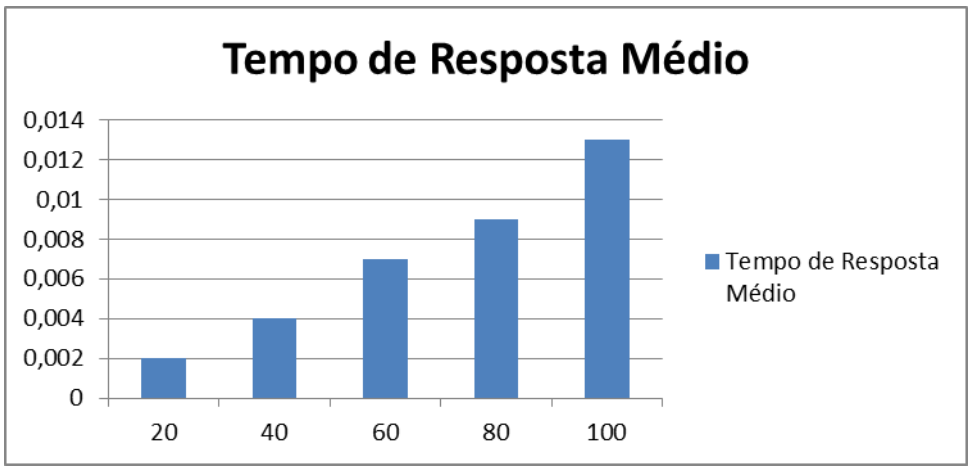
O Gráfico 3 ilustra a taxa de transações por segundo entre as iterações utilizando as configurações de *tuning* propostas para o sistema operacional.

Gráfico 3 – Taxa de transações por segundo para cada iteração (*tuning* sistema operacional)



O Gráfico 4 ilustra o tempo médio de resposta obtido em cada iteração com o banco de dados utilizando o processo de *tuning* de sistema operacional.

Gráfico 4 – Tempo de resposta médio para cada iteração (*tuning* sistema operacional)



Com base nos resultados obtidos utilizando o processo de *tuning* proposto no Capítulo 8.3, pode-se verificar que com essa configuração a taxa máxima de transferência que o servidor atingiu foi na iteração com 40 usuários.

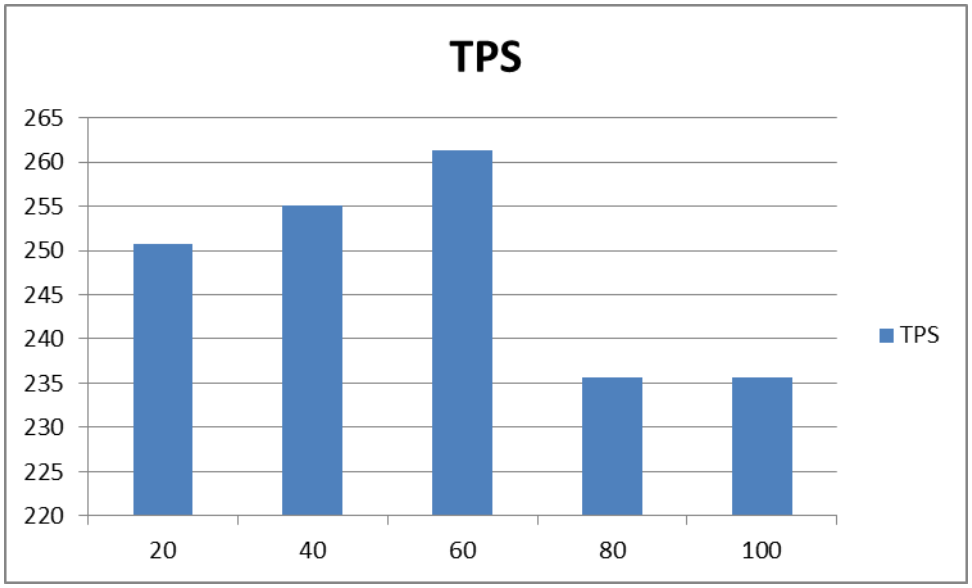
Após a obtenção dos resultados utilizando o processo de *tuning* proposto no Capítulo 8.3, foi realizado o processo de *tuning* no banco de dados, conforme descrito no Capítulo 8.4. Os resultados obtidos para cada iteração estão descritos na tabela a seguir.

Tabela 7 – Resultados obtidos com *tuning* do banco de dados

Usuários	TPS	Execuções	Linhas	Tempo de Resposta Médio
20	250,69	45109	2871919	0,002
40	255,08	45785	2909635	0,004
60	261,28	47074	3034358	0,006
80	235,68	41833	2708353	0,009
100	235,64	42677	2774684	0,012

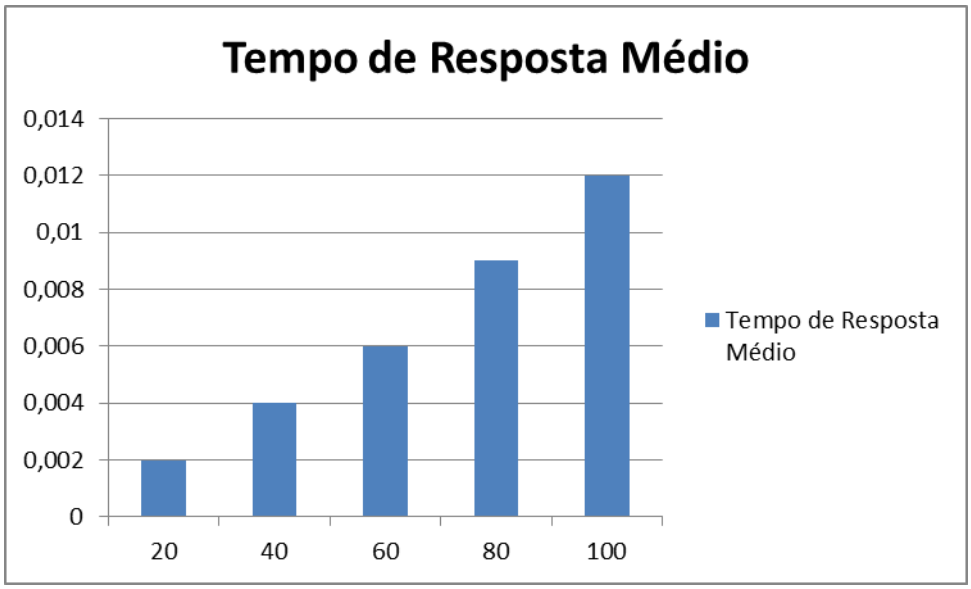
O Gráfico 5 ilustra a taxa de transações por segundo entre as iterações utilizando as configurações de *tuning* propostas para o banco de dados.

Gráfico 5 – Taxa de transações por segundo para cada iteração (*tuning* banco de dados)



O Gráfico 6 o tempo médio de resposta obtido em cada iteração com o banco de dados utilizando o processo de *tuning* no banco de dados.

Gráfico 6 – Tempo de resposta médio para cada iteração (*tuning* banco de dados)



Com base nos resultados obtidos utilizando o processo de *tuning* proposto no Capítulo 8.4, pode-se verificar que com essa configuração a taxa máxima de transferência que o servidor atingiu foi na iteração com 60 usuários.

9.3 COMPARAÇÃO DOS RESULTADOS

Com os resultados obtidos através da execução da ferramenta *Benchmark Factory for Database*, segue a tabela comparativa com as médias de todas as iterações realizadas entre a execução, utilizando as configurações-padrões e as configurações propostas no Capítulo 8.3.

Tabela 8 – Comparação entre a linha base e *tuning* do sistema operacional da média dos resultados obtidos

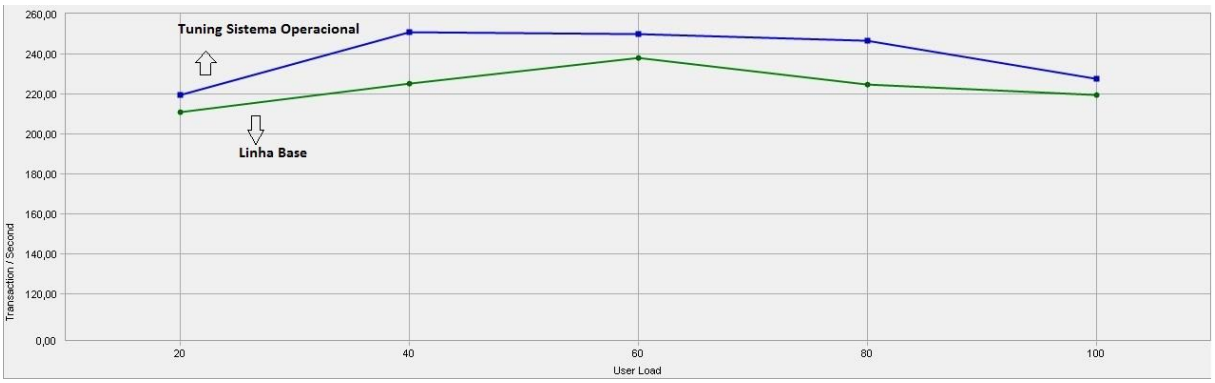
Configuração	Média de TPS	Média de Execuções	Média de Linhas	Média de Tempo de Resposta
Linha Base	223,77	40119	2573585	0,0074
Tuning Linux	238,85	42698	2743358	0,007
Variação	6,73%	6,42%	6,59%	-5,71%

Como se pode observar na tabela, somente com as configurações de melhores práticas do sistema operacional já é possível aumentar o desempenho do

banco de dados, fazendo que ele tenha um aumento médio de 6,73% nas transações por segundo e com uma redução média de 5,71% no tempo de resposta do mesmo.

O Gráfico 7 a seguir ilustra a diferença de transações por segundo para cada iteração realizada na configuração-padrão e com o processo de *tuning* aplicado no sistema operacional.

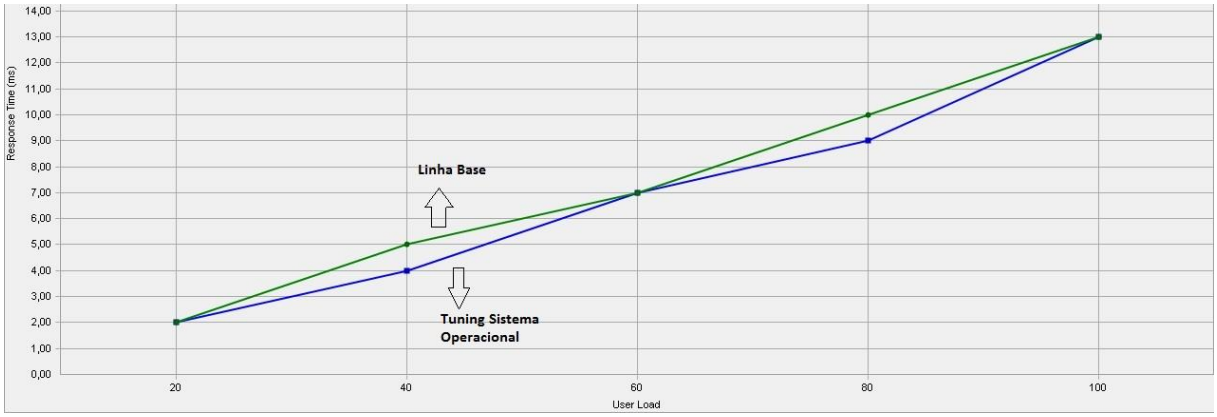
Gráfico 7 – Comparação do número de transações entre a linha base e *tuning* do sistema operacional



Como se pode observar no Gráfico 7, com as configurações de sistema operacional propostas no Capítulo 8.3, utilizando as melhores práticas recomendadas e fazendo com o que o Linux utilize de forma mais eficiente os recursos de memória, disco e rede em todas as cargas de trabalho o número de transações por segundo foi superior a linha base utilizando a configuração padrão do sistema.

O Gráfico 8 a seguir ilustra a diferença dos tempos de resposta para cada iteração realizada na configuração-padrão e com o processo de *tuning* realizado no sistema operacional.

Gráfico 8 – Comparação do tempo de resposta entre a linha base e *tuning* do sistema operacional



Como se pode observar no Gráfico 8, o tempo de resposta do processo de *tuning* do sistema operacional apresentou um melhor desempenho com carga de 40 e 80 usuários executando operações no banco de dados. Com isso, pode-se concluir que as configurações realizadas no subsistema de rede e memória do sistema fez que o banco de dados retornasse ao cliente de forma mais eficiente.

A tabela a seguir mostra a comparação entre a linha base e o processo de *tuning* realizado no banco de dados conforme o Capítulo 8.4.

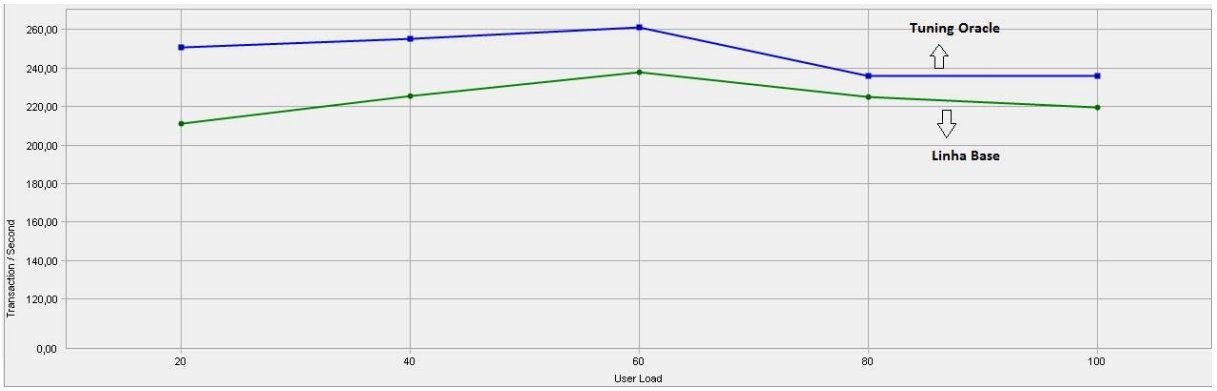
Tabela 9 – Comparação entre a linha base e *tuning* do banco de dados da média dos resultados obtidos

Configuração	Média de TPS	Média de Execuções	Média de Linhas	Média de Tempo de Resposta
Linha Base	223,77	40119	2573585	0,0074
Tuning Oracle	247,68	44496	2859790	0,0066
Variação	10,68%	10,91%	11,12%	-10,81%

Como se pode observar na tabela, utilizando-se as configurações de melhores práticas do sistema operacional e as configurações de melhores práticas do banco de dados, é possível aumentar o desempenho do banco de dados, fazendo que ele tenha um aumento médio de 10,73% nas transações por segundo e com uma redução média de 10,81% no tempo de resposta do mesmo.

O Gráfico 9 a seguir ilustra a diferença de transações por segundo para cada iteração realizada na configuração-padrão e com o processo de *tuning* aplicado no sistema operacional e banco de dados.

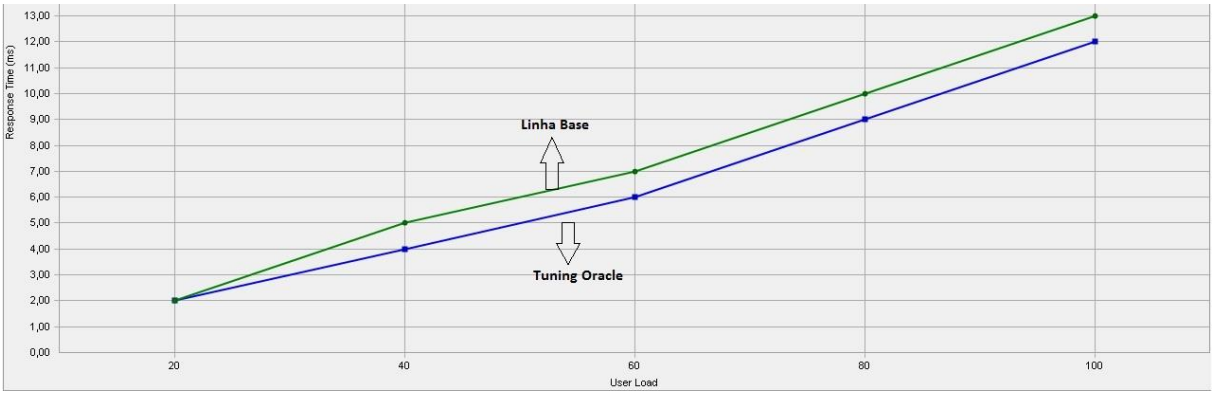
Gráfico 9 – Comparação do número de transações entre a linha base e *tuning* do banco de dados



Como se pode observar, na iteração com 20 usuários foi onde se obteve a maior diferença entre as configurações, já na iteração com 80 usuários foi onde se obteve a menor diferença entre as configurações. Com isso, pode-se concluir que o correto dimensionamento na utilização de memória do banco de dados faz com que ele consiga manter em *cache* as operações mais executadas, assim executando mais transações por segundo.

O Gráfico 10 a seguir ilustra a diferença dos tempos de resposta para cada iteração realizada na configuração-padrão e com o processo de *tuning* realizado no sistema operacional e banco de dados.

Gráfico 10 – Comparação do tempo de resposta entre a linha base e *tuning* do banco de dados



Como se pode observar no Gráfico 10, utilizando o correto dimensionamento dos parâmetros do banco de dados, juntamente com as configurações realizadas no sistema operacional, fez com que o banco de dados respondesse de forma mais eficiente sem perder o desempenho na execução das tarefas.

Na execução dos testes de *benchmark*, utilizando os parâmetros descritos no Capítulo 8, foi possível identificar através das ferramentas de monitoramento do sistema operacional que o subsistema de disco atingiu sua taxa máxima de transferência durante os testes utilizando as configurações propostas do sistema operacional. Para aumentar o desempenho do subsistema de disco seria necessário a adição de mais discos ao conjunto utilizado. Por esse motivo não foi possível atingir o objetivo de redução do tempo de resposta e transações por segundo.

10 CONCLUSÃO

O presente trabalho teve como objetivo realizar o processo de *tuning* no sistema operacional *Red Hat Linux* e banco de dados *Oracle*, sendo que o mesmo foi executado no banco de dados, envolvendo sua correta parametrização e configuração. Assim, é possível concluir que o processo de *tuning* como um todo envolve um profundo conhecimento dos componentes que farão parte do processo, tais como: sistema operacional, banco de dados ou alguma aplicação.

O processo de *tuning* de sistema operacional proposto no trabalho está direcionado para servir na utilização de banco de Oracle, mas algumas premissas abordadas também podem servir como base no processo de *tuning* para outros bancos de dados, como configuração do escalonador de disco e como o *Linux* faz a gerência da memória, as quais são boas práticas a serem aplicadas.

Já o processo de *tuning* abordado para banco de dados Oracle, foca em seu correto dimensionamento na utilização dos recursos de *hardware* e sistema operacional em que o mesmo se encontra instalado. Como por exemplo, o mesmo *tuning* aplicado no *Oracle* instalado no *Linux* é diferente se o *Oracle* estiver instalado em outro sistema operacional. O dimensionamento incorreto na utilização de memória do banco de dados faz com que o sistema operacional não tenha a capacidade de realizar o *cache* de dados, bem como armazenar objetos necessários em memória. O dimensionamento incorreto de memória também afeta o desempenho do banco de dados que, no cenário ideal, deve conseguir carregar todos os objetos mais utilizados em memória, evitando o acesso a disco que é mais lento.

Além do processo de *tuning* a ser realizado no sistema operacional e aplicação, faz-se necessário um entendimento no *hardware* onde essa aplicação está sendo hospedada. O resultado dos testes não foi bom, devido a problemas de desempenho do subsistema de disco que atingiu sua taxa máxima de leitura e gravação.

O *tuning* somente deverá ser realizado se o sistema estiver apresentando problemas de desempenho, pois efetuar esse processo sem necessidade pode causar mais problemas que benefícios. Por isso, antes de realizar qualquer processo de *tuning* é necessário analisar o sistema operacional através de suas ferramentas de diagnóstico, como também identificar os gargalos e corrigi-los, sendo através de

um ajuste de configuração ou no aumento na capacidade de *hardware*. Já em bancos de dados *Oracle,* é necessário analisar as estatísticas do banco utilizando ferramentas como *Statspack* ou *AWR*, de modo que o banco ainda fornece informações de *tuning* através de suas *VIEWS*.

Através do desenvolvimento desse trabalho foi possível aplicar as técnicas de *tuning* apresentadas nos sistemas operacionais *Red Hat Linux* e banco de dados *Oracle* utilizados na Universidade de Caxias do Sul. Com a das melhores práticas estudadas e utilizando *hardwares* mais robustos foi possível melhorar o desempenho de todas as aplicações que utilizam o banco de dados *Oracle*.

O presente trabalho fez a abordagem do processo de *tuning* e sistema operacional e banco de dados focando unicamente em sua parametrização, a próxima etapa para melhorar ainda mais o desempenho do banco de dados é o *tuning* de objetos do banco de dados, como por exemplo a criação de índices ou ajuste em funções e procedimentos.

REFERÊNCIAS

- REFERÊNCIAS BIBLIOGRÁFICAS

BENEDETTI, Davi Martinelli. **Tuning de Banco de Dados.** 2004. 101 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade de Caxias do Sul, Caxias do Sul, 2004.

BRAGHETTO, Kelly Rosa. **Padrões de fluxos de processos em banco de dados relacionais.** 2006. 36 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade de São Paulo, São Paulo, 2006.

CILIENDO, Eduardo. **Tuning Red Hat Enterprise Linux on IBM Eserver xSeries Servers.** 2004. Disponível em: <http://www.redbooks.ibm.com/redpapers/pdfs/redp3861.pdf>. Acesso em: 02 out. 2014.

CILIENDO, Eduardo; KUNIMASA, Takechika. **Linux Performance and Tuning Guidelines.** 2007. Disponível em: <http://www.redbooks.ibm.com/redpapers/pdfs/redp4285.pdf>. Acesso em: 26 set. 2014.

DELL SOFTWARE. **Benchmark Factory™ for Databases.** 2014. Disponível em: <http://www.quest.com/documents/dsd-benchmark-us-ag-20100122.pdf>. Acesso em: 11 out. 2014.

DIAS, Karl et al. **Automatic Performance Diagnosis and Tuning in Oracle**. Redwood Shores: 2004.

DJORDJEVIC, Borislav; TIMCENKO, Valentina. Ext4 file system in Linux Environment: Features and Performance Analysis. **International Journal Of Computers.**Hangzhou, p. 37-45. 06 abr. 2006.

ELMASRI, Ramez; NAVATHE, Shamkant B.. **Sistemas de Banco de Dados**. 6. ed. Arlington: Pearson, 2011.

JONES, Tim. **Inside the Linux 2.6 Completely Fair Scheduler.** 2009. Disponível em: <http://www.ibm.com/developerworks/library/l-completely-fair-scheduler/>. Acesso em: 25 set. 2014

MACEDO, Bruno P. et al. **Uma abordagem prática sobre otimização de Banco de Dados utilizando o gerenciamento automático de memória no Sistema Gerenciador de Banco de Dados Oracle**. Bauru: 2013.

MORREALE, Peter W.. **Documentation for /proc/sys/vm/* kernel version 2.6.29.** Disponível em: <https://www.kernel.org/doc/Documentation/sysctl/vm.txt>. Acesso em: 03 out. 2014.

NEMETH, Evi; SNYDER, Garth; HEIN, Trent R.. **Manual Completo do Linux: Guia do Administrador. 2. ed**. São Paulo: Pearson Prentice Hall, 2007.

NOVAIS, José Luís Pereira. **Benchmark de Bases de Dados de Suporte a Serviços de Informação.** 2006. 160 f. Dissertação (Mestrado) - Curso de Sistemas de Informação, Universidade do Minho, Braga, 2006.

NUNES, João Paulo Geraldini. **Diferenças entre os otimizadores de consulta do banco de dados Oracle baseado em regras e baseado em custo.** 2009. 24 f. TCC (Graduação) - Curso de Engenharia de Computação, Universidade de São Francisco, Itatiba, 2009.

PRADO, Fabio. **O que é Tuning?** 2014. Disponível em: <http://www.fabioprado.net/2014/04/o-que-e-tuning-como-tunar.html>. Acesso em: 01 out. 2014.

ORACLE CORPORATION. **Oracle9i Database Performance Tuning Guide and Reference.** 2002. Disponível em:

<https://docs.oracle.com/cd/B10501_01/server.920/a96533.pdf>. Acesso em: 10 out. 2014.

ORACLE CORPORATION. **Oracle® Database Performance Tuning Guide 11g Release 2 (11.2).** 2014. Disponível em: < https://docs.oracle.com/cd/E11882_01/server.112/e41573.pdf >. Acesso em: 11 out. 2014.

ORACLE CORPORATION.**Database Reference.** 2014. Disponível em: < https://docs.oracle.com/cd/E11882_01/server.112/e40402/toc.htm >. Acesso em: 11 out. 2014.

RED HAT. **Red Hat Enterprise Linux 5 Tuning and Optimizing Red Hat Enterprise Linux for Oracle 9i and 10g Databases.** 2008. Disponível em: < https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Tuning_and_Optimizing_Red_Hat_Enterprise_Linux_for_Oracle_9i_and_10g_Databases/ >. Acesso em: 07 out. 2014.

RED HAT. **Red Hat Enterprise Linux 6 Performance Tuning Guide.** 2011. Disponível em: <https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/Performance_Tuning_Guide/index.html>. Acesso em: 1 out. 2014.

RED HAT. **Oracle Database 11g Release 2 on Red Hat Enterprise Linux 6.** 2014. Disponível em: < https://access.redhat.com/sites/default/files/attachments/oracle_11gr2_on_rhel6_0.pdf >. Acesso em: 1 out. 2014.

WATSON, John. **OCA Oracle Database 11g Administração I.** Bookman. 2010.