UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA
CURSO DE TECNOLOGIA EM REDES DE COMPUTADORES

OTIMIZAÇÃO DE PERFORMANCE DE SISTEMAS *WEB* ATRAVÉS DE TÉCNICAS DE CLUSTERIZAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

Rafael Boufleuer

Santa Maria, RS, Brasil

2013

OTIMIZAÇÃO DE PERFORMANCE DE SISTEMAS *WEB* ATRAVÉS DE TÉCNICAS DE CLUSTERIZAÇÃO

Rafael Boufleuer

Trabalho de Conclusão de Curso (TCC) do Curso de Tecnologia em Redes de Computadores, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Tecnólogo em Redes de Computadores.

Orientador: Prof. Ms. Celio Trois

Santa Maria, RS, Brasil
2013

UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO TÉCNICO INDUSTRIAL DE SANTA MARIA
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES

A Comissão Examinadora, abaixo assinada, aprova o Trabalho de Conclusão de Curso

OTIMIZAÇÃO DE PERFORMANCE DE SISTEMAS *WEB* ATRAVÉS DE TÉCNICAS DE CLUSTERIZAÇÃO

elaborado por
Rafael Boufleuer

COMISSÃO EXAMINADORA

Celio Trois, Ms.
(presidente/orientador)

Walter Priesnitz Filho, Ms. (UFSM)

Tiago Antônio Rizzetti, Ms. (UFSM)

Santa Maria, 18 de Janeiro de 2013.

RESUMO

TRABALHO DE CONCLUSÃO DE CURSO
CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES
UNIVERSIDADE FEDERAL DE SANTA MARIA

OTIMIZAÇÃO DE PERFORMANCE DE SISTEMAS *WEB* ATRAVÉS DE TÉCNICAS DE CLUSTERIZAÇÃO

AUTOR: RAFAEL BOUFLEUER
ORIENTADOR: CELIO TROIS
Data e Local da Defesa: Santa Maria, 18 de Janeiro de 2013.

Este trabalho consiste na aplicação de técnicas de clusterização em sistemas W*eb* desenvolvidos na linguagem de programação Java. Os principais objetivos são: melhorar a performance destes sistemas, proporcionar uma estrutura de alta disponibilidade e também otimizar os recursos computacionais existentes. Para o desenvolvimento do presente trabalho, todos os estudos, implementações e testes serão efetuados no Sistema Integrado de Gestão Acadêmica da Educação - SIGA-EPCT.

Palavras-Chave: Cluster, Sistemas W*eb*, SIGA-EPTC, Glassfish.

ABSTRACT

COMPLETION OF COURSE WORK
SUPERIOR COURSE OF TECHNOLOGY IN COMPUTER NETWORKS
FEDERAL UNIVERSITY OF SANTA MARIA

OPTIMIZING PERFORMANCE OF *WEB* SYSTEMS USING CLUSTERING TECHNIQUES

AUTHOR: RAFAEL BOUFLEUER
ADVISER: CELIO TROIS
Defense Place and Date: Santa Maria, January 25th, 2013.

This work consists in the application of clustering techniques in W*eb* systems developed in Java programming language. The main objectives are: to improve the performance, to provide high-availability and also to optimize computational resources. To development of this work, all studies, implementations and tests will be made on a computer system called Sistema Integrado de Gestão Acadêmica da Educação – SIGA-EPTC.

Keywords: Clustering, *Web* Systems, SIGA-EPTC, Glassfish.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SIGA-EPCT	-	*Gestão Acadêmica da Educação*
API	-	*Application programming interface*
CC-NUMA -	-	*Cache-Coherent Nonuniform Memory Access*
CDUs	-	*Casos de Uso*
MPP	-	*Massively Parallel Processors*
DaaS	-	*Database as a Service*
DAS	-	*Domain Administration Server*
GMS	-	*Group Management Service*
HTTP	-	*Hyper Text Transfer Protocol*
IaaS	-	*Infrastructure as a Service*
LTS	-	*Long Term Support*
NTP	-	*Network Time Protocol*
PaaS	-	*Platform as a Service*
PCs	-	*Personal Computers*
SaaS	-	*Software as a Service*
SMP	-	*Symmetric Multiprocessors*
URL	-	*Uniform Resource Locator*

LISTA DE ANEXOS

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento da Internet, cada vez mais as organizações precisam resolver seus problemas de capacidade, disponibilidade e escalabilidade, necessitando de um maior poder computacional do que um único computador pode proporcionar. A simples conexão de recursos computacionais não garante alto desempenho para as aplicações. Com a utilização da clusterização para conectar múltiplos computadores e coordenar os esforços computacionais existentes, é possível superar estas limitações (Buyya, 1999, Gorino, 2006). De acordo com Tanembaum (2008) um cluster é definido como um grupo de computadores completos interconectados trabalhando juntos. Clusters são uma boa alternativa para multiprocessamento com escalabilidade incremental e alta disponibilidade, por um preço menor do que uma única máquina de grande porte.

Existe na literatura uma considerável confusão entre uma rede de computadores e um sistema paralelo ou distribuído. A principal diferença é que em um sistema distribuído, um conjunto de computadores independentes parece ser, para o usuário, um único sistema coerente. Já em uma rede de computadores, os diversos tipos de máquinas e sistemas operacionais são totalmente visíveis para os usuários (Tanenbaum, 2003). Segundo Buyya (1999) o uso de clusters para prototipagem, depuração e execução paralela de aplicações vem se tornando cada vez mais uma alternativa popular ao invés da utilização de plataformas de computação paralelas de alto custo. Um importante fator que fez com que a utilização dos clusters aumentasse foi a padronização de ferramentas e utilitários usadas para aplicações paralelas. Neste contexto, a padronização possibilitou que as aplicações fossem desenvolvidas, testadas e executadas com mais rapidez.

A performance e a escalabilidade dos clusters tem sido muito estudadas na literatura. Porém, entender o comportamento da alta disponibilidade durante a falha de componentes ou softwares, e a relação entre performance e alta disponibilidade tem recebido menos atenção. Portanto, percebe-se a necessidade de se ter sistemas que necessitam de alta disponibilidade e desempenho, mas que acabam trazendo um alto custo em virtude das redundâncias e técnicas necessárias. Para tornar isso mais acessível, além do uso da virtualização, serão utilizados Softwares Livres (Open Source) na implementação do cluster. Isso garante uma economia

considerável, sem perder a confiabilidade e a disponibilidade do sistema (Reis; Nagaraja, 2003).

Este trabalho consiste na aplicação de técnicas de clusterização em sistemas W*eb* desenvolvidos na linguagem de programação Java. Os principais objetivos deste trabalho são: melhorar a performance destes sistemas, proporcionar uma estrutura de alta disponibilidade e também otimizar os recursos computacionais existentes.

Para o desenvolvimento do presente trabalho, todos os estudos, implementações e testes serão efetuados utilizando o Sistema Integrado de Gestão Acadêmica da Educação - SIGA-EPCT, que será disponibilizado pelo cluster através do Servidor de Aplicação Glassfish.

O presente trabalho está organizado em capítulos. No Capítulo 2 será realizada uma revisão bibliográfica, onde serão abordadas as motivações para a utilização da computação paralela e suas principais arquiteturas. Além disso, serão apresentadas as estruturas de clusters, aplicabilidades, classificações, vantagens, características de clusters de computadores e também trabalhos relacionados.

O Capítulo 3 traz os materiais e métodos utilizados no presente trabalho. Neste capítulo será apresentado o sistema disponibilizado no cluster que é o SIGA-EPCT – Sistema Integrado de Gestão Acadêmica, o Servidor de Aplicação Glassfish e o Oracle iPlanet *Web* Server. Além disso, será demonstrada a implementação do cluster e os equipamentos utilizados para execução do trabalho. O Capítulo 4 está destinado aos resultados obtidos no trabalho, apresentando os testes realizados e observações que devem ser levadas em consideração quando for realizada a clusterização de sistemas *Web*. O Capítulo 5 tece as considerações finais.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo irá destacar uma forma de visualizar a computação através de duas eras, abordando quais as motivações para a utilização da computação paralela e suas principais arquiteturas.

De acordo com Buyya (1999) a indústria computacional é uma das que mais crescem, alimentada pelo rápido desenvolvimento de tecnologias nas áreas de software e hardware. Pode-se destacar a computação em duas eras:

• Era da computação seqüencial;

• Era da computação paralela.

Cada era começou com o desenvolvimento de arquiteturas de hardware seguido pelo software, onde as tecnologias para desenvolvimento de sistemas computacionais na era seqüencial evoluíram com o passar dos anos. O mesmo processo ainda está acontecendo na era da computação paralela. Mas a maior razão da criação e utilização da computação paralela, é que o paralelismo é uma das melhores maneiras de superar os gargalos de um único processador. Além disso, a razão do custo/performance de um pequeno cluster em oposição a um único computador com o mesmo poder computacional é consideravelmente menor.

2.1 Arquiteturas de computação paralela

Com o passar dos anos muitos sistemas de computadores diferentes com suporte para alta performance surgiram. De acordo com Buyya (1999), suas classificações são baseadas em como seus processadores, memória e interconexão são dispostos. Os sistemas mais comuns são:

• Massively Parallel Processors (MPP): Um MPP geralmente é um grande sistema paralelo com uma arquitetura onde, por exemplo, nenhum nó compartilha memória ou disco e possui uma memória principal e um ou mais processadores, onde cada nó executa uma cópia separada do sistema operacional. Normalmente consiste de centenas de nós, que são conectados por uma rede de interconexão de alta velocidade.

• Symmetric Multiprocessors (SMP): Os sistemas SMP normalmente possuem de 2 a 64 processadores e podem ser considerados uma arquitetura de compartilhamento de recursos. Neste sistema, todos os processadores compartilham todos os recursos disponíveis (memória, dispositivos de entrada e saída entre outros), onde uma única cópia do sistema operacional é executada neste sistema.

• Cache-Coherent Nonuniform Memory Access (CC-NUMA): O CC-NUMA é um sistema escalável multiprocessador tendo uma arquitetura de acesso coerente não uniforme a memória cache. Esse tipo de sistema tem este nome (NUMA) pelos tempos não uniformes de acesso a mais próxima e a mais remota parte da memória.

• Distributed Systems: Sistemas distribuídos podem ser considerados uma rede convencional de computadores independentes. Possui uma imagem múltipla do sistema, onde cada nó executa o seu próprio sistema operacional, e as máquinas individuais de um sistema distribuído poderiam ser, por exemplo, uma combinação de MPPs, SMPs, clusters e computadores individuais.

• Clusters: Cluster é uma coleção de workstations ou PCs que estão conectados por alguma tecnologia de rede. Para fins de computação paralela, um cluster geralmente consiste em PCs interconectados por uma rede de alta velocidade para proporcionar alta performance. Um cluster funciona como uma coleção integrada de recursos e pode ter uma única imagem do sistema abrangendo todos os nós.

• Grid Computing: De acordo com Foster (2002), grid é uma tecnologia parecida com a tecnologia de cluster. A principal diferença é que uma grid suporta ambientes mais heterogêneos do que clusters e também conecta computadores que não confiam totalmente uns nos outros. A grid é otimizada para cargas de trabalho que consistem de muitas tarefas independentes entre si e que portanto não precisam se comunicar durante o processamento. Grids administram distribuição de tarefas para computadores que trabalharão independentemente. Alguns recursos como os de armazenamento podem ser compartilhados,

mas resultados intermediários de uma tarefa em um nó não alteram a computação de tarefas em outros nós.

• Cloud: Cloud computing é um tipo de sistema paralelo ou distribuído que consiste em uma coleção de computadores interconectados e virtualizados para disponibilização de serviços. Entre os tipos de serviços oferecidos, pode-se citar quatro tipos (BARBOSA, 2009):

• IaaS (Infrastructure as a Service): disponibilização de recursos de hardware, como espaço em disco e capacidade de processamento. Um exemplo é o Amazon S3;

• DaaS (Database as a Service): um tipo especializado de armazenamento que envolve serviços de Banco de Dados. Exemplos: Amazon SimpleDB, Google Big Table.

• PaaS (Platform as a Service): provê serviços para facilitar o desenvolvimento e distribuição de aplicações. Exemplos: Google AppEngine, Microsoft Azure Services platform.

• SaaS (Software as a Service): provê aplicações inteiras que, ao invés de serem utilizadas por um mecanismo de aquisição de licenças e download de software, são acessadas diretamente através da Internet. Exemplos: Aplicações de escritório (processadores de texto e planilhas) do Google. As diferenças entre cloud e grid podem ser expressas da seguinte forma:

Distribuição de recursos: Cloud computing é um modelo centralizado enquanto que grid um modelo descentralizado onde a computação pode ocorrer em vários domínios administrativos.

Propriedade: Grid é uma coleção de computadores conectados compostos por múltiplas partes em múltiplos lugares, onde os usuários podem compartilhar os recursos (CLOUD, CLUSTER, GRID, 2013).

As tecnologias desenvolvidas para as Grids evoluíram e vem sendo utilizadas nos meios acadêmicos, para oferecer grandes capacidades de processamento e de armazenamento, utilizando técnicas de processamento paralelo, de forma a facilitar a realização de pesquisas científicas. Já o foco em gerar economia de escala e oferecer serviços para clientes externos, pode ser percebido com força nos ambientes Cloud que são disponibilizados geralmente por grandes empresas (BARBOSA, 2009).

O presente trabalho utilizará a arquitetura de cluster. Nas seções 3.4 e 3.5 será explicado o motivo pelo qual optou-se pela utilização de cluster ao invés de grid ou cloud. Em virtude

desta escolha, a arquitetura de cluster será explanada com maiores detalhes nas próximas subseções.

2.2 Cluster

Um cluster é um tipo de sistema de processamento paralelo ou distribuído, que consiste em uma coleção de computadores interconectados trabalhando juntos como se fossem uma única fonte de computação integrada (Buyya, 1999). Segundo Stallings (2010) o termo computador completo significa um sistema que pode funcionar por si só, à parte do cluster. Na literatura, cada computador em um cluster normalmente é chamado de nó. Este arranjo de nós fornece alto desempenho, disponibilidade e escalabilidade incremental por um custo menor. É desejável que haja a capacidade de tolerância a falhas, o que significa que se um computador falhar ao executar uma aplicação, outro computador no cluster pode assumir e completar a aplicação. Porém, é necessário algum software de gerenciamento para atribuir as requisições vindas dos clientes aos servidores, para que a carga seja balanceada e a alta utilização, alcançada (Stallings, 2010).

A utilização de clusters permite as organizações aumentarem seu poder de processamento usando tecnologias padronizadas (componentes de hardware e software comuns), que podem ser adquiridos por um relativo baixo custo. Isto provê, uma maior capacidade de expansão para as organizações, podendo aumentar consideravelmente sua capacidade de processamento enquanto preservam seus investimentos existentes sem a necessidade de muitas despesas extras (Buyya, 1999). A lista abaixo, contém algumas das razões que motivaram a crescente utilização dos clusters e algumas vantagens de utilização (Júnior et al., 2005, Buyya, 1999):

• Alto Desempenho - possibilidade de se resolver problemas complexos através do processamento paralelo, diminuindo o tempo de resolução do mesmo;

• Escalabilidade - possibilidade de que novos componentes sejam adicionados à medida que cresce a carga de trabalho. O único limite é a capacidade da rede;

• Tolerância a Falhas - o aumento de confiabilidade do sistema como um todo, caso alguma parte falhe;

• Baixo Custo – pode-se obter redução de custos e processamento de alto desempenho utilizando-se simples PCs;

• Independência de fornecedores - utilização de hardware aberto, software de uso livre e independência de fabricantes e licença de uso.

• Computadores individuais estão se tornando mais poderosos. Isto é, a performance dos computadores vem crescendo muito nos últimos anos, e provavelmente continuará crescendo, com processadores mais rápidos e mais eficientes.

• A banda de comunicação entre computadores está aumentando pois novas tecnologias estão sendo implementadas e difundidas.

• Computadores utilizados em clusters são baratos e prontamente disponíveis para disponibilizar alta performance.

• Clusters podem ser facilmente melhorados. A capacidade dos nós pode ser incrementada acrescentando memória ou processadores.

2.2.1 Aplicabilidade e classificação dos Clusters

O uso de clusters vem sendo explorado nas mais diferentes comunidades devido a tal solução oferecer um equilíbrio entre o desempenho desejado e o custo do sistema. Os clusters não são utilizados somente em sistemas em que se busca alto desempenho. Sua aplicabilidade se estende a outras áreas, podendo ser dividida em três categorias básicas quanto à funcionalidade: alta disponibilidade, balanceamento de carga e computação de alto desempenho (BAKER, 2000).

Cluster de Alta Disponibilidade

Segundo Buyya (1999), os serviços oferecidos devem possuir uma alta disponibilidade o tempo todo. A qualquer momento, um ponto de falha deve ser recuperável sem que o funcionamento da aplicação do usuário seja afetada, e a utilização do sistema deve ser migrada para um outro nó ou instância transparentemente. Isso pode ser alcançado através da utilização de pontos de verificações e tecnologias de tolerância a falhas (espelhamento, failover) para a recuperação do ponto de falha. São estas paradas não planejadas, que devem ser evitadas, que influenciam diretamente na qualidade do serviço e nos prejuízos financeiros das empresas (JÚNIOR et al., 2005).

A alta disponibilidade auxilia na continuidade da operação dos sistemas em serviços de rede, armazenamento de dados ou processamento, mesmo se houver falhas em um ou mais dispositivos, sejam eles hardware ou software. Nos clusters de alta disponibilidade, os equipamentos são usados em conjunto para manter um serviço ou equipamento sempre ativo, replicando serviços e servidores, evitando máquinas paradas e ociosas (Júnior et al., 2005).

A disponibilidade é, em muitos casos, mais importante que escalabilidade, pois serviços on-line irão se diferenciar pela disponibilidade dos serviços oferecidos. Alta disponibilidade também influencia na manutenção dos sistemas. Na internet, os serviços precisam estar disponíveis 24 horas por dia, e tendo o serviço indisponível, mesmo que durante algumas horas a noite, pode causar perdas significativas para seus usuários (Buyya, 1999).

Cluster para Balanceamento de Carga

Quando um serviço é requisitado por muitos usuários e existem diversas máquinas responsáveis por responder pelo mesmo serviço, este tipo de cluster é utilizado. Os servidores são capazes de realizar uma redistribuição da carga de tarefas, balanceando a utilização de

todos os servidores. Se um nó falhar, as requisições são redistribuídas entre os nós disponíveis.

Um cluster com balanceamento de carga possui uma alta escalabilidade e pode ser facilmente expandido com a inclusão de novos nós (Gorino, 2006). Além disso, a distribuição da carga para um determinado nó, pode ser programada estaticamente ou pode mudar dinamicamente dependendo do estado atual da rede (Buyya, 1999).

De acordo com Buyya (1999), balanceamento de carga em redes é a distribuição de carga ou tráfego da rede entre os nós do cluster. A decisão de distribuição da carga para um determinado nó, pode ser programada estaticamente ou pode mudar dinamicamente dependendo do estado atual da rede. Os nós podem estar conectados entre si dentro do cluster, mas mais importante que isso, devem estar diretamente ou indiretamente conectados com o dispositivo responsável pelo balanceamento e distribuição de carga. Através do balanceamento de carga, busca-se aumentar a disponibilidade do sistema, que se refere a porcentagem de tempo em que um sistema está disponível para uso efetivo (Hwang, 1998).

Cluster de Alta Performance

Este tipo de cluster tem como foco o alto desempenho. Um cluster de computadores pode ser visto como uma solução alternativa para universidades e empresas de pequeno e médio porte, para obterem processamento de alto desempenho na resolução de problemas através de aplicações paralelizáveis. Além disso, o custo é razoavelmente baixo se comparado com os altos valores necessários para a aquisição de um supercomputador na mesma classe de processamento (Júnior et al., 2005).

Quando se fala em processamento de alto desempenho, que pode ser traduzido em processamento paralelo e processamento distribuído, pensa-se em grandes máquinas dedicadas (supercomputadores) com um alto custo. Porém, devido aos clusters os custos foram reduzidos e existem máquinas rápidas que viabilizam o uso do processamento de alto desempenho na solução de problemas em diversas áreas (Júnior et al., 2005).

2.3 Aplicações *Web*

Pode-se definir uma aplicação *Web* como uma aplicação de software que utiliza a *Web* como ambiente de execução (WINCKLER, 2002). Aplicações *Web* envolvem Sites *Web* ou sistemas *Web* que são acessados e visualizados através de um software chamado *browser*, instalado no computador cliente, utilizando-se da infra-estrutura da internet, através do protocolo HTTP (Hyper Text Transfer Protocol). Aplicações *Web* podem ser disponibilizadas por servidores *Web* possibilitando o acesso rápido ao usuários do sistema. O sistema *Web* utilizado no presente trabalho é o SIGA-EPCT - Sistema Integrado de Gestão Acadêmica da Educação que é disponibilizado pelo Servidor de Aplicação Glassfish.

2.4 Servidores de Aplicação

Application Servers, ou servidores de aplicação, são softwares que fornecem a infra-estrutura de serviços para a execução de aplicações distribuídas. Os servidores de aplicação são executados em servidores e são acessados pelos clientes através de uma conexão de rede (APPLICATION SERVERS, 2013) . Os principais objetivos são a centralização da Informação, Banco de Dados, Controles de Acessos, visando Organização, Centralização, Integridade, Segurança e Disponibilidade da informação conseqüentemente obtendo Aumento de Produtividade (SERVIDOR DE APLICAÇÃO, 2013). Pode-se citar algumas características dos servidores *Web*:

- Tolerância a falhas: através de políticas para recuperação e distribuição de componentes em clones dos servidores.
- Balanceamento de carga: com a análise da carga nos servidores permite a distribuição da carga de forma a maximizar a utilização dos recursos disponíveis.
- Console de gerenciamento: permite o gerenciamento de vários servidores de aplicação através de um único sistema gráfico.

O GlassFish Enterprise Server é um servidor de aplicação utilizado no presente trabalho, e será apresentado na sessão 3.2.

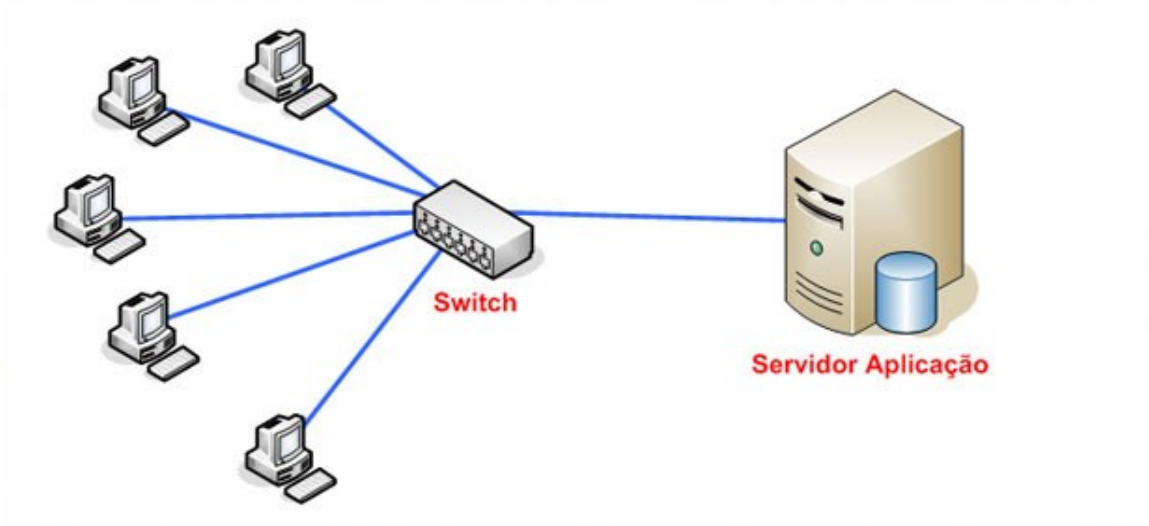

Figura 1 – Servidor de Aplicação

2.5 Ferramentas para Automação de Testes

A utilização de um conjunto de ferramentas para realização e análise de testes, facilita a execução do conjunto de testes e a leitura dos resultados obtidos. O Selenium (BRUNS, 2009) é um software de código aberto que permite executar os testes diretamente nos navegadores mais populares (Firefox, Internet Explorer, Opera, Safari), o que torna possível testar a compatibilidade de um sistema *Web* às diferentes plataformas operacionais. O Selenium permite a captura das ações realizadas pelo usuário no navegador Web, onde cada ação que é realizada manualmente na página é registrada, e expressa através de uma API própria do software. Desta forma, permitindo que o código gerado possa ser exportado para uma das linguagens de programação suportadas pelo Selenium.

Na categoria de ferramentas de testes funcionais, onde foram avaliadas as ferramentas Selenium, Badboy e Canoo (VELOSO et al., 2011), a que demonstrou ser a mais adequada foi

aferramenta Selenium, principalmente nos quesitos de gerador manual de testes (mecanismo de captura-reprodução, uso de linguagens de alto nível, etc.), arquitetura da ferramenta (uso de tecnologia Web, etc.) e executor de teste (executor de teste com possibilidade de pausa e retomada de execução, agrupador e escalonador de testes, etc).

Na figura 3 pode-se observar a ferramenta Selenium, onde é especificado o endereço referente a máquina principal do cluster contendo o sistema SIGA-EPTC. Na parte central da figura, observa-se parte do código do teste que foi executado, responsável por fazer a autenticação no sistema e realizar o teste automaticamente com a necessidade apenas de um click.

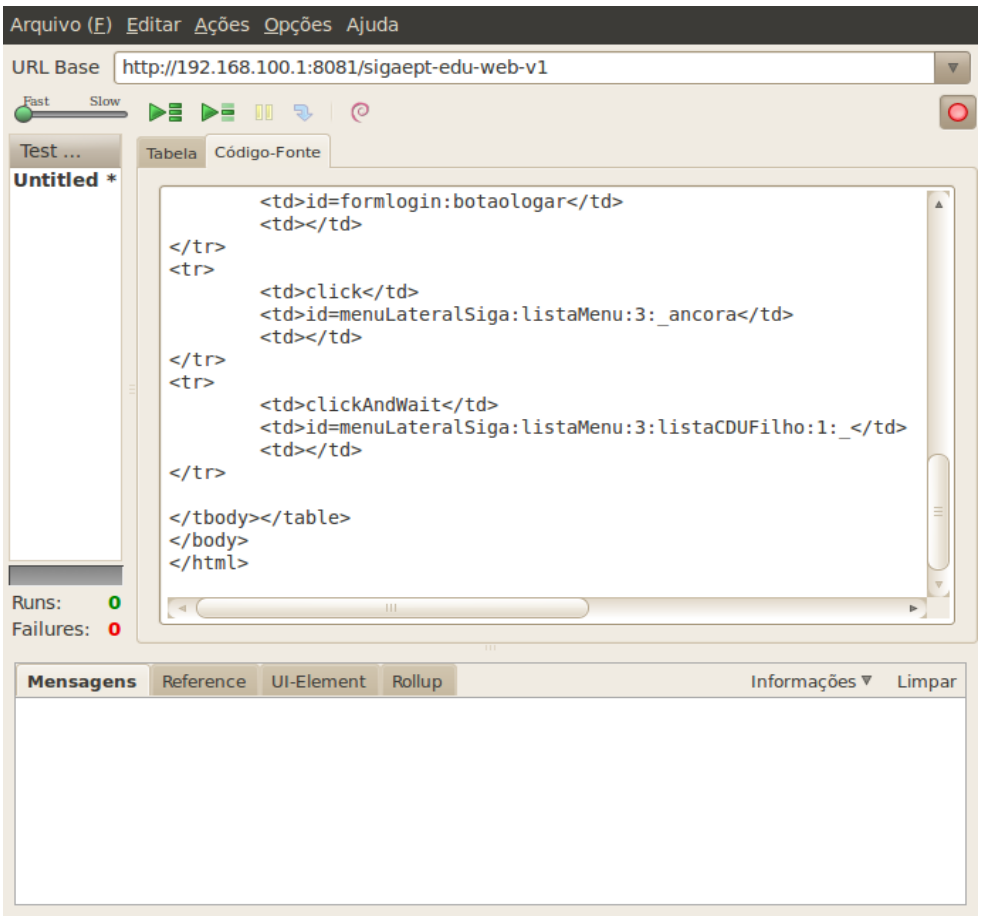

Figura 2 – Ferramenta Selenium

2.6 Trabalhos Relacionados a Cluster de Computadores

Devido ao elevado potencial de processamento paralelo e distribuído que os clusters oferecem, existem estudos realizados utilizando técnicas, ferramentas e experimentos sobre o assunto. Também foram foram analisados outros trabalhos que usam o cluster como balanceamento de carga.

Em seu trabalho, Aversa propôs e avaliou a implementação de um protótipo de servidor *Web* distribuído, obtendo resultados positivos quanto a escalabilidade e inclusive custo, quando da aplicação de sua técnica em sistemas de pequeno porte. Aversa sugere que os clusters de servidores *Web* podem tirar grande proveito do balanceamento de carga (AVERSA & BESTAVROS, 2000). O estudo de Meredith (MEREDITH et al., 2003) realizou uma comparação entre diversas ferramentas de configuração e manutenção de cluster.

Nguyen e Peirre (NGUYEN & PIERRE, 2001) também criaram um modelo analítico e o experimentaram em um simulador de um sistema de arquivos paralelos, com o intuito de estudar a escalabilidade dos clusters. Como resultado, mostrou que o crescimento de um cluster depende do grau de saturação da rede de comunicação. Griebler em seu trabalho realizou uma avaliação de diversas ferramentas de gerenciamento de clusters, obtendo uma comparação entre suas diversas características, mostrando como as ferramentas de gerenciamento de cluster têm um papel fundamental na correta distribuição dos recursos de um cluster de computadores (Griebler, 2008).

3 MATERIAIS E MÉTODOS

O cluster do presente trabalho foi desenvolvido para analisar as vantagens da utilização desta estrutura no sistema SIGA-EPCT avaliando as questões de desempenho, balanceamento de carga, disponibilidade e tolerância a falhas. Embora sua motivação principal tenha sido o escopo do projeto SIGA-EPCT, esta estrutura pode ser utilizada para aplicações similares implementadas na linguagem Java e que usem o servidor de aplicação Glassfish, assim como o sistema estudado no presente trabalho.

3.1 SIGA-EPCT – Sistema Integrado de Gestão Acadêmica da Educação

O SIGA-EPCT - Sistema Integrado de Gestão Acadêmica da Educação é um sistema integrado de gestão acadêmica que está sendo desenvolvido com tecnologias livres e de forma colaborativa pelas próprias instituições participantes da Rede de Educação Profissional, Científica e Tecnológica. O SIGA-EPCT, gerencia os processos acadêmicos das instituições de EPCT - Educação Profissional, Científica e Tecnológica, englobando os módulos de ensino, pesquisa e extensão (RENAPI, 2012).

A Figura 2 demonstra a arquitetura de um sistema *Web* disponibilizado sem a implementação de um cluster, e de um sistema *Web* clusterizado. No sistema não clusterizado, todos os usuários acessam o sistema *Web* que está localizado em apenas uma máquina, responsável por tratar todas as requisições. Por outro lado, no sistema clusterizado, através do balanceamento de carga, ocorre a distribuição das requisições entre os diversos nós e instâncias do cluster, possibilitando alta disponibilidade e otimização de performance do sistema *Web*. Na Figura 3, pode-se observar a tela de gerenciamento da funcionalidade de Diário de classe no sistema.

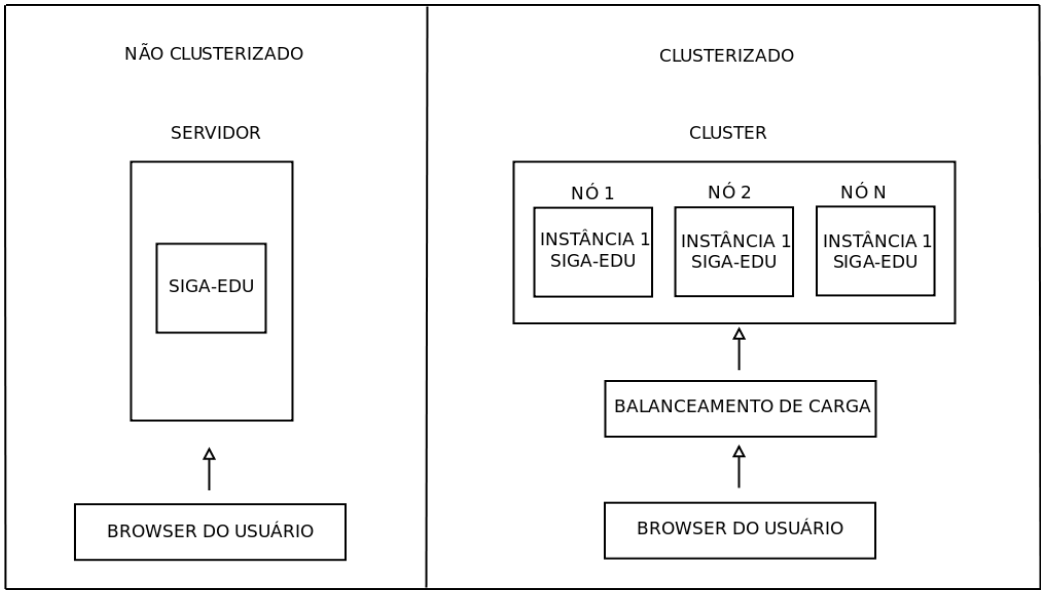

Figura 3 – Arquitetura de sistema *Web* não clusterizado e clusterizado

Figura 4 – Funcionalidade Diário de Classe

3.2 Servidor de Aplicação Glassfish

O GlassFish Enterprise Server é um servidor de aplicação baseado na plataforma Java, provendo recursos de clusterização, alto desempenho, alta disponibilidade e escalabilidade (ORACLE, 2012). Além disso, possui diversas funcionalidades como deploy de aplicações, criação e configuração de domains e instâncias de servidores, controle de instâncias de servidores e gerenciamento de clusters. Também possibilita o monitoramento e gerenciamento de performance através de uma central de administração (ORACLE ADMINISTRATION GUIDE, 2012). O GlassFish Enterprise Server 2.1 é o servidor de aplicação utilizado pelo SIGA-EPCT, porém na implementação do cluster foi utilizada a versão 2.1.1 em virtude de que esta versão apresenta correção de *bugs* com relação a clusterização.

3.3 Oracle iPlanet *Web* Server 7

O Servidor de Administração iPlanet possui recursos de administração que permitem o gerenciamento e monitoramento de Servidores *Web*. Pode ser gerenciado pela interface gráfica ou por linha de comando, sendo utilizado em conjunto com o Glassfish para implementação do cluster (IPLANET, 2012). Ele é o responsável pelo gerenciamento dos nós e instâncias criados no Glassfish e também pela distribuição das requisições e do balanceamento de carga para o cluster através da utilização do *plugin GlassFish Loadbalancer Configurator* 3.1.

3.4 Implementação do Cluster

Na implementação do cluster do presente trabalho, todos os nós são ativos possuindo o mesmo poder computacional e recebendo o mesmo número de requisições. Isso é definido nos arquivos de configuração do iPlanet, onde o tráfego gerado pelos usuários é redirecionado para as instâncias através do *plugin* de balanceamento de carga, que está instalado no Servidor iPlanet localizado no nó principal (Buyya, 1999).

Para a implementação do cluster, basicamente foram utilizados 2 servidores Dell PowerEdge R710 com processadores Intel Xeon quad-core, 12 GB de memória RAM, onde foram configuradas as máquinas virtuais que implementam o cluster. Cada máquina virtual possui 1 processador e 3 GB de memória RAM, seguindo as especificações sugeridas pelo servidor de aplicação Glassfish. Em um dos servidores foram criados o nó principal do cluster e a máquina responsável pela disponibilização do sistema de forma *standalone*. A segunda máquina será utilizada para comparar os resultados de desempenho obtidos com e sem a clusterização. No outro servidor foram criados mais dois nós escravos que farão parte do cluster.

Todas as máquinas virtuais foram configuradas no VirtualBox 4.1.16, utilizando-se o sistema operacional Ubuntu 10.04 LTS. A escolha desta versão de sistema operacional deve-se ao fato de que o sistema SIGA-EPCT é homologado para tal. A interligação dos componentes da arquitetura do cluster pode ser observada na Figura 5. O nó principal, também chamado de DAS (*Domain Administration Server*), possui a listagem de todos os nós e instâncias do cluster que podem ser visualizados tanto por interface gráfica quanto pela linha de comando do Linux. É possível também obter informações de status de cada instância ou nó, iniciar ou parar os mesmos, realizar deploy de aplicações entre outras funcionalidades. Na máquina principal foi configurado o PostgreSQL 8.3 contendo a base de dados única do cluster, que será acessada por todos os nós e instâncias. Além disso, possui também o Servidor de Administração iPlanet e o *plugin* de balanceamento instalados. Este *plugin* do Glassfish realiza a comunicação entre o iPlanet e o Glassfish, e é utilizado para distribuição das requisições entre as instâncias do cluster, analisando o arquivo de configuração

loadbalancer.xml, que contém entre outras informações, a proporção de requisições que serão balanceadas para cada instância.

O *plugin* de balanceamento de carga suporta os métodos de *Cookie* e URL Rewrinting para gerenciamento de sessões (Administration Guide, 2012). No método utilizando *Cookie*, que é utilizado no presente trabalho, o *plugin* utiliza um cookie específico para guardar as informações de cada usuário. O navegador do usuário precisa suportar cookies para que este método funcione. Se o navegador não suportar esta opção, o método URL Rewriting é usado. Neste método, as informações são enviadas junto à URL, funcionando até em casos onde o navegador do usuário não suporta cookies. Para que este segundo método funcione, a aplicação precisa conter algumas características que garantam que cada URL contenha as informações da sessão.

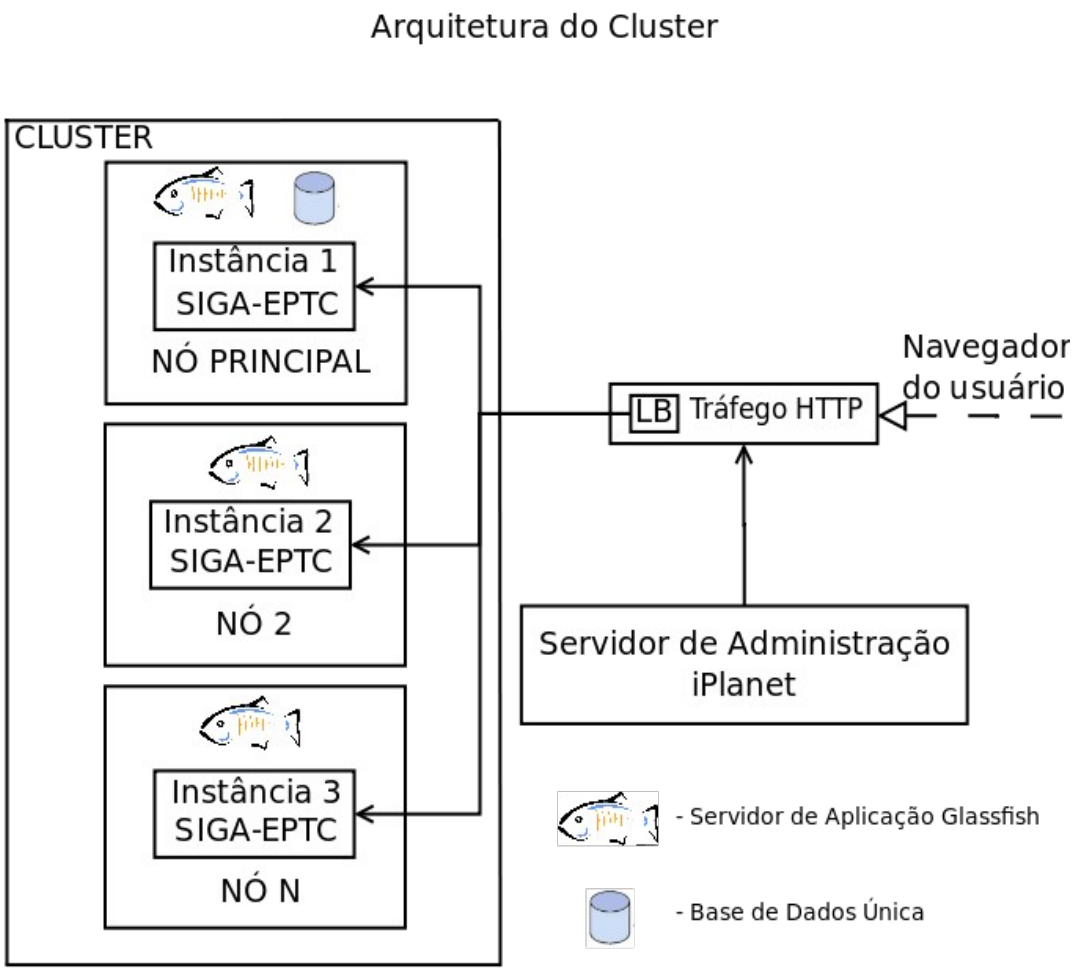

Figura 5 – Interligação entre os componentes do cluster

Os nós escravos do cluster apenas contém o Glassfish instalado, e todos assim como o nó principal, possuem a mesma aplicação e apontam pra mesma base de dados localizada no nó principal. Também é importante garantir que quando um nó ou instância falhar e o usuário for redirecionado para outra instância, que o sistema continue acessível sem problema nenhum, pois esta transição deve ser transparente para o usuário.

Quando uma instância ou mesmo um nó falhar, automaticamente o balanceador de carga redireciona as requisições destinadas ao componente que falhou para os outros componentes ativos no cluster, possibilitando a alta disponibilidade e evitando que o sistema fique indisponível por motivos de falhas tanto de *software* quando de *hardware*. Os benefícios da centralização do gerenciamento do cluster são significativas, pois tudo pode ser controlado em um único local pelo administrador. Por outro lado, é importante ressaltar que se o nó principal falhar, mesmo que os outros nós continuem ativos, o sistema ficará indisponível, pois é o nó central que gerencia toda a estrutura e possui a base de dados única utilizada pelo cluster. Na Figura 6 pode-se observar as instâncias do cluster e suas características.

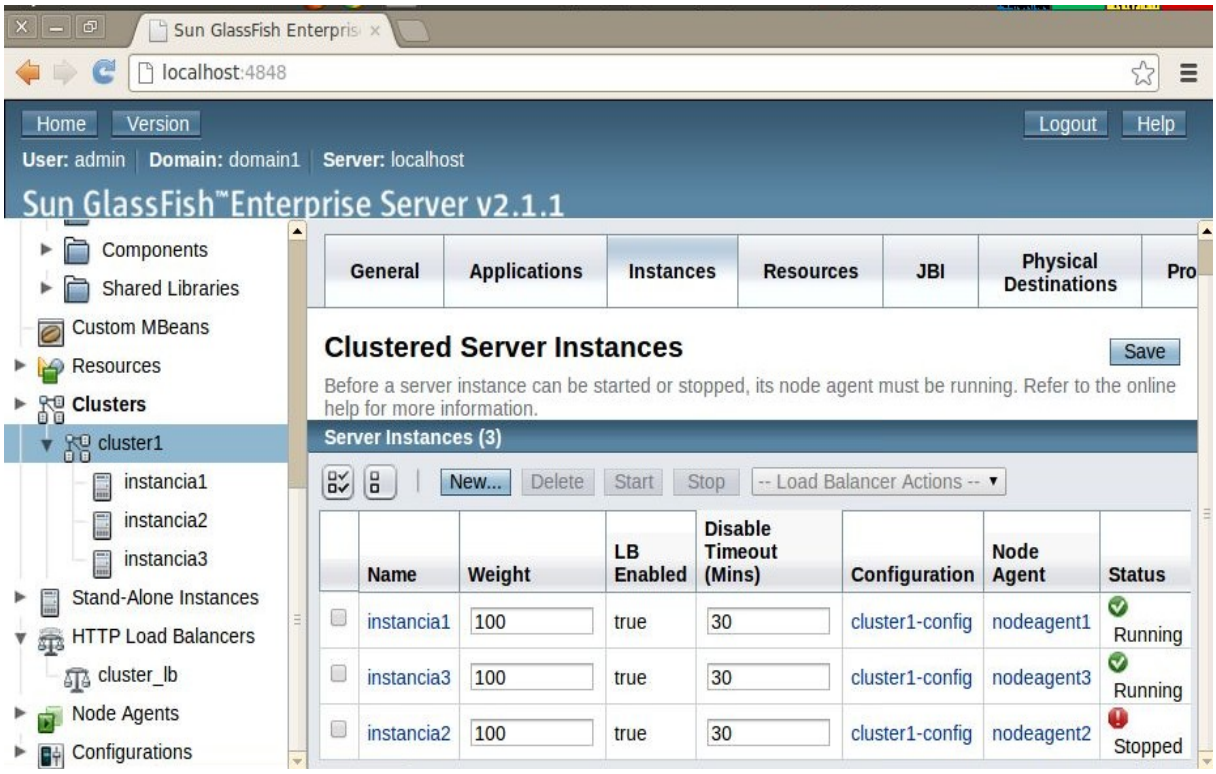

Figura 6 – Instâncias do cluster

O atributo peso das instância tem por objetivo fazer o balanceamento de carga entre as instâncias. Na Figura 6 pode-se observar que foi utilizado o valor 100 em todas as instâncias, significando que todas receberão o mesmo número de requisições. Se existirem instâncias com poderes computacionais diferentes, o atributo *weight* pode ser utilizado para balancear a carga, observando a capacidade de cada nó. Para entender melhor como funciona o atributo peso, podemos exemplificar, em um cluster com três instâncias, utilizando os valores 30, 30 e 40. Neste caso, a cada 100 requisições as duas primeiras instâncias receberiam 30 e a última receberia 40.

O campo *LBEnabled* habilita a instância a operar no balanceamento de carga e o campo *Disable Timeout* especifica o tempo, em segundos, que a instância pode ficar sem responder. Após este tempo a instância é desabilitada. Ainda como informações relevantes têm-se a qual nó a instância pertence e se esta instância está ativa ou não no cluster (ORACLE, 2012).

A Figura 7 demonstra os nós configurados no cluster, o *hostname* da máquina que possui o nó, o status de cada nó e quais instâncias estão paradas. Além disso, informa quais instâncias que precisam ser reinicializadas por alguma razão como por exemplo configurações alteradas no cluster.

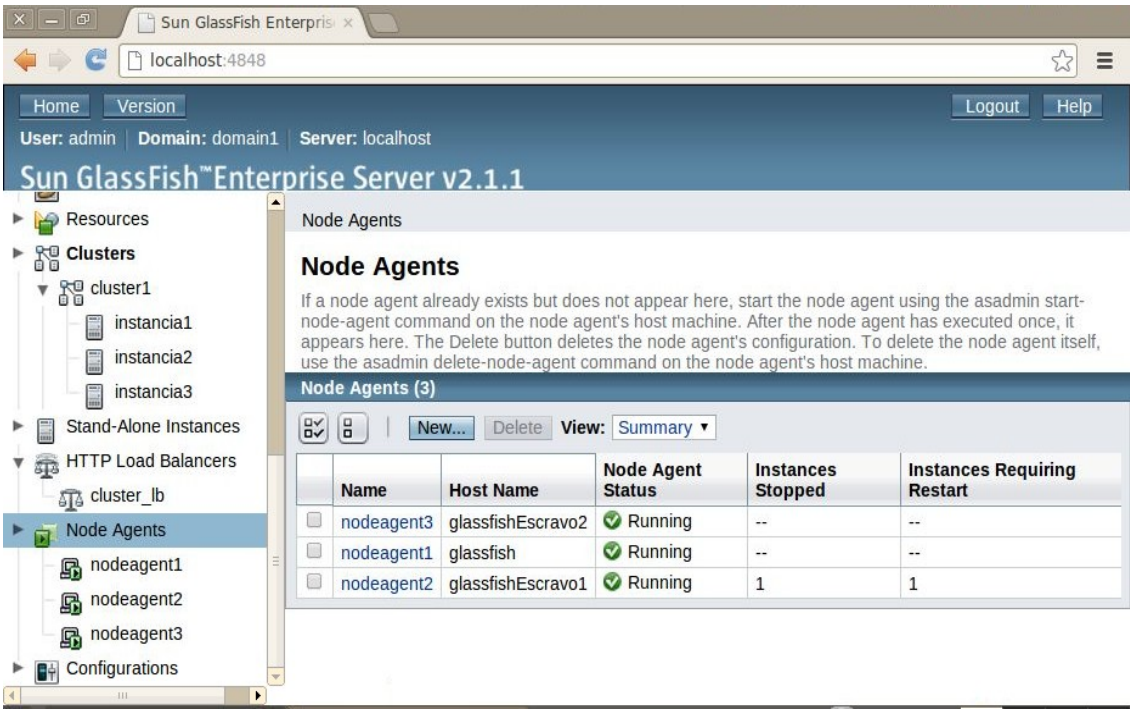

Figura 7 – Nós do cluster

3.5 Clusterização de Aplicações *Web*

Embora a motivação principal do presente trabalho tenha sido o escopo do projeto SIGA-EPTC, a estrutura do cluster e as modificações necessárias para a aplicação podem ser utilizadas para aplicações similares implementadas na linguagem Java e que utilizem o servidor de aplicação Glassfish.

No presente trabalho foram necessárias algumas alterações nas funcionalidades de login e autenticação do sistema. No processo de desenvolvimento e implementação da aplicação *Web* é interessante verificar quais modificações ou adequações são necessárias para que a aplicação funcione em um cluster. Para realização do presente trabalho utilizou-se a versão 2.1.1 do servidor de aplicação. É importante salientar que a versão do Glassfish homologada pelo SIGA-EPCT é a 2.1, porém esta versão apresenta *bugs* com relação a clusterização. Ressalta-se que uma boa alternativa para os sistemas *Web* é realizar a compatibilidade com o Glassfish 3.1, por apresentar funcionalidades adicionais, além de outros *bugs* corrigidos. Como este não era o escopo do trabalho, optou-se por utilizar a versão 2.1.1 que é totalmente compatível com a 2.1.

Para que uma sessão possa ser distribuída em diversas instâncias localizadas em diferentes hosts, alguns pontos devem ser levados em consideração (Administration Guide, 2012):

• O Glassfish suporta replicação de sessão em memória em outras máquinas do cluster para manter a sessão;

• Para garantir que a replicação de sessão em memória está funcionando corretamente, os hosts do cluster devem estar na mesma sub-rede;

• Os relógios de todos os hosts do cluster devem estar sincronizados;

• O uso de replicação de sessão em memória requer que o *Group Management Service* (GMS) esteja habilitado. Este é o componente responsável por avisar o cluster e o *Domain Administration Server* (DAS) que houve uma falha em alguma instância do cluster;

• Todas as instâncias devem ter a mesma aplicação *Web*. Além disso, é necessário que o arquivo web-xml contenha o elemento (*distributable*), habilitando a replicação de sessão para a aplicação;

• Criação ou alteração do arquivo sun-web.xml, com parâmetros que definam como a replicação de sessões ocorrerá no cluster.

• Todos os objetos que serão guardados na sessão devem ser serializáveis.

• A opção *availabilityenabled* precisa estar habilitada.

• Realizar a configuração dos hostnames no arquivo /etc/hosts em todas as máquinas do cluster.

• Para a utilização de uma base de dados única, é necessário habilitar o acesso remoto no PostgreSQL. Devem ser realizadas configurações que possibilitem que o PostgreSQL aceite conexões externas editando dois arquivos de configuração: pg_hba.conf e postgresql.conf.

3.6 Equipamentos e Cenário para a Execução dos Testes

A seção 3.4 descreve as características do *hardware* utilizado para implementação do cluster. Como citado anteriormente, os *softwares* utilizados foram: servidor de aplicação Glassfish, servidor *Web* iPlanet, gerenciador de máquinas virutais Virtual Box, *plugin* de balanceamento de carga , o Selenium e o *browser* Mozilla Firefox.

Nos servidores e nas máquinas virtuais realizou-se uma nova instalação do sistema operacional Linux Ubuntu 10.04 LTS 64 bits. Os servidores foram conectados a um servidor NTP (*Network Time Protocol*), para que seus relógios fossem sincronizados, isso se faz necessário para que a comunicação entre os nós do cluster seja realizada de forma correta.

A topologia utilizada para implementação do presente trabalho foi uma rede Fast Ethernet entre os computadores que testaram o sistema (cliente) e as máquinas virtuais com o sistema SIGA-EPCT. Além disso, os dois computadores físicos foram ligados por uma conexão direta Gigabit Ethernet. Para efetuar os testes de desempenho, utilizou-se computadores com processadores Intel Core i7, 8 GB de memória RAM, utilizando o navegador *Web* Mozilla Firefox para realização do acesso simultâneo aos sistemas disponibilizados. Na Figura 8, pode-se observar o cenário de conexão dos servidores físicos e das máquinas virtuais.

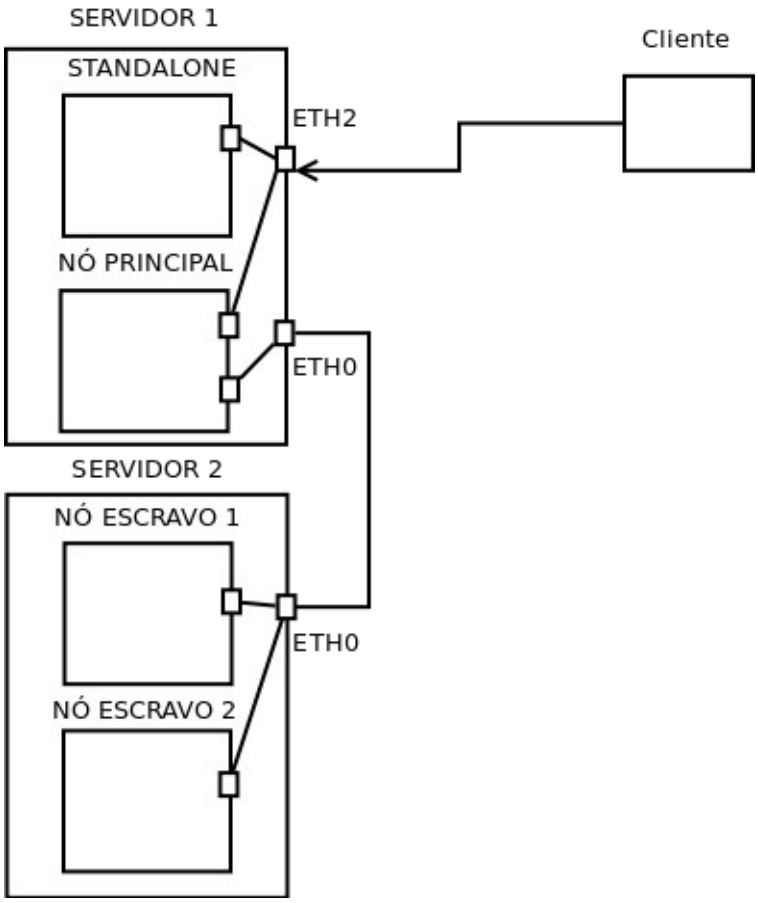

Figura 8 – Estrutura física do cluster

Para acessar o cluster os clientes acessam o sistema diretamente pela eth2 do servidor 1, onde está localizado o nó principal. Os dois computadores físicos foram ligados por uma conexão direta Gigabit Ethernet através da eth0 do servidor 1 e da eth0 do servidor 2. É através desta conexão que o nó principal faz a comunicação com os outros nós do cluster.

Para o acesso ao sistema na máquina *standalone*, os usuários também se conectam a eth2 do servidor 1.

3.7 Caso de Teste

As diversas funcionalidades do sistema são chamadas de Casos de Uso (CDU). O caso de uso que será testado no presente trabalho será o de Manter Diário de Classe. Optou-se por escolha funcionalidade pois o mesmo exige alto poder computacional.

Este caso de uso descreve os procedimentos para o registro da freqüência do aluno, o registro de aula em seus respectivos componentes curriculares, descreve também como manter a avaliação (notas/conceito) parciais dos alunos em seus respectivos componentes curriculares e status de encerramento do diário de classe, que representa a entrega do diário de classe pelo professor, seja ele impresso ou eletrônico. O Diário de Classe é o conjunto de informações que envolvem aluno, professor e classe em uma determinada aula (SIGAEPTC, 2013).

A ferramenta Selenium foi utilizada para criar um caso de teste no CDU previamente descrito. Foi feito uma busca de todas as informações referentes aos diários de classe do sistema, para isto utilizou-se uma base de dados populada com 875 diários de classe. Os resultados dos testes de desempenho realizados na máquina *standalone* e no cluster podem ser observados na seção 4.2

3.7 Modificações no SIGA-ECPT

Em um momento inicial tentou-se executar o sistema SIGA-EPCT sem efetuar nenhum tipo de alteração no código-fonte, porém, observou-se o seguinte erro:

• Cannot serialize session attribute ManterDiarioClasseMB for session

No erro acima citado, existe algum atributo ou atributos na classe que não estão serializados. Erros similares aconteciam quando alguma classe não estava serializada, necessitando de alterações nas classes e atributos não serializados. Além das alterações no código-fonte, também foram modificados alguns arquivos de configuração para que o sistema funcionasse no cluster, como por exemplo a criação do arquivo sun-web.xml, que contém

configurações para replicação de sessão em memória, serialização de classes e configurações no cluster criado no glassfish. Este arquivo está disponível como Anexo I.

Foi alterado também o arquivo web.xml, para tornar a aplicação compatível com a estrutura do cluster e distribuir as sessões em memória. Este arquivo está disponível como Anexo II.

4 RESULTADOS

Este capítulo irá analisar os resultados dos testes executados, trazendo uma discussão referente aos mesmos.

4.1 Testes de Disponibilidade

Para testar a disponibilidade do sistema, foram efetuados testes nos quais alguns nós dos clusters eram parados de forma normal, ou seja, através do console de finalização do servidor de aplicação e também de forma abrupta, através do comando “kill -9 java”. Em ambos os casos, quando um usuário estava logado em algum nó do cluster e este nó falhava, o usuário era redirecionado automaticamente para outro nó pelo *plugin* de balanceamento de carga configurado no servidor iPlanet. Observou-se que este processo é realizado de forma transparente para o usuário, que continua acessando as funcionalidades do sistema sem a necessidade de fazer o *login* e sem receber qualquer tipo de mensagem de erro.

Percebeu-se que no momento da falha em qualquer um dos nós, todos os outros nós do cluster são avisados e o *plugin* de balanceamento de carga direciona o usuário para outro nó ativo no cluster. Este nó utiliza-se do artifício de replicação de sessão em memória pra pegar o contexto que o usuário estava. Quando o nó volta a funcionar todos os nós do cluster são avisados novamente e, quando necessário este já será novamente alocado para disponibilização do sistema.

4.2 Testes de Performance

Conforme explicado na seção 3.7, os testes foram realizados no CDU Manter Diário de Classe em virtude de ser um dos módulos que exigem mais poder computacional no sistema testado. Para realização destes testes foram realizadas requisições simultâneas, provindas de vários computadores diferentes. Foram observados os tempos de resposta das requisições em ambos cenários: sistema *standalone* e sistema clusterizado. Foram escolhidos os valores de 1, 2, 5 e 10 acessos simultâneos, pois por padrão o Glassfish trata 5 conexões simultâneas.

Após a execução do mesmo teste no sistema clusterizado, percebeu-se uma diferença de tempo, sendo que, ao contrário do esperado, o tempo de execução foi maior que o sistema disponibilizado de forma *standalone*. O Quadro 1 demonstra a comparação de tempo do sistema clusterizado e não clusterizado. Na primeira coluna tem-se o número de acessos simultâneos. Nas segunda e terceira colunas são apresentados os tempos médios medidos em segundos para o sistema disponibilizado de forma *standalone* e clusterizado, respectivamente.

Acessos simultâneos	Sistema *standalone*	Sistema clusterizado
1	35,4	49,6
2	51,1	71,4
5	128,5	163,5
10	528	638,8

Quadro 1 – Tempo de resposta do sistema versus número de acessos simultâneos

A empresa desenvolvedora do servidor Glassfish disponibiliza um guia para otimização de performance para aplicações clusterizadas (ORACLE TUNING YOU APPLICATION, 2013). Após uma análise do código fonte do sistema, percebeu-se que várias sugestões de otimização indicados no guia não são implementados no código, como por exemplo a utilização de StringBuffers, atribuição de *null* à variáveis não mais utilizadas, argumentos não modificados nos métodos não são declarados como *final*,

passagem de parâmetros por referência, dentre outros. Em virtude da complexidade e do tamanho do sistema, não houve tempo hábil para implementar as modificações necessárias no código-fonte para que o mesmo ficasse otimizado para ambiente clusterizado.

No Quadro 2 demonstra-se o total de dados transferidos durante os acessos ao cluster. Na primeira coluna tem-se o número de acessos simultâneos e na segunda coluna são apresentados os valores medidos em *mega bytes*.

Acessos simultâneos	Tráfego gerado
1	52,1
2	112,5
5	285,3
10	545,75

Quadro 2 – Tráfego gerado versus número de acessos simultâneos

Pode-se observar que houve uma proporção na quantidade de informação trafegada na rede do cluster em relação ao número de acessos simultâneos.

5 CONSIDERAÇÕES FINAIS

Cada vez mais a utilização de clusters permite o aumento do poder de processamento para disponibilização de serviços e sistemas. Além disso, são uma boa alternativa para multiprocessamento com escalabilidade incremental e alta disponibilidade por um preço relativamente baixo, com a utilização da virtualização e softwares livres na sua implementação.

Este trabalho apresentou a aplicação de técnicas de clusterização em sistemas *Web* desenvolvidos na linguagem de programação Java. Os objetivos buscados pelo trabalho foram: melhorar a performance destes sistemas, proporcionar uma estrutura de alta disponibilidade e também otimizar os recursos computacionais existentes. Todos os estudos, implementações e testes foram efetuados utilizando o Sistema Integrado de Gestão Acadêmica da Educação – SIGA-EPCT.

A estrutura do cluster implementada no presente trabalho proporciona alta disponibilidade através do balanceamento de carga entre os nós. Além disso, com a falha de algum nó escravo, o usuário conectado é redirecionado automaticamente para outro nó do cluster sem perder a conectividade e sem a necessidade de fazer o *login* novamente. Com isso, consegue-se um sistema mais robusto, confiável e sempre disponível.

Nos testes de performance apresentados na seção 4.2, realizou-se a compararação entre a máquina standalone e o cluster, onde foi testado um dos módulos do sistema que mais exigem poder computacional do sistema. Percebeu-se nos testes realizados uma diferença de tempo, sendo que, ao contrário do esperado, o tempo de execução foi maior que o sistema disponibilizado de forma *standalone*. Percebeu-se a necessidade de modificação no código fonte para otimização do sistema em cluster. Em virtude da complexidade e do tamanho do sistema, não houve tempo para implementar essas modificações código-fonte.

Observou-se também a quantidade de informações trafegadas na rede, porém esta informação não foi suficiente para comprovar a redução da performance do sistema no cluster comparado ao sistema *standalone*. Verificou-se algumas questões de melhorias no código-

fonte, sugeridas pelo fabricante do servidor de aplicação, que não puderam ser implementadas.

Para trabalhos futuros, pode-se implementar as sugestões de otimização do sistema SIGA-EPTC para cluster e verificar se houve melhora de performance. Além disso, pode-se utilizar a metodologia do presente trabalho para disponibilizar outros sistemas *Web* implementados em java e que utilizam o servidor Glassfish.

REFERÊNCIAS

APPLICATION SERVERS. Disponível em: <http://www.iweb.com.br/iweb/pdfs/20031008-appservers-01.pdf>. Acessado em 10/10/2012

AVERSA, L.; BESTAVROS, A. **Load balancing a cluster of web servers using distributed packet rewriting**. IEEE Int'l Performance, Computing and Communication Conf. Pág. 24 – 29, 2000.

BARBOSA, P. F.; Charão, S. A. **Grid Computing e Cloud Computing – Uma Relação de Diferenças, Semelhanças, Aplicabilidade e Possibilidades de Cooperação entre os dois Paradigmas**. Programa de Pós-Graduação em Informática – Universidade Federal de Santa Maria (UFSM), 2009.

BAKER, M. **Cluster Computing White Paper**, Final Release, Version 2.0, University of Portsmouth, UK, 2000.

BEVILACQUA, A. **Dynamic load balancing method on a heterogeneous cluster of workstations**. Informatica. Volume 23, número 1, pág. 49 – 56, março de 1999.

BUYYA, R. **High Performance Cluster Computing**, Architectures and Systems. Prentice Hall, 1999.

BRUNS, A., KORNSTÄDT, A., and WICHMANN, D, 2009. **Web Application Tests with Selenium**. Published by the IEEE Computer Society. pp. 1.

CARNS, P. H.; LIGON III, W. B.; MCMILLAN, S. P.; ROSS, R. B. **An Evaluation of Message Passing Implementations on Beowulf Workstations**. Proceedings of the 1999 IEEE Aerospace Conference. Março de 1999.

CLOUD, CLUSTER, GRID. Disponível em: <http://staffweb.itsligo.ie/staff/pflynn/Telecoms%203/Level%208%20Projects%202012/Peter%20Donagher%20Thesis%20Proposal.pdf>. Acessado em 16/01/2013

FOSTER, I. **What is the grid?** a three point checklist. Argonne National Laboratory & University of Chicago , 2002.

FOSTER I., YONG, Z.; RAICU, I.; Lu, S. **Cloud Computing and Grid Computing 360-Degree Compared** – Department of Computer Science, University of Chigado, 2008.

JÚNIOR e FREITAS, ESLI e REINALDO. **Construindo Supercomputadores com Linux** - Cluster Beowulf. Monografia apresentada ao Curso de Redes de Comunicação da CEFET-GO, Goiânia 2005.

GORINO, F. V. R. **Balanceamento de Carga em Clusters de Alto Desempenho: Uma Extensão para a LAM/MPI**. Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, 2006.

GRIEBLER, D. J. ; CASALI, V. ; SCHEPKE C. . **Avaliação de Ferramentas de Gerenciamento de Clusters**. In: 6a Escola Regional de Redes de Computadores, 2008, Porto Alegre. Escola Regional de Redes de Computadores - ERRC, 2008. v. 6.

HWANG, K; Xu, Z. **Scalable Parallel Computing,** Technology, Architecture, Programing. *WCB*/*McGraw-Hill*, *1998.*

IPLANET. Disponível em: <http://docs.oracle.com/cd/E19146-01/821-1828/gbrne/index.html>. Acessado: 12/12/2012

MEREDITH, M.; CARRIGAM, T.; BROLMAN, J.; et al. **Exploring beowulf clusters**. Jornal of Computing in Small Colleges. Volume 18, número 4, pág. 268–284, 2003.

NGUYEN, V. A. K.; PIERRE, S. **Scalability of computer clusters**. Electrical and Computer Engineering. Canadian Conference on Volume 1, pág. 405 – 409, de 13 a 16 de maio de 2001.

NAGARAJA, K.; Krishnan, N.; Bianchini, Martin, R.P.; Nguyen, T.D. **Quantifying and Improving the Availability of High-Performance Cluster-Based Internet Services**. Department of Computer Science, Rutgers University, 2003.

ORACLE. Disponível em: <http://docs.oracle.com/cd/E19879-01/821-0188/relnotessges.html>. Acessado em 10/10/2012.

ORACLE ADMINISTRATION GUIDE. Disponível em: <http://docs.oracle.com/cd/E19879-01/820-4335/ablaq/index.html>. Acessado em 12/10/2012.

ORACLE TUNING YOU APPLICATION. Disponível em: <http://docs.oracle.com/cd/E19159-01/819-3681/abebe/index.html>. Acessado em 17/01/2013.

REIS, A.C.F.; Júnior, C.G.S.; Ferreira, G.F.S.; Almeida, J.C.P.; Silva, M.M.; Gavinier, S.A.S. **Cluster de Alta Disponibilidade**. FATEC-GT, São Paulo.

RENAPI. Disponível em: <http://www.renapi.gov.br/sigaepct/o-projeto>. Acessado em 10/11/2012.

SADASHIV, N.; KUMAR, S. M. D. **Cluster, Grid and Cloud Computing: A Detailed Comparison.** The 6th International Conference on Computer Science & Education. Singapore, 2011.

SERVIDOR DE APLICAÇÃO. Disponível em: <http://habil.eti.br/web/servidor_de_aplicacao.php>. Acessado em 11/12/2012

STALLINGS, W. **Arquitetura e Organização de Computadores**. São Paulo: Pearson Prentice Hall, 2008.

TANENBAUM, A.S. **Redes de Computadores**. Elsevier Editora, 2003.

VELOSO, J. S., Alcântara dos S. Neto, P., Santos, I. S., and de Sousa Britto, R. (2011).**Avaliação de Ferramentas de Apoio ao Teste de Sistemas de Informação**. Departamento de Informática de Estatística - Universidade Federal do Piauí (UFPI), Ininga – Teresina – PI – Brasil. pp. 12-16.

WILLIAMS, R. D. **Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations**. Jornal Concurrency. Volume 3, pág. 457 – 481, 1991.

WINCKLER, M; PIMENTA, S. M. **Avaliação de Usabilidade de Sites *Web*.** Instituto de Informática - UFRGS - Porto Alegre - RS -Brasil, 2002.

ANEXOS

Anexo A – Arquivo de configuração sun-web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<sun-web-app>
  <context-root>sigaept-edu-web-v1</context-root>

  <session-config>
    <session-manager persistence-type="replicated">
        <manager-properties>
                <property name="relaxCacheVersionSemantics" value="true"/>
        </manager-properties>
    </session-manager>
  </session-config>

</sun-web-app>
```

Anexo B – Arquivo de configuração web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
      id="WebApp_ID" version="2.5">
      <distributable />
      <display-name>sigaept-edu-web-v1</display-name>
      <servlet>
            <servlet-name>SigaReportServlet</servlet-name>
            <servlet-class>org.sigaept.edu.pentaho.RelatServlet
      </servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>SigaReportServlet</servlet-name>
            <url-pattern>/reports_pentaho</url-pattern>
      </servlet-mapping>
      <servlet>
            <servlet-name>SigaPrimefaces</servlet-name>
            <servlet-class>org.primefaces.resource.ResourceServlet
      </servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>SigaPrimefaces</servlet-name>
            <url-pattern>/primefaces_resource/*</url-pattern>
      </servlet-mapping>
      <context-param>
        <param-name>org.apache.myfaces.NUMBER_OF_VIEWS_IN_SESSION
      </param-name>
        <param-value>40</param-value>
      </context-param>
      <context-param>
            <param-name>com.sun.faces.enableRestoreView11Compatibility
      </param-name>
            <param-value>true</param-value>
       </context-param>
      <context-param>
            <param-name>primefaces.THEME</param-name>
            <param-value>cupertino</param-value>
      </context-param>
      <context-param>
```

```
      <param-name>tituloAplicacao</param-name>
      <param-value>SIGA-EPCT-EDU</param-value>
</context-param>
<welcome-file-list>
      <welcome-file>index.html</welcome-file>
      <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<context-param>
      <param-name>javax.faces.CONFIG_FILES</param-name>
      <param-value>
             /WEB-INF/faces-config-nav.xml
            ,/WEB-INF/faces-config-mb.xml
            ,/WEB-INF/faces-config-app.xml
      </param-value>
</context-param>
<context-param>
      <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
      <param-value>.xhtml</param-value>
</context-param>
<context-param>
      <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
      <param-value>server</param-value>
</context-param>
<context-param>
      <param-name>org.richfaces.SKIN</param-name>
      <param-value>classic</param-value>
</context-param>
<context-param>
      <param-name>org.richfaces.CONTROL_SKINNING</param-name>
      <param-value>enable</param-value>
</context-param>
<context-param>
      <param-name>org.richfaces.CONTROL_SKINNING_CLASSES</param-name>
      <param-value>enable</param-value>
</context-param>
<context-param>
      <param-name>facelets.DEVELOPMENT</param-name>
      <param-value>true</param-value>
</context-param>
<context-param>
      <param-name>org.ajax4jsf.VIEW_HANDLERS</param-name>
      <param-value>com.sun.facelets.FaceletViewHandler</param-value>
</context-param>
<filter>
      <display-name>RichFaces Filter</display-name>
      <filter-name>richfaces</filter-name>
```

```
      <filter-class>org.ajax4jsf.Filter</filter-class>
</filter>
<listener>

<listenerclass>com.sun.faces.config.ConfigureListener</listenerclass>
</listener>
<filter-mapping>
      <filter-name>richfaces</filter-name>
      <servlet-name>Faces Servlet</servlet-name>
      <dispatcher>REQUEST</dispatcher>
      <dispatcher>FORWARD</dispatcher>
      <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
<servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
      <load-on-startup>0</load-on-startup>
</servlet>
<servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
<!-- Configuração SigaAuth -->
<context-param>
      <param-name>SigaAuth.arquivoConfiguracao</param-name>
      <param-value>/WEB-INF/config/SigaAuth.xml</param-value>
</context-param>
<servlet>
      <servlet-name>IniciarPolicyServlet</servlet-name>
<servletclass>org.sigaept.auth.web.IniciarPolicyServlet
</servletclass>
      <load-on-startup>0</load-on-startup>
</servlet>

<!-- Extensions Filter -->
<filter>
      <filter-name>MyFacesExtensionsFilter</filter-name>
<filterclass>org.apache.myfaces.webapp.filter.ExtensionsFilter
</filter-class>
      <init-param>
            <param-name>uploadMaxFileSize</param-name>
            <param-value>1000m</param-value>
      </init-param>
      <init-param>
            <param-name>uploadThresholdSize</param-name>
            <param-value>1000k</param-value>
      </init-param>
```

```
</filter>
<!--extension mapping for adding <script/>,
 <link/>, and other resource        tags to JSF-pages
-->
<filter-mapping>
      <filter-name>MyFacesExtensionsFilter</filter-name>
      <!--
            servlet-name must match the name of your
            javax.faces.webapp.FacesServlet entry
      -->
      <servlet-name>Faces Servlet</servlet-name>
</filter-mapping>
<filter-mapping>
      <filter-name>MyFacesExtensionsFilter</filter-name>
      <url-pattern>/faces/myFacesExtensionResource/*</url-pattern>
</filter-mapping>
<session-config>
  <session-timeout>300</session-timeout>
</session-config>
<error-page>
      <exception-type>javax.faces.application.ViewExpiredException
</exception-type>
      <location>/pages/redireciona_sessionExpired.jsp</location>
</error-page>
<error-page>
      <error-code>404</error-code>
      <location>/pages/redireciona_notFound.jsp</location>
</error-page>
</web-app>
```