Otimização de Desempenho e Escalabilidade do Sistema Falibras-Web com o Uso de Grid Computing

João Paulo Ferreira da Silva[1], **Patrick Henrique da Silva Brito**[1]

[1]Universidade Federal de Alagoas (UFAL) – Campus Arapiraca
Av. Manoel Severino Barbosa, Bom Sucesso, CEP:57309-005, Arapiraca - AL

joao.silva@arapiraca.ufal.br, patrick@ic.ufal.br

Resumo. *O sistema Falibras-Web tem por objetivo a tradução online do texto disponível em qualquer página da internet para a LIBRAS. Porém, o mesmo apresentava problemas relacionados ao desempenho do tradutor e ao número limitado de requisições simultâneas que, numa simples arquitetura cliente-servidor, tornou-se inviável. Neste trabalho, apresentamos uma nova arquitetura de tradução distribuída do sistema Falibras-Web baseada em grid computing com suporte a agregação de recursos de modo voluntário e replicação de dados com o uso de cache. A arquiteura proposta foi avaliada em um ambiente experimental heterogêneo apresentando aumento tanto na escalabilidade como no desempenho do tempo médio de tradução.*

1. Introdução

Devido ao crescimento do uso de sistemas de educação à distância e, consequentemente, da necessidade de sites que ajudem na propagação de informações significativas para os usuários surdos em toda a *Web*, surge o sistema Falibras-Web [Natália M. Franco 2013], que tem como propósito a tradução online do texto disponível em qualquer página da internet, da língua portuguesa, para a Língua Brasileira de Sinais (LIBRAS).

Inicialmente, o Falibras-Web utilizava-se de uma arquitetura cliente-servidor, onde o módulo cliente consistia em uma extensão do navegador Web Firefox. Esse módulo requisitava traduções textuais ao módulo servidor, responsável pelo processo de tradução. Apesar dos resultados iniciais terem sido promissores, essa versão apresentava problemas estruturais que comprometeram sua viabilidade técnica. Os principais problemas encontrados são relacionados ao desempenho do tradutor e ao número limitado de requisições que o servidor conseguia atender.

Devido ao cenário acima, houve a necessidade de projetar uma arquitetura distribuída para o Falibras-Web. Dadas as restrições de recursos e a natureza colaborativa do projeto, optou-se pela utilização de *Grid Computing* [I. Foster 1999], aliado ao conceito de *Volunter Computing* [LHC@Home 2015]. A principal motivação para a adoção de *grid*, ao invés de *cluster*, por exemplo, foi a necessidade de se ter recursos distribuídos em vários locais utilizando recursos computacionais heterogêneos, além de se contar com um número variável e dinâmico de nós, facilitando a entrada e saída de voluntários na rede de processamento colaborativo.

Além do aumento da escalabilidade do sistema, a nova arquitetura proposta visa melhorias no desempenho e disponibilidade em um contexto que envolve grande demanda de requisições simultâneas de traduções. Este trabalho apresenta uma nova ar-

quitetura de tradução do Falibras-Web baseada em *Grid Computing* para o balanceamento de carga, com especial destaque para a questão do *Grid Computing* aplicado ao uso de *Volunter Computing*. Em relação à problemática do desempenho e disponibilidade, a arquitetura proposta adota o conceito de replicação de dados com o uso de *cache* [Andrew S. Tanenbaum 2007].

A nova solução proposta foi avaliada em um experimento prático e apresentou resultados satisfatórios, com ganhos representativos tanto no desempenho, quanto na escalabilidade e disponibilidade do sistema. O experimento envolveu requisições simultâneas de traduções, a partir de um corpus padrão adotado no projeto Falibras [Patrick H. Brito 2012].

2. Metodologia

O *framework* utilizado para o desenvolvimento da nova arquitetura de *software* baseada em *grid* foi o JPPF 5.0.4 [JPPF 2015]. Por ser escrito em Java, uma linguagem de programação multiplataforma, e por ser um projeto *opensource*. O JPPF fornece diversas abstrações para representar tarefas, trabalhos, etc. As *interfaces* utilizadas para este trabalho foram os *Tasks* e os *Jobs*. *Tasks* são usados para representar qualquer tarefa que será executada no *grid*. Já um *Job* é formado por um conjunto de *Tasks*. E ambos são executados usando o algoritmo de escalonamento *Round-Robim* [Tanenbaum 2008]. No contexto do projeto em questão, cada requisição de tradução é considerada uma *Task*.

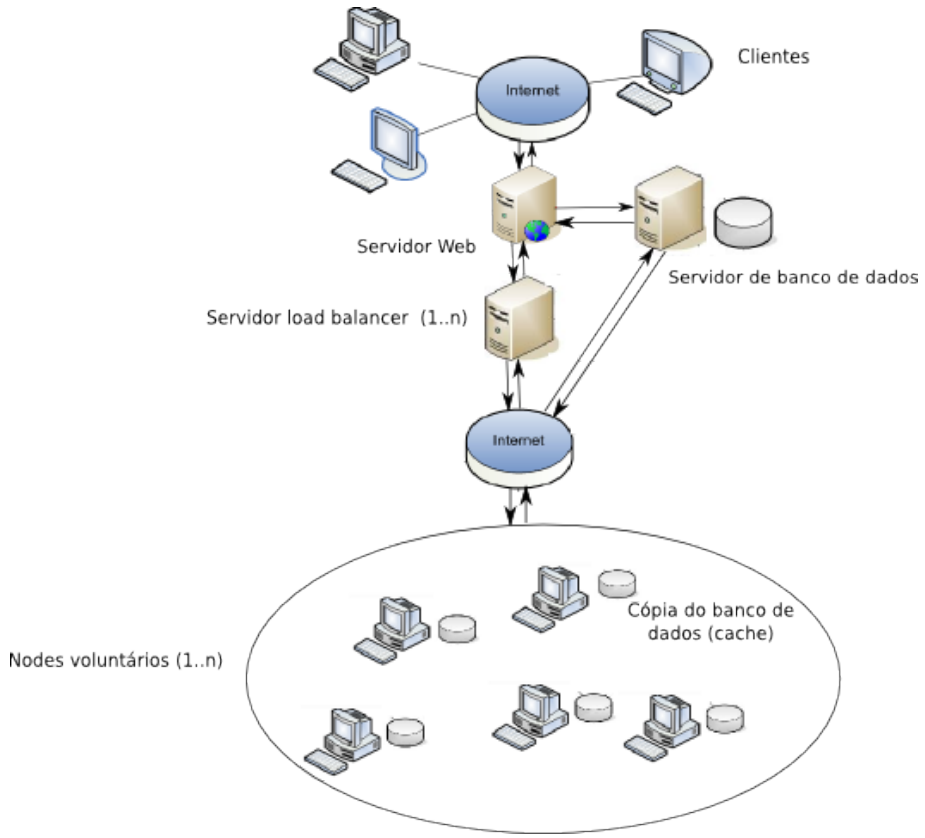

Figura 1. Arquitetura de tradução do Falibras-Web baseada em grid computing.

A Figura 1 representa a arquitetura de tradução distribuída do Falibras-Web. A mesma é composta por vários componentes apresentados a seguir.

- *Nodes* - Usuários voluntários que contribuem com seus recursos computacionais ao *grid* processando as traduções;
- *Clientes* - Requisitam as traduções ao sistema via serviços *Web* [Reis 2005];
- *Servidor web* - Responsável por receber as solicitações dos clientes e encaminhá-las ao servidor de tradução;
- *Servidor de banco de dados* - Responsável por armazenar o banco de dados de tradução, além de disponibilizar os dados de *cache* para os *nodes*;
- *Servidor "load balancer"* - Tem o papel de escalonar tarefas de tradução aos *nodes* utilizando o algoritmo *Round-Robin*, com a possibilidade de ter mais de um servidor.

De um modo geral, quando o cliente requisita uma tradução de uma ou mais frases ao sistema através da interface do Falibras-Web, a frase é inserida em uma *task* dentro de um *job*. Onde é transferido para o *servidor "load balancer"*, atuando como um escalonador, e enviando para algum *node* ocioso conectado ao *grid*. Em seguida, o *node* processa a tradução da frase e retorna uma lista de glosas. Onde a mesma percorre o caminho inverso e, no final, é exibida a frase traduzida para o cliente.

Para que um usuário se torne um *node* voluntário, basta que o mesmo baixe e execute a aplicação denominada "Falibras node"que é uma extensão do componente "JPPF node". Com isso, no momento em que o computador do usuário estiver inativo, o "Falibras node" iniciará uma proteção de tela e solicitará uma cópia do banco de dados de tradução, a ser armazenada na memória RAM como um *cache* de tradução. Tal solicitação é realizada através de *webservices RESTFul* [Leonard Richardson 2007]. Em seguida, é estabelecida a conexão do *node* ao *grid*, pronto para realizar traduções. Para o desenvolvimento dos *webservices*, foi utilizado a biblioteca Jersey 2.22.1 [Jersey 2015].

3. Resultados

Foram realizadas uma série de experimentos de testes de performance [Sommerville 2011] para aferir o comportamento do *grid* em situações como o aumento progressivo de clientes requisitando traduções e baixa disponibilidade de nodes voluntários.

O ambiente utilizado para a execução de cada experimento era formado por 5 computadores modelo HP EliteDesk 800 equipados com processadores Intel Core i5 vPro e 4GB de RAM sendo que, 4 destes com sistema operacional Linux Ubuntu 14.04 x64 (1 para a simulação dos clientes, 1 servidor mestre e 2 escravos) e 1 com sistema operacional Windows 8 Pro x64 (escravo), interligados numa rede local. O programa usado para a simulação dos clientes consistiu em um *script* escrito em *Java* contendo um *ArrayList* com 24 frases diferentes. A chamada do método de tradução de frases foi inserida em uma estrutura de repetição com execução infinita e, a cada iteração, sorteava-se uma frase da lista para a tradução no *grid*. Esse trecho de código foi inserido em várias *threads* e executadas na máquina cliente. Com isso, foi possível simular vários clientes requisitando traduções de diferentes frases. Para o monitoramento, foi utilizado o painel *console* do JPFF. Cada teste foi executado durante 5 minutos.

Para o primeiro teste, foram instanciados 16 clientes requisitando traduções com apenas 1 *node* voluntário conectado ao *grid*. Foram processadas 1568 traduções com tempo médio aproximado de 407 ms para cada tradução. O tempo médio de transporte de

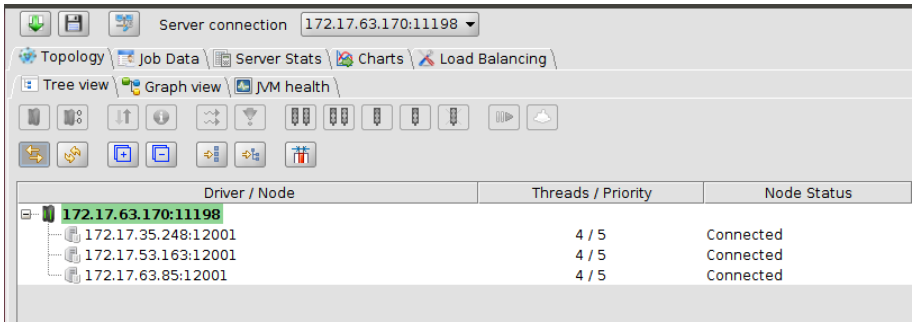

Figura 2. Painel console do JPPF exibindo 3 nodes conectados.

Network Overhead foi de 79 ms e o de espera na fila foi de 274 ms (760 ms no total). Para o segundo teste, foram usados 2 nodes voluntários com os mesmos 16 clientes. Foram processadas 1588 traduções com tempo médio de 241 ms para cada tradução. O tempo médio de transporte de *Network Overhead* foi de 76 ms e o de espera na fila foi de 110 ms (427 ms no total). E finalmente no terceiro teste, foram instanciados 16 clientes com 3 nodes voluntários. Foram processadas 1588 traduções com tempo médio de 196 ms por tradução. O tempo de médio de transporte de *Network Overhead* foi de 81 ms e o de espera na fila foi de apenas 53 ms (330 ms no total), como é representado na Tabela 1.

Tabela 1. Resultados dos testes de performance.

Testes de Performance	Quantidade de nodes voluntários	Quantidade de clientes	Quantidade de traduções processadas	Tempo médio de processamento por tradução (ms)	Network Overhead (ms)	Tempo médio de espera na fila (ms)
Teste 1	1	16	1568	407 ms	79 ms	274 ms
Teste 2	2	16	1588	241 ms	76 ms	110 ms
Teste 3	3	16	1588	196 ms	81 ms	53 ms

Numa segunda comparação, foi feita uma avaliação do impacto da adoção da nova arquitetura, quando comparada com a centralizada, que se baseava numa arquitetura cliente-servidor básica. Ao executar o mesmo experimento nessa arquitetura, o sistema apresentou tempo médio de tradução de 3032 ms mesmo considerando-se uma solução com apenas um único nó. Constatamos que esse ganho acentuado se deveu à adoção do mecanismo de *cache*, pois sem este recurso o tempo médio de tradução era de aproximadamente 6542 ms. No quesito escalabilidade, ao executar requisições concorrentes de tradução, a arquitetura anterior só conseguiu atender 16 requisições de tradução, apresentando erro fatal logo após. Tal erro evidenciava falta de recursos computacionais (exceção de falta de memória). Atribuímos essa grave limitação à ausência de um servidor de escalonamento, responsável por gerenciar uma fila de requisições.

4. Conclusões e Trabalhos Futuros

O balanceamento de carga é uma técnica bastante explorada em áreas como a computação paralela e sistemas distribuídos. *Computing Grids* são sistemas distribuídos com grande capacidade de escalabilidade e heterogeneidade com a possibilidade do uso de *Volunter Computing* para agregação de recursos gratuitos fornecidos por voluntários, dimuindo custos na aquisição de recursos. O processo de replicação de dados com o uso de *cache* diminui o tráfego de rede, oferecendo ao mesmo tempo, desempenho no processamento.

Os experimentos realizados para o teste de performance atingiram os resultados esperados, mostrando que o uso de *Grid Computing* em aplicações paralelas pode apresentar ganhos siginificativos de desempenho no tempo de reposta para aplicações que se beneficiam com o uso dessa técnica. Dessa forma, espera-se que este trabalho contribua com uma melhor infraestrutura para a utilização real do Falibras-Web no contexto de um novo módulo educacional, voltado para a educação em LIBRAS.

Em trabalhos futuros, pretende-se avaliar a atuação do *grid* em cenários reais de alta demanda, envolvendo serviços de tradução integrados a redes sociais e dispositivos móveis. Outro trabalho previsto é a implementação de criptografia no processo de transmissão de dados entre os componentes do *grid*, de forma a aumentar a confidencialidade das mensagens.

Referências

Andrew S. Tanenbaum, M. V. S. (2007). *Sistemas Distribuídos: Princípios e Paradigmas*. Person Education Inc., 2nd edition.

I. Foster, K. K. (1999). The grid: Blueprint for a new computing infrastructure. In Francisco, C. S., editor, *The Grid: Blueprint for a new Computing Infrastructure*. Morgan Kaufmann Publishers.

Jersey (2015). Jersey: Restful web services in java. https://jersey.java.net/. Acessado em: 14/10/2015.

JPPF (2015). Jppf: The open source grid computing solution. http://www.jppf.org/. Acessado em: 14/10/2015.

Leonard Richardson, S. R. (2007). *RESTFul Web Services*. O'Reilly, 1st edition.

LHC@Home (2015). What is volunteer computing? http://lhcathome.web.cern.ch/about/what-volunteer-computing. Acessado em: 14/10/2015.

Natália M. Franco, Patrick H. Brito, L. C. C. (2013). Falibras-web: Acessibilidade de pessoas surdas na web em libras utilizando design colaborativo. *Nuevas Ideas en Informática Educativa TISE 2013*.

Patrick H. Brito, Natália M. Franco, L. C. C. (2012). Falibras: uma ferramenta flexível para promover acessibilidade de pessoas surdas. *Memorias del XVII Congreso Internacional de Informática Educativa, TISE*.

Reis, V. Q. (2005). Escalonamento em grids computacionais: estudo de caso. Master's thesis, Departamento de Informática e Estatística - Universidade Federal de Santa Catarina.

Sommerville, I. (2011). *Engenharia de Software*. Person Education Inc., 9th edition.

Tanenbaum, A. S. (2008). *Sistemas Operacionais Modernos*. Person Education Inc., 3nd edition.