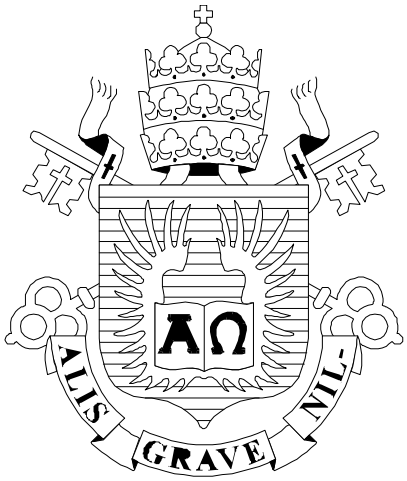


PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° XX/XX

Instruções para Autores na Série Monografias em Ciência da Computação

Autor 1 Bernardo Breder

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. XX/AA ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Mês/Month, AAAA

Instruções para Autores na Série Monografias em Ciência da Computação *

Autor 1(Bernardo Breder)[1]

[1]Instituto de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-RIO)

bbreder@inf.puc-rio.br

Abstract. This paper presents a new approach to Web programming using the advantages of Multi Agent. Among other aspects, the solution Multi Agent can be used to increase the cooperation of agents in computational effort of a Web Server. With this, the agents can help to process the request information of the servers, and delegate to agent respond. This collaboration with agents of the computational effort that has a Web server, it will be better comment in the work.

Keywords: Multi Agent, Distributed Server, Agent Breder

Resumo. Este trabalho apresenta uma nova abordagem de programação Web utilizando as vantagens que o mundo Multi Agente. Entre vários aspectos, a solução Multi Agente pode ser usado para aumentar a colaboração dos Agentes no esforço computacional de um Servidor Web. Assim, os Agentes iram ajudar a processar as informações requisitadas aos Servidores, sendo delegado a elas responderem. Essa colaboração dos Agentes com o esforço computacional que um Servidor Web possui, será melhor abordado no trabalho.

Palavras-chave: Multi Agente, Servidor Distribuído, Agent Breder

---

* (este, ou outro, quando for aplicável) Trabalho patrocinado pelo Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil (e agência de fomento e o número do processo, se aplicável). (Em Inglês: This work has been sponsored by the Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil)

Sumário

1 Introdução 1

2 Ambiente 1

2.1 Servidor 1

2.2 Usuário 1

3 Arquitetura 1

3.1 Servidor 2

3.2 Ciclo de Vida de um Agente Servidor 3

3.3 Linguagem de Programação 4

4 Desempenho 5

5 Conclusão 6

Referências 7

1 Introdução

O parque de servidores[1], distribuídos pelo mundo inteiro, é muito acessado por diferentes pessoas com objetivos distintos. Esse servidores podem falhar por mais protegidos e seguros que sejam[1]. Assim, a credibilidade do serviço pode ser comprometido em função das quedas dos servidores em horários críticos.

O objetivo do projeto é descentralizar o parque de servidores em dispositivos ou máquinas que estão espalhado pelo mundo. Dessa forma, as requisições dos servidores não serão mais respondidos por eles e sim por máquinas distribuídas pelo mundo que estão dispostos a disponibilizar o seu processador e sua memória através de uma página Web.

O capítulo 2 irá abordar qual são os ambientes disponíveis pela arquitetura, tanto para o servidor, quanto para o cliente. O capítulo 3 irá discutir o funcionamento da arquitetura. O Capítulo 4 irá apresentar resultados de desempenho da arquitetura. Por último, o capitulo 5 irá concluir o estudo da arquitetura apresentando seus trabalhos futuros.

2 Ambiente

Esse tópico irá abordar o ambiente de desenvolvimento e a linguagem de programação para desenvolver um aplicação Web através dessa arquitetura. Além disso, será apresentado o meio pelo qual o servidor será inicializado pelo dispositivo ou pela máquina. Por fim, será mostrado o caminho pelo qual o usuário irá usufruir do serviço disponibilizado por este ambiente.

2.1 Servidor

O servidor de uma aplicação será carregado por qualquer dispositivo ou máquina que possui um Navegador de Internet. Através de um site, um agente servidor [7] poderá ser inicializado, disponibilizando os recursos do dispositivo ou máquina para o agente servidor responder às requisições solicitadas.

Dessa forma, o ambiente de um agente servidor é constituído de um navegador que sofre de diversas limitações impostas pela máquina virtual JavaScript [2] e pelo protocolo HTTP [3]. Um exemplo de limitação está no acesso restrito ao sistema de arquivo da máquina ou dispositivo do usuário.

2.2 Usuário

O usuário irá acessar a aplicação Web através do Navegador de Internet na forma tradicional. Para o usuário da aplicação, nenhuma mudança no procedimento de acessar será feita. Assim, o usuário não sofrerá nenhum impacto na arquitetura imposta pela aplicação web.

3 Arquitetura

Os tópicos abaixo irão descrever a arquitetura implantada no sistema. A sessão 3.1 descreve a arquitetura do servidor e seu funcionamento. A sessão 3.2 irá mostrar o

ciclo de vida de um agente servidor. Por último, a sessão 3.3 irá comentar sobre a linguagem de programação que um desenvolvedor de aplicação irá utilizar para construir um projeto.

3.1 Servidor

Um parque de servidor é composto por várias máquinas que respondem a requisições de algumas aplicações. Esses servidores sofrem com inúmeros problemas de manutenção, segurança e custo[1].

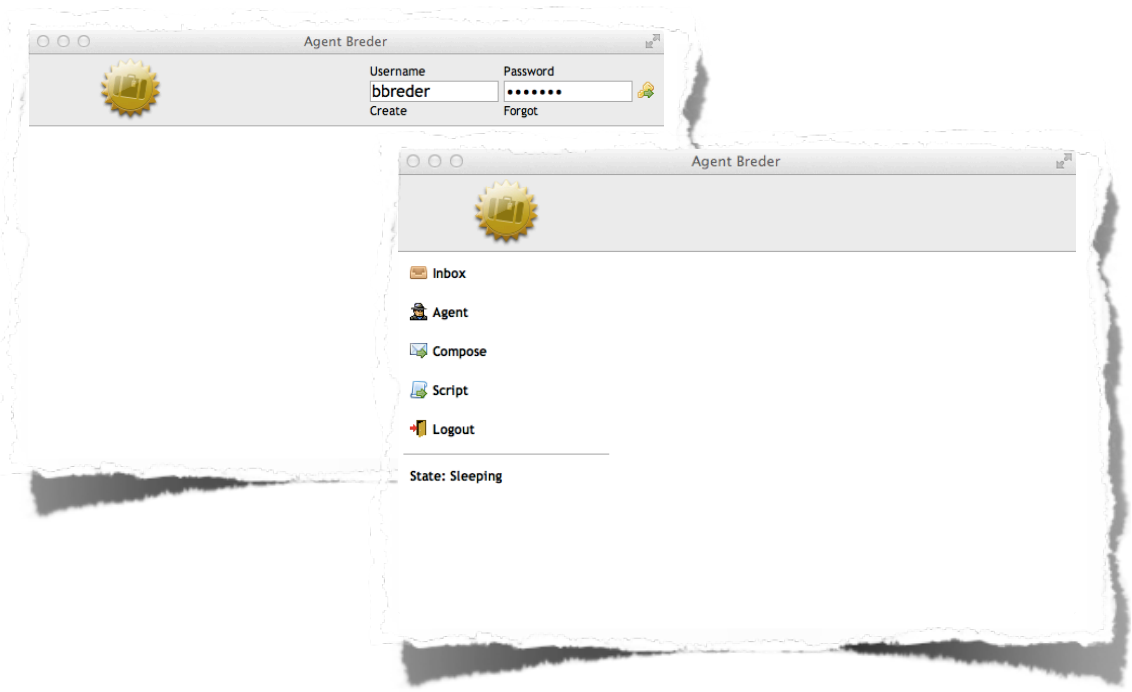
Entre vários problemas, um parque de servidor pode sofrer de queda de luz, de internet, defeito no hardware, aquecimento das máquinas e entre outros problemas[1]. Além disso, esses problemas podem ser intensificados quando o parque de servidor se concentra num mesmo local utilizando os mesmo recursos.

O projeto Agent Breder tem como objetivo descentralizar os servidores de uma aplicação, distribuindo o seu processamento para os dispositivos e máquinas pessoais espalhadas pelo mundo inteiro.

A idéia principal da solução é utilizar dos dispositivos moveis ou máquinas pessoais para ajudar a processar as requisições de um servidor centralizado. Através dessa ajuda, o parque de servidor poderá deixar de existir ou irá diminuir a sua escala, tornado a responsabilidade dos dispositivos e máquinas pessoais o papel de responder às requisições das aplicações.

Dessa forma, o servidor de aplicação passará a ser descentralizado, espalhado pelo mundo, diminuindo a probabilidade de ocorrer os problemas citados. Além disso, a solução irá aumentar a escalabilidade do servidor de aplicação, a medida que o número de dispositivos e máquinas pessoais aumentem, ajudando ainda mais no processamento.

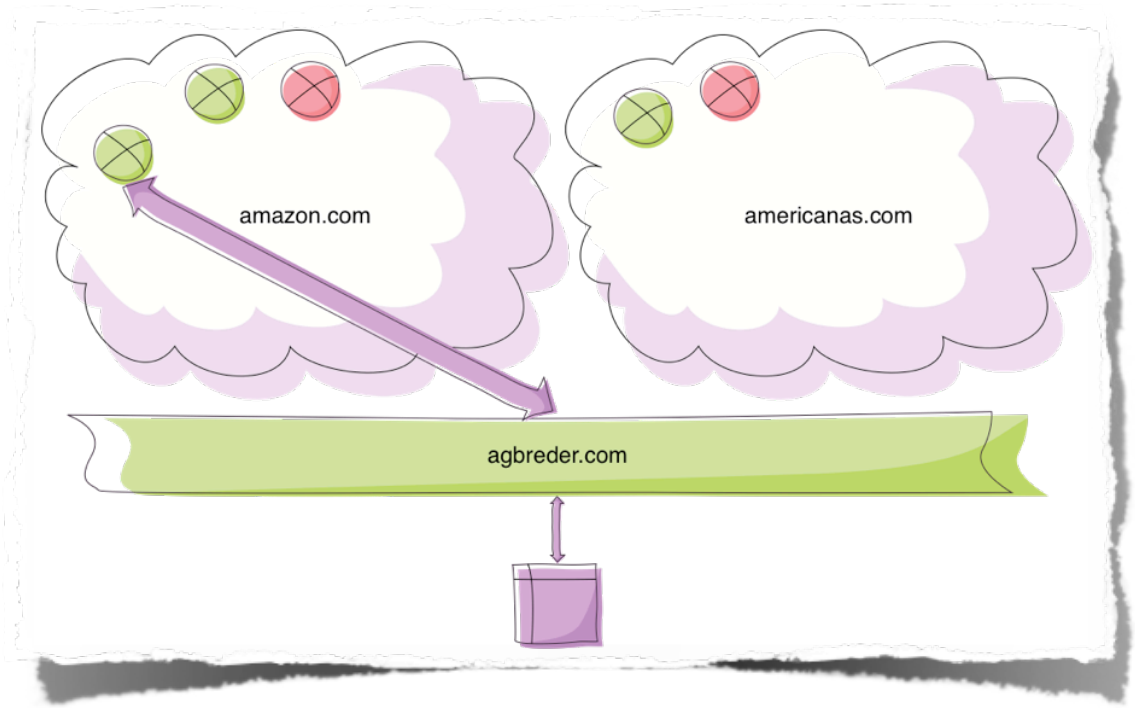
Através dessa arquitetura, um servidor de aplicação pode ser inicializado ou colaborado através de um Navegador de Internet estando nas maiorias dos dispositivos celulares e em quase todas as máquinas pessoais. Para disponibilizar os recursos do seu dispositivo para ajudar ao servidor, basta entrar no site do www.agbreder.com e efetuar a sua autenticação.



Quando um dispositivo dispõem dos seus recursos para processar requisições de um servidor, um agente de servidor será criado, sendo redirecionado parte das requisições para este dispositivos, fazendo com que ele seja uma fração de um servidor de aplicação.

A arquitetura é dividida em duas parte. A primeira representa um servidor central, escapável, que recebe todas as requisições de todas as aplicações e redireciona para a segunda parte responde-la. A segunda parte os agentes servidores que estão dispostos a receber requisições e responde-la, automaticamente.

A primeira parte da arquitetura representa um servidor que somente gerencia a troca de mensagens entre o navegador do usuário e o dispositivo disponível a processar a requisição. A segunda parte representa uma camada de real processamento da requisição, fazendo com que seja efetuado a compilação do projeto da requisição e o execute, retornado a página HTML.

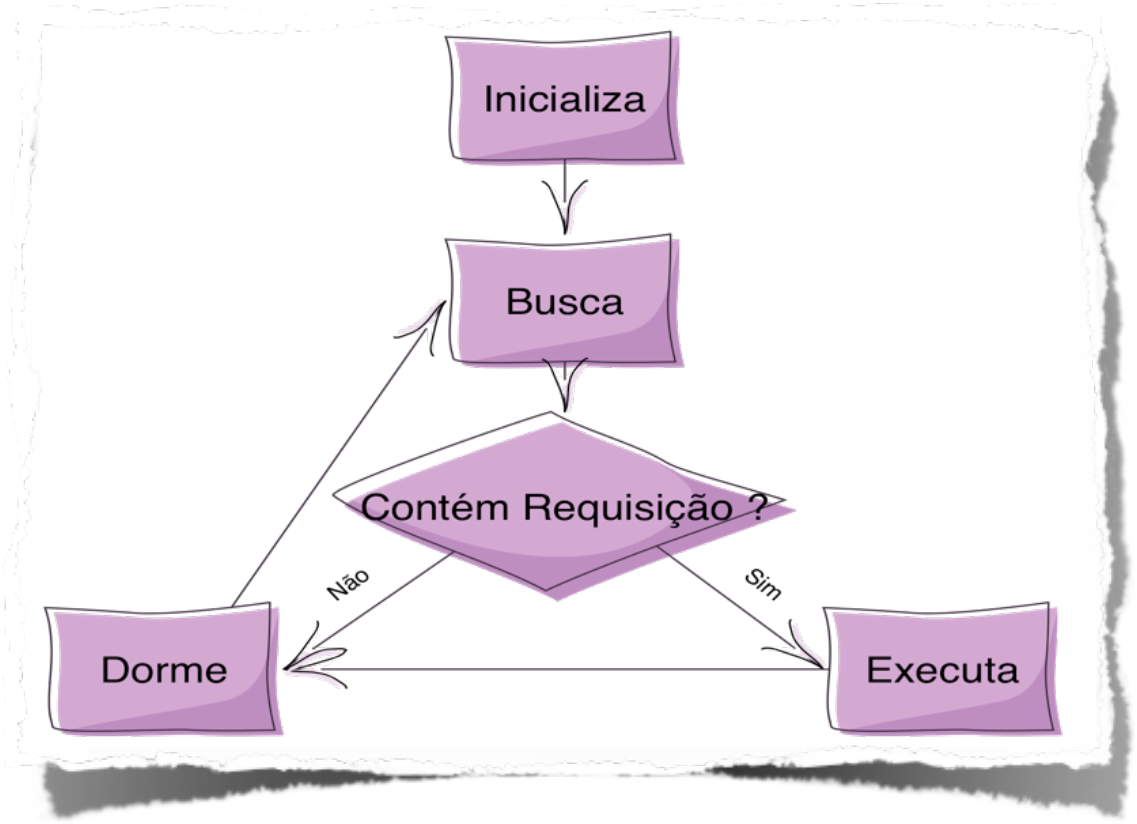


A compilação da aplicação e a sua execução associado a uma requisição efetuada pela segunda camada é totalmente implementado em Javascript. Porém, a Linguagem de Programação para criar um aplicação web utilizando está arquitetura é processada pela linguagem Agent Breder Language na qual a sua Máquina Virtual também está implementada em Javascript. Portanto, o projeto de aplicação web é feito na Linguagem Agent Breder Language, mas o real processamento é todo feito em Javascript.

3.2 Ciclo de Vida de um Agente Servidor

Um agente servidor é iniciado na autenticação do usuário com o sistema. A inicialização do agente servidor faz com que seu estado mude para "buscando".

O estado "buscando" representa um espera por uma requisição de algum usuário que deseja acessar alguma página do sistema. Devido a limitações do protocolo HTTP, essa espera é implementada através de requisições constantes ao servidor central perguntando se existe alguma requisição não respondida. De fato, essas requisições HTTP constante pode ser um "gargalo" para um queda de desempenho tanto do servidor central, quanto do agente servidor.



Quando o servidor central indica que existe uma requisição não tratada a um agente servidor, o seu estado muda de "buscando" para "executando". Nesse estado, o agente recebe do servidor central a requisição que deve ser respondida. Quando recebido, o agente irá compilador o projeto, com a linguagem de programação que será descrita no próximo tópico, e irá executar com os argumentos descritos na requisição. No termino da execução, o agente responde ao servidor central a página HTML correspondente a requisição solicitada.

Quando um servidor central receber a resposta HTML de um requisição tratada por algum agente servidor disponível, será enviado para o usuário final o conteúdo da resposta em que esteja esperando. Quando essa operação é concluída, o agente servidor que respondeu a requisição volta ao estado "buscando" para adquirir novas requisições não tratada ou respondidas.

O agente servidor somente finaliza seu ciclo de vida quando o usuário sai do sistema.

3.3 Linguagem de Programação

A Linguagem de Programação utilizada para implementar uma aplicação nesse sistema foi construída completamente na linguagem JavaScript. Isso porque o agente servidor precisaria compilar e executar um projeto no ambiente de um Navegador.

Essa linguagem foi construída com o objetivo de trazer mais ferramentas para os desenvolvedores de uma aplicação e também para limitar o acesso ao recursos do navegador.

A linguagem foi inspirada na Linguagem Breder[4] que foi reimplementada para com a linguagem JavaScript. Essa reimplementação diminui o escopo das funcionalidades da linguagem Breder, mais trouxe a possibilidade de se compilar um código

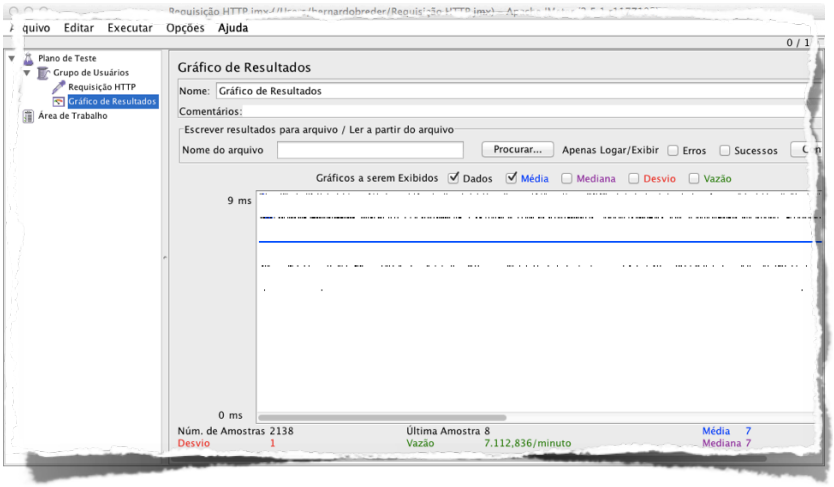
fonte através de um navegador. Além disso, uma queda de desempenho foi agravado devido a reimplementação está escrito na linguagem JavaScript.

Para acessar mais informações do funcionamento da Linguagem Breder[4], basta entrar no site oficial "www.breder.org/lang" e buscar pelo link de documentação.

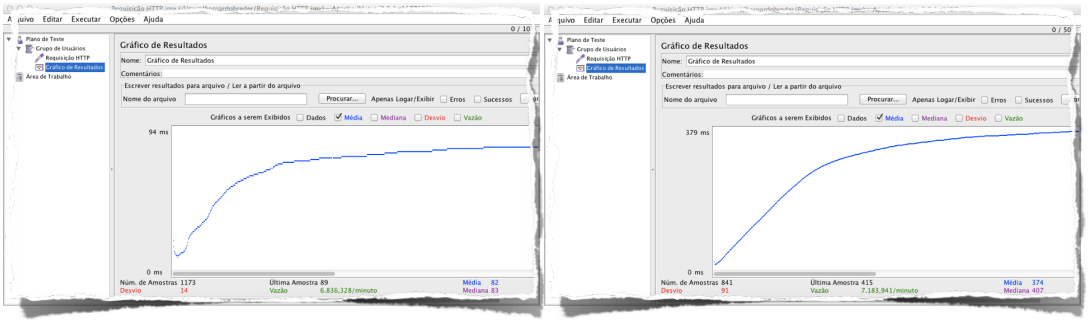
4 Desempenho

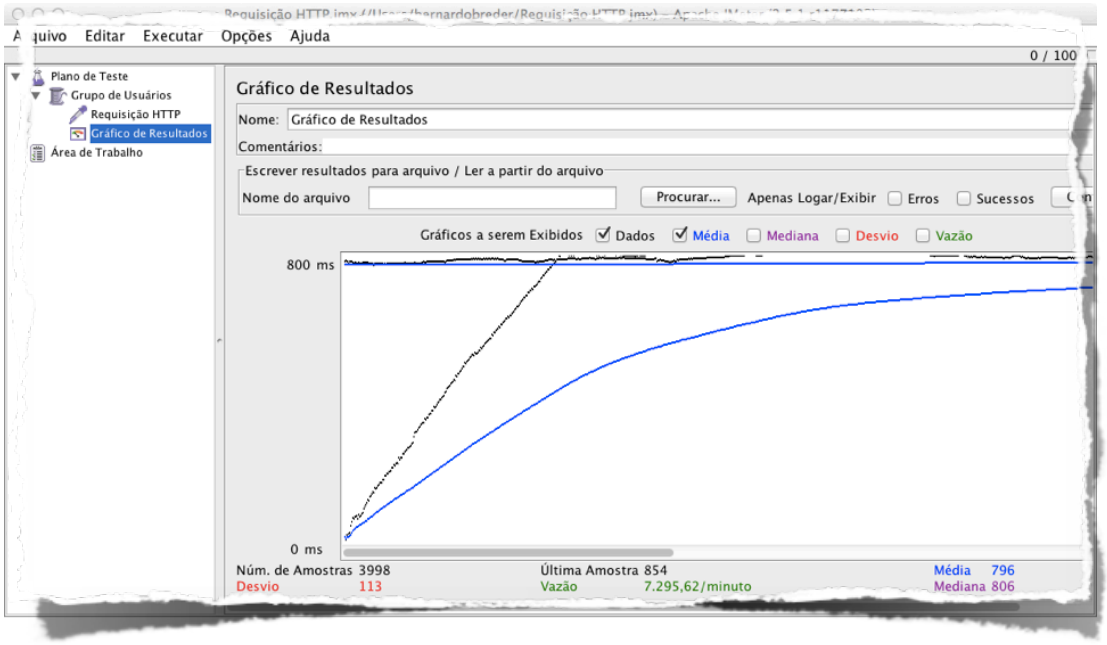
Esse tópico irá mostrar alguns gráficos que demonstram o grau de desempenho que um agente servidor suporta para uma aplicação. Todos os gráficos abaixo utilizam de uma ferramenta de Teste de Stress[6] chamada JMeter[5] que foi utilizado para testar o sistema.

O teste abaixo mostra um usuário por vez requisitante uma página no sistema. Esse teste mostra que o tempo de resposta para um usuário por vez é de 7ms na media. O tempo de resposta corresponde ao tempo de requisição de uma página, recebimento por algum agente servidor, compilação do projeto, execução do algoritmo de geração de página e envio ao solicitante.

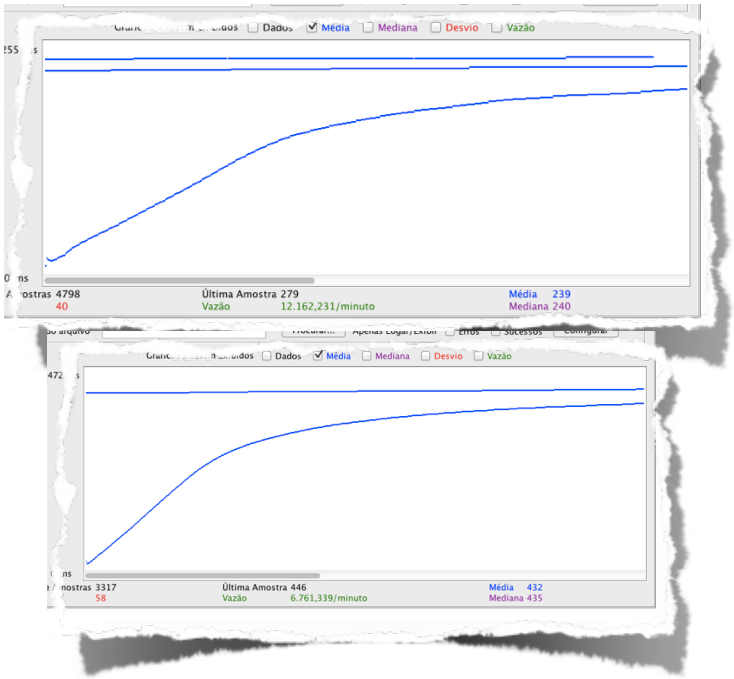


O gráfico abaixo aumenta o número de usuários que estão requisitando uma página do sistema. Esse número de usuários variam de 10, 50, 100 requisições por vez e seu tempo de resposta é de 82, 374 e 796 milissegundos, respectivamente.





A seguir, o gráfico irá mostrar a diferença que 2 agentes de servidor atuam ao invés de somente 1 agente. Como mostrado no gráfico, ao acrescentar mais um agente, o tempo médio de resposta passa de 432 milissegundos para 239 milissegundos, dando um acrescimento de 80% no ganho de desempenho.



5 Conclusão

O projeto Agente Breder foi construído para descentralizar o parque de servidores. Uma aplicação de exemplo foi construído com o objetivo de medir o grau de desempenho e sua viabilidade.

O resultado do grau de desempenho foi bem satisfatório devido aos servidores estarem mesmo descentralizados e possui somente um servidor central com o mínimo

possível de responsabilidade. Além disso, a aplicação web construída que descreve o comportamento do agente servidor tem trago bons resultados de desempenho.

O resultado de portabilidade do sistema para os dispositivos mais limitados tem trago ótimos resultados. Testes foram efetuados para iPhone, iPad, Android e até dispositivos mais primitivos trouxeram resultados esperados.

O desempenho da máquina virtual Breder cai devido a sua implementação original ser passada da linguagem C Ansi para Javascript.

Com conclusão do exemplo criado para demonstrar o ciclo completo do sistema, seus resultados foram todos os esperados, tanto a nível de desempenho, quanto a nível de portabilidade e transparência para o usuário final.

Referências

[1] Virtualização A TI virtual. Disponível em http://www.ibm.com/expressadvantage/br/articles_businessunit_4Q03.phtml

[2] Desvantagem do Javascript. Disponível em http://www.avellareduarte.com.br/projeto/recursos/recursos4/recursos4e.htm

[3] Protocolo HTTP 1.1. Disponível em http://www.w3.org/Protocols/rfc2616/rfc2616.html

[4] Breder Language. Disponível em http://www.breder.org/lang

[5] JMeter. Disponível em http://jmeter.apache.org/

[6] Teste de Stress. Disponível em http://qualidade-de-software.blogspot.com/2010/01/teste-de-estresse.html

[7] Mobile Agents: The Next Generation in Distributed Computing. Por Robert Gray, David Kotz, Saurab Nog, Daniela Rus, George Cybenko