André Spanguero Kanayama

Distribuição de carga em execução de scripts no sistema VisionDataset

Brasil

2013, v-1.1

André Spanguero Kanayama

Distribuição de carga em execução de scripts no sistema VisionDataset

Desenvolvimento de um método para a execução de scripts no VisionDataset, sistema cujo desenvolvimento já foi iniciado anteriormente.

Universidade de São Paulo – USP

Instituto de Matemática e Estatística

Orientador: Prof. Dr. Roberto Hirata Jr.

Brasil

2013, v-1.1

"Simple' does not mean 'easy'. I have learned
that the things that seem the simplest are often the most powerful of all."
(Christie Golden, Thrall: Twilight of the Aspects)

Sumário

Parte I

Objetiva

Introdução

Com o advento da Internet, cada vez mais usa-se ferramentas colaborativas on-line para o desenvolvimento científico. Com o objetivo de facilitar esta colaboração, surgiu o VisionDataset. Criado por Bruno Klava e pela professora Nina S. T. Hirata, este sistema tinha como objetivo centralizar as imagens utilizadas nos projetos desenvolvidos pelo grupo de pesquisa em visão computacional do IME-USP.

Na área de processamento de imagens, muitas vezes temos que trabalhar com imagens muito grandes, como imagens médicas ou espaciais, que chegam a ter mais de 100 Megapixels. Baixar tais imagens para o computador, para depois realizar algum tipo de processamento e depois subi-las novamente para o VisionDataset é inviável, pois além de consumir muita banda, o processamento seria muito pesado, principalmente se o usuário estiver utilizando um computador pessoal. Sendo assim, desejamos criar uma estrutura para que a execução de scripts seja possível no VisionDataset, dando liberdade para o usuário usar os recursos que precisa, mas sem afetar o funcionamento do sistema.

Este trabalho consiste em desenvolver um método para realizar a execução de scripts sobre imagens no VisionDataset de forma eficiente, para isso serão utilizadas outras máquinas, no princípio apenas virtuais, que reportam para o servidor o seu estado, sendo assim ele pode escolher em qual delas, e como fará o processamento das imagens automaticamente, sendo que o usuário verá apenas o resultado final.

1 Conceitos e Tecnologias

1.1 Máquinas virtuais

Uma máquina virtual é um computador que funciona dentro de outro, através de software. A principal vantagem de uma máquina virtual é seu isolamento em relação ao servidor (também conhecido como *host*), sendo assim mesmo que aconteça algum problema com a máquina virtual, ainda é possível gerenciá-la remotamente. A máquina virtual não precisa ter o mesmo sistema operacional que o *host*, portanto também existe a opção de ter vários sistemas operacionais rodando simultaneamente em um computador só, diminuindo custos com infraestrutura.

"As máquinas virtuais podem ser divididas em três tipos:

- Tipo 1 - Sistema em que o monitor é implementado entre o hardware e os sistemas convidados (*guest system*).
- Tipo 2 - Nele o monitor é implementado como um processo de um sistema operacional real, denominado sistema anfitrião (*host system*).
- Tipos Híbridos - Os monitores de tipo 1 e 2 raramente são usados em sua forma conceitual em implementações reais. Na prática, várias otimizações são inseridas nas arquiteturas apresentadas, com o objetivo principal de melhorar o desempenho das aplicações nos sistemas convidados. Como os pontos cruciais do desempenho dos sistemas de máquinas virtuais são as operações de E/S, as principais otimizações utilizadas em sistemas de produção dizem respeito a essas operações.

Outra importante categoria de máquinas virtuais são as máquinas virtuais para computadores fictícios projetados para uma finalidade específica. Atualmente a mais importante máquina virtual desta família é a JVM (máquina virtual Java). Existem simuladores para ela em quase todos os computadores atuais, desde computadores de grande porte até telefones celulares, o que torna as aplicações Java extremamente portáveis."(http://pt.wikipedia.org/wiki/Máquina_virtual)

1.2 PostgreSQL

O PostgreSQL é um sistema gerenciador de bancos de dados objeto-relacional. Ele possui o código aberto e é capaz de rodar em vários sistemas operacionais, além de ser

muito confiável, sendo um dos sistemas gerenciadores de bancos de dados mais usados no mundo. Sua versão mais recente é capaz de lidar com tabelas de até 32 TB sendo que em cada tabela podem haver até 1600 colunas.

1.3 Maven

O Maven é uma tecnologia desenvolvida pela Apache Software Foundation e que é utilizada para gerenciamento e compilação de projetos feitos na linguagem Java. Com ele é possível descrever as dependências de um projeto em um arquivo, chamado pom.xml, e quando a compilação for realizada, todas elas são verificadas e, caso não sejam satisfeitas, os pacotes necessários são baixados automaticamente, tornando a compilação e o compartilhamento do código muito mais fáceis. Além disso o Maven oferece inúmeras ferramentas para testes e documentação.

1.4 Aplicação Web

Uma aplicação web é um sistema projetado para ser utilizado através do navegador. Esse sistema fica em um servidor, no qual é feito o processamento das requisições, sendo assim é possível utilizá-lo à partir de qualquer computador, independentemente do sistema operacional e do hardware. Para entender o funcionamento da aplicação web, especialmente no caso do VisionDataset, é necessário conhecer algumas tecnologias:

1.4.1 Servidor HTTP Apache

"Servidor web pode ser um programa de computador responsável por aceitar pedidos HTTP de clientes, geralmente os navegadores, e servi-los com respostas HTTP, incluindo opcionalmente dados, que geralmente são páginas web, tais como documentos HTML com objetos embutidos (imagens, etc.) ou um computador que executa um programa que provê a funcionalidade descrita anteriormente."
(http://pt.wikipedia.org/wiki/Servidor_web)

O Apache é um servidor HTTP livre e atualmente é o mais utilizado no mundo. Com ele é possível redirecionar um endereço no *host* para a porta 8080 das máquinas virtuais, utilizada pelo Tomcat.

1.4.2 Apache Tomcat

Um servlet é uma classe Java utilizada para gerar páginas HTML dinâmicas. Ele é responsável por receber chamadas HTTP, processá-las e devolver uma resposta.

O Tomcat, também desenvolvido pela Apache Software Foundation, é um container de servlets, ou seja, gerencia o ciclo de vida, dá suporte a multithread, segurança e páginas. O Tomcat tem a capacidade de rodar integradamente com um servidor HTTP Apache.

1.4.3 JavaServer Pages (JSP)

JavaServer Pages é uma tecnologia que permite a criação de páginas web dinamicamente. Com ela é possível acessar o banco de dados no servidor e exibir a página de acordo com seu conteúdo. Para poder executar JSPs é preciso usar um container de servlets, como o Tomcat.

1.5 VisionDataset

Como define o criador do sistema, Bruno Klava: "Este projeto tem por objetivo a criação de uma base de imagens para o grupo de pesquisa em visão computacional do IME/USP. Tal base de imagens servirá para centralizar num mesmo local as imagens utilizadas pelos membros do grupo, facilitando a utilização das mesmas imagens nos trabalhos desenvolvidos pelo grupo e permitindo a comparação de resultados de diferentes técnicas".

O VisionDataset é uma aplicação web colaborativa feita em Java e que utiliza o sistema gerenciador de banco de dados PostgreSQL, juntamente com o framework Apache Struts. Com ele é possível criar e gerenciar usuários, álbuns de imagens, fazer o upload de arquivos em massa, criar anotações e *tags* em imagens. Posteriormente, Rafael de Oliveira Lopes Gonçalves e o professor Roberto Hirata Jr. fizeram algumas melhorias, entre elas a integração com o sistema de segmentação de imagens SegmentIt.

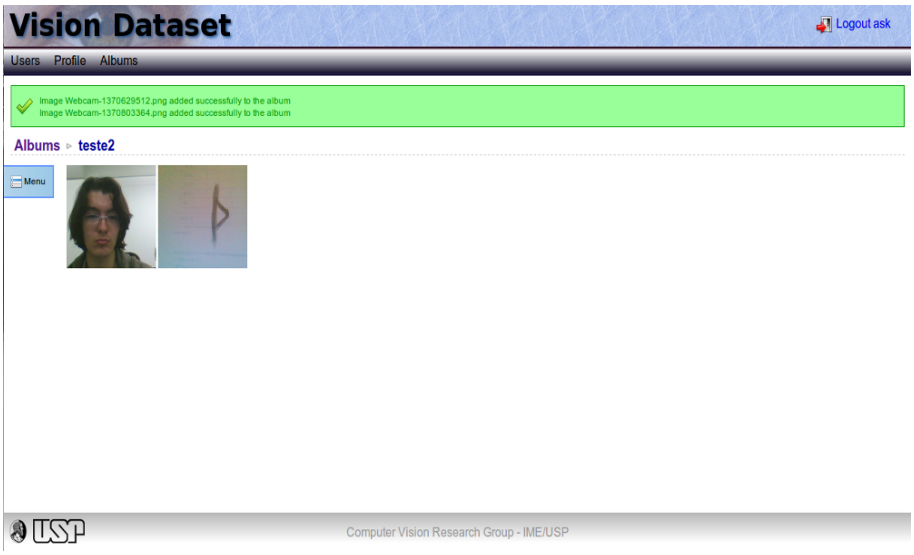

Figura 1 – Screenshot do VisionDataset

2 Atividades Realizadas

2.1 Pesquisa sobre ferramentas similares

No início do trabalho foram pesquisadas tecnologias semelhantes ao VisionDataset para saber se há alguma com o mesmo propósito ou com recursos que poderiam agregar alguma funcionalidade. Estas tecnologias são:

- AdessoWiki
  Uma plataforma colaborativa para programação e escrita de documentos, focados no processamento de imagens. Possui um método para execução de scripts em python muito eficiente, porém não funciona como um repositório de imagens.

- Harvard Dataverse Network
  É um repositório de dados dedicado à colaboração científica. Porém não é voltado a imagens e não possui um sistema de edição e execução de scripts.

Além disso, foram revistas algumas tecnologias que haviam sido pesquisadas previamente por Fábio Tsuguta Matsumoto e que na época não supriam as necessidades dos usuários do VisionDataset, como CAIMAN(CAncer IMage ANalysis), BASE (BioArray Software Environment), CARMEN (Code, Analysis, Repository and Modelling for e-Neuroscience), LabKey Server e MediGRID.

2.2 Configuração do ambiente

Para rodar e disponibilizar o VisionDataset no servidor da Rede eScience do IME-USP foi criada uma máquina virtual dedicada apenas à essa tarefa. Pudemos então instalar o Tomcat, o PostgreSQL (sistema gerenciador de banco de dados utilizado pelo VisionDataset) e outros pacotes usados pelo sistema. Com os pacotes instalados, o Maven foi configurado para usar as versões mais atuais das dependências. Tendo tudo configurado na máquina virtual, tivemos então que configurar o Apache no servidor web físico, para poder atribuir um endereço de acesso externo ao Tomcat da máquina virtual. Para tanto, foi feito um ProxyPass no Apache, adicionando-se as seguintes linhas ao arquivo de configuração na pasta *sites-available*:

```
ProxyPass /VisionDatasetdev http://192.168.231.207:8080/VisionDatasetdev
ProxyPassReverse /VisionDatasetdev http://192.168.231.207:8080/VisionDatasetdev
```

2.3 Correção no envio de e-mails

Tendo o sistema rodando no servidor, percebeu-se que ele não enviava e-mails para os usuários, o que não se pode ignorar, já que a criação de novos usuários depende da confirmação do e-mail destes. Pesquisando o erro, percebeu-se então que o sistema estava configurado para rodar no servidor antigo, utilizando o servidor de e-mails da própria Rede Vision. Foi feita então uma modificação no código para que ele enviasse os e-mails utilizando uma conta no Gmail (serviço de e-mails da Google) dedicada apenas para esta tarefa, esta conta pode também servir como um canal de comunicação entre o administrador do sistema e o usuário.

2.4 Criação da estrutura para salvar os scripts no banco de dados

As seguintes classes foram criadas:

```
br.usp.ime.vision.dataset.dao.impl.ScriptDAOImpl
br.usp.ime.vision.dataset.util.ScriptUtils
br.usp.ime.vision.dataset.servlet.ImagesServlet
br.usp.ime.vision.dataset.entities.ImageScript
br.usp.ime.vision.dataset.entities.Script
```

Elas contêm os atributos e métodos necessários para a inserção e recuperação dos scripts no banco de dados.

2.5 Scripts sendo guardados no banco de dados do sistema

Na página de imagens foi colocada uma caixa para inserção de texto, onde o usuário deve seguir as instruções para adicionar um script na linguagem Python, que fica atrelado à imagem. Ao salvar o script, ele pode ser acessado através do menu na própria imagem, neste menu podemos selecionar o script que queremos editar ou rodar. O usuário tem a opção de fazer o processamento apenas na imagem ou no álbum todo, além disso ele também pode escolher se quer que o resultado fique disponibilizado como um anexo, ou em um "subalbum"para ser visualizado no próprio sistema. Também é possível escolher se quer ser notificado via e-mail quando o processamento terminar.

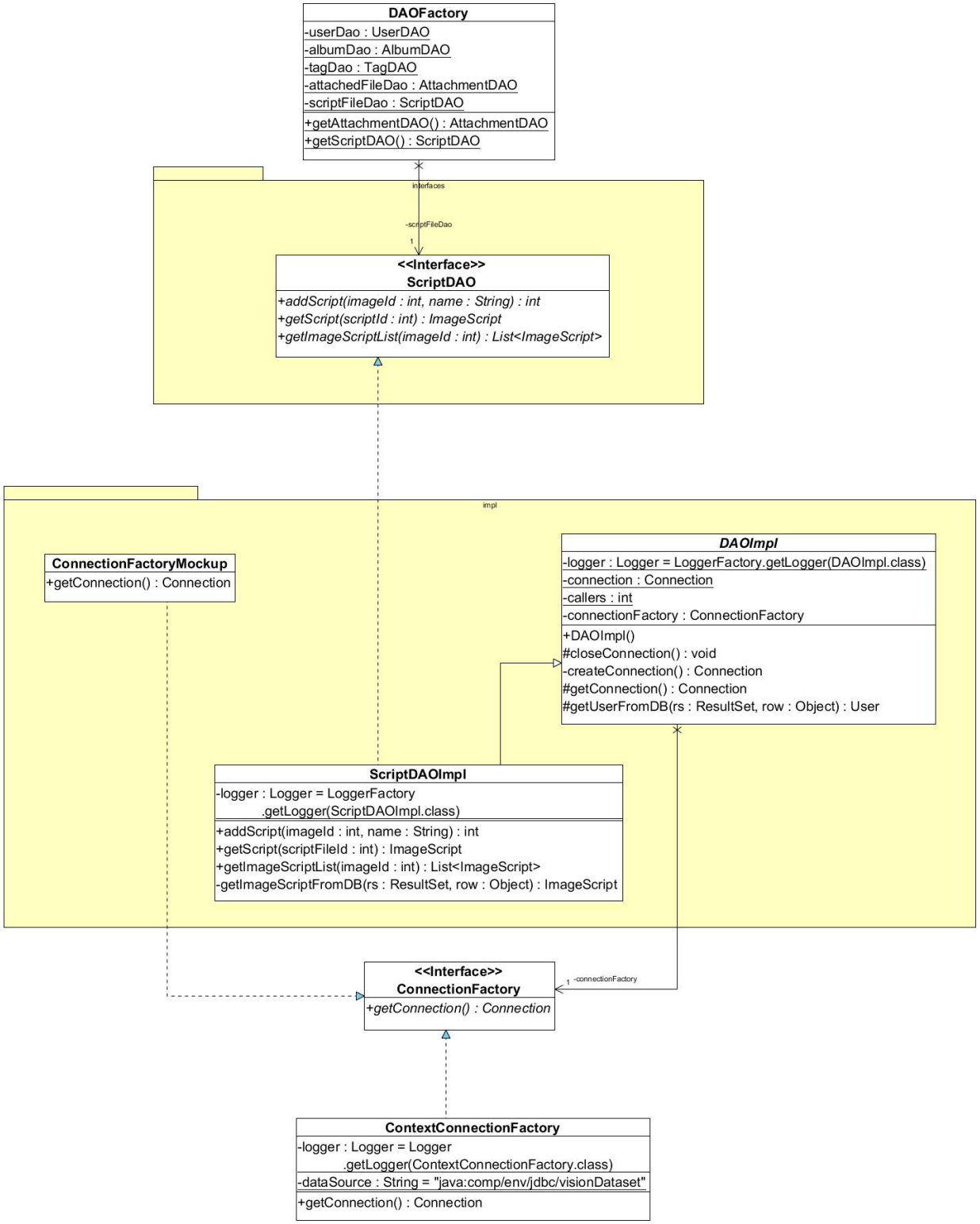

Figura 2 – Diagrama de classes da DAO dos scripts.

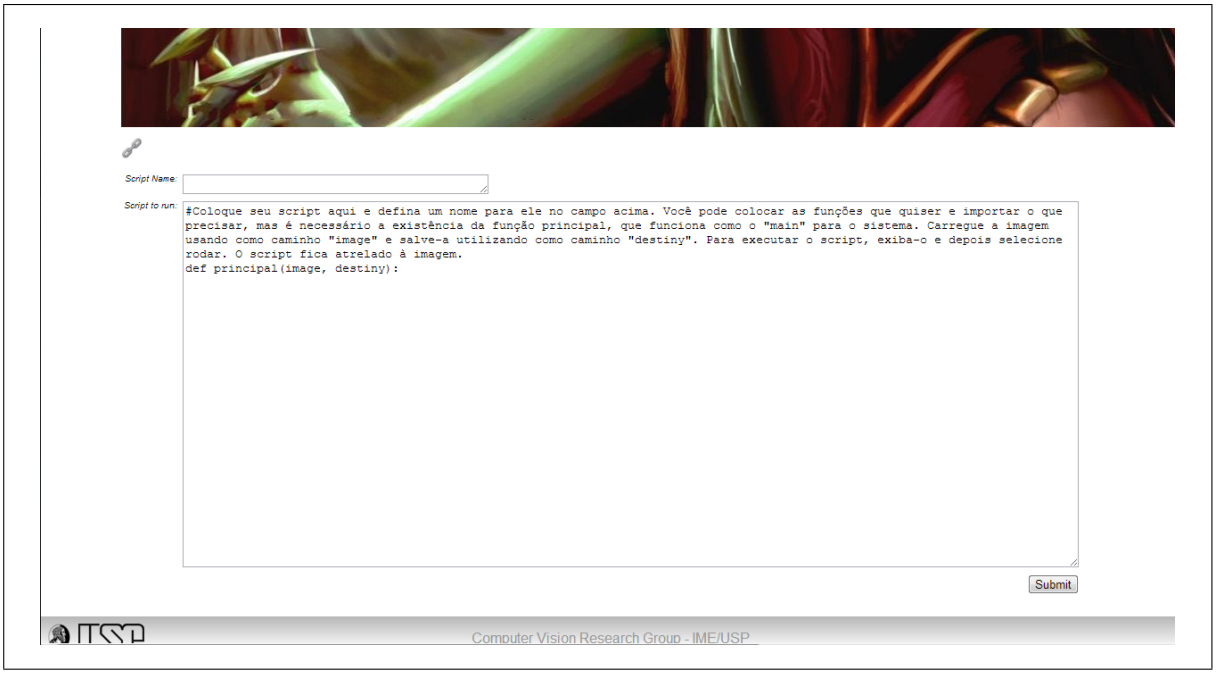

Figura 3 – Tela para inserção de scripts.

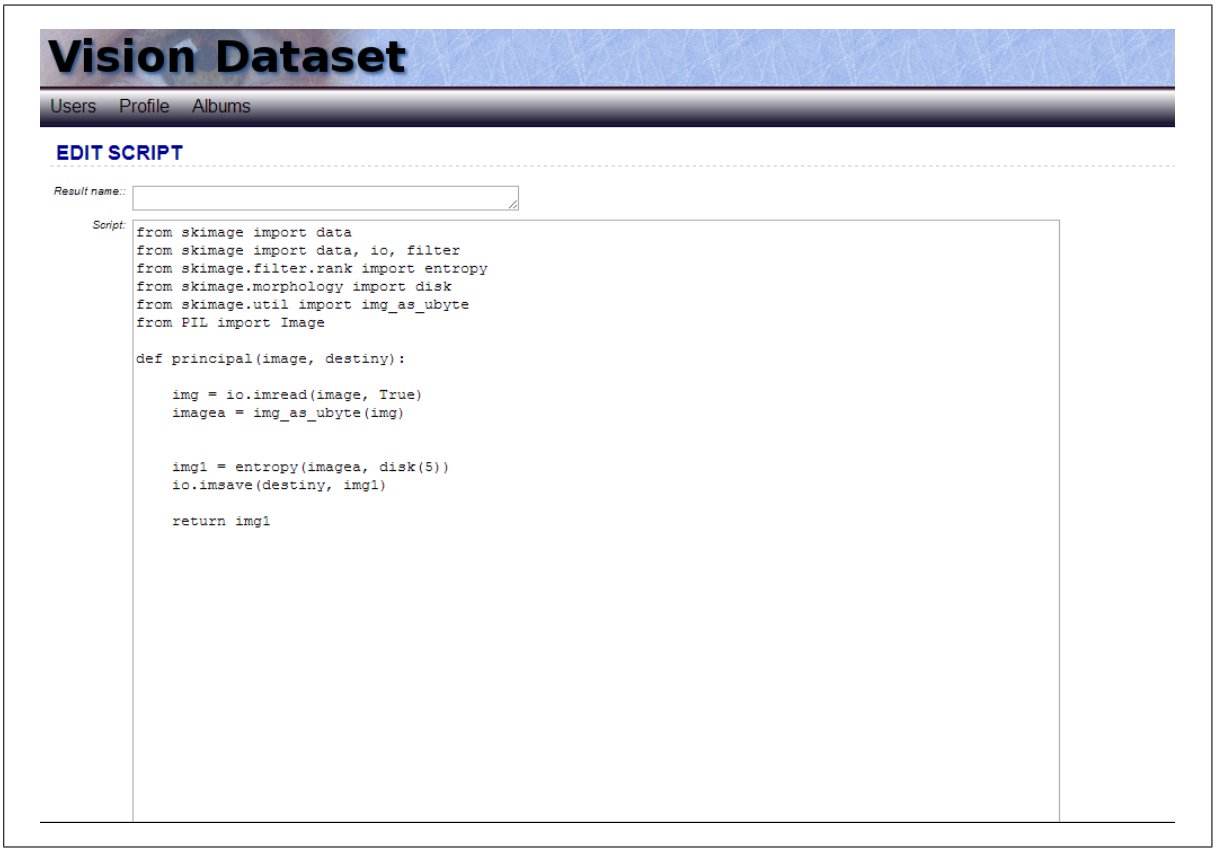

Figura 4 – Tela para edição de scripts.

Figura 5 – Opções de execução.

2.6 Método para a execução dos scripts

Ao executar um script, um gerenciador feito em Python é chamado. Para que ele consiga executar o script do usuário, é necessário que este contenha uma função chamada "principal", que recebe dois parâmetros: a entrada, de onde a imagem deve ser lida e a saída, que é onde a imagem deve ser salva. Desde que esse requisito seja cumprido, o usuário pode definir as funções que desejar e importar as bibliotecas que precisar e o script gerenciador conseguirá executá-lo.

2.7 Criação de daemons e scripts para comunicação entre máquinas

Para realizar a execução dos scripts de forma distribuída entre várias máquinas, foi criado um script gerenciador, que se comunica através de sockets com as máquinas virtuais de processamento, fazendo com que elas executem os scripts sobre as imagens. Para fazer com que as outras máquinas também tivessem acesso às imagens, foi utilizado o *sshfs*, que monta a pasta de imagens e scripts na máquina através do *ssh*. A seguir está uma descrição dos scripts e daemons criados, para esclarecer melhor como tudo funciona:

- *executor.py* - Script utilizado no servidor onde está hospedado o VisionDataset. Ele é responsável por delegar em qual máquina o processamento deve ser feito. A preferência é para a máquina que tiver mais memória disponível, já que nos testes notou-se que a memória é o principal limitante de desempenho. Mais detalhes sobre a técnica estão no capítulo de testes de resultados. Para cada usuário que solicita o processamento de uma imagem, existe uma instância diferente do *executor*.

- *execvm.py* - Script que é acionado nas máquinas virtuais para realizar o processamento sobre a imagem desejada, utilizando o script escolhido pelo usuário.

- *execvdsdaemon* - Um daemon que fica rodando nas máquinas virtuais, esperando instruções de como deve ser executado o processamento nas imagens, também cria diferentes threads para cada imagem.

- *machinestatusdaemon* - Daemon que roda nas máquinas virtuais e quando é acionado devolve a quantidade de memória disponível na máquina, assim o gerenciador *executor.py* pode escolher em qual delas fará o processamento.

- *preparamaquina.sh* - Quando uma máquina é resetada, este script deve ser executado para que ela funcione como uma máquina de processamento (pode-se colocá-lo para executar durante o boot do sistema).

- *packages* - Script que deve ser executado quando deseja-se instalar todos os pacotes necessários para que a máquina consiga rodar os scripts, mantendo assim uma certa homogeneidade entre elas.

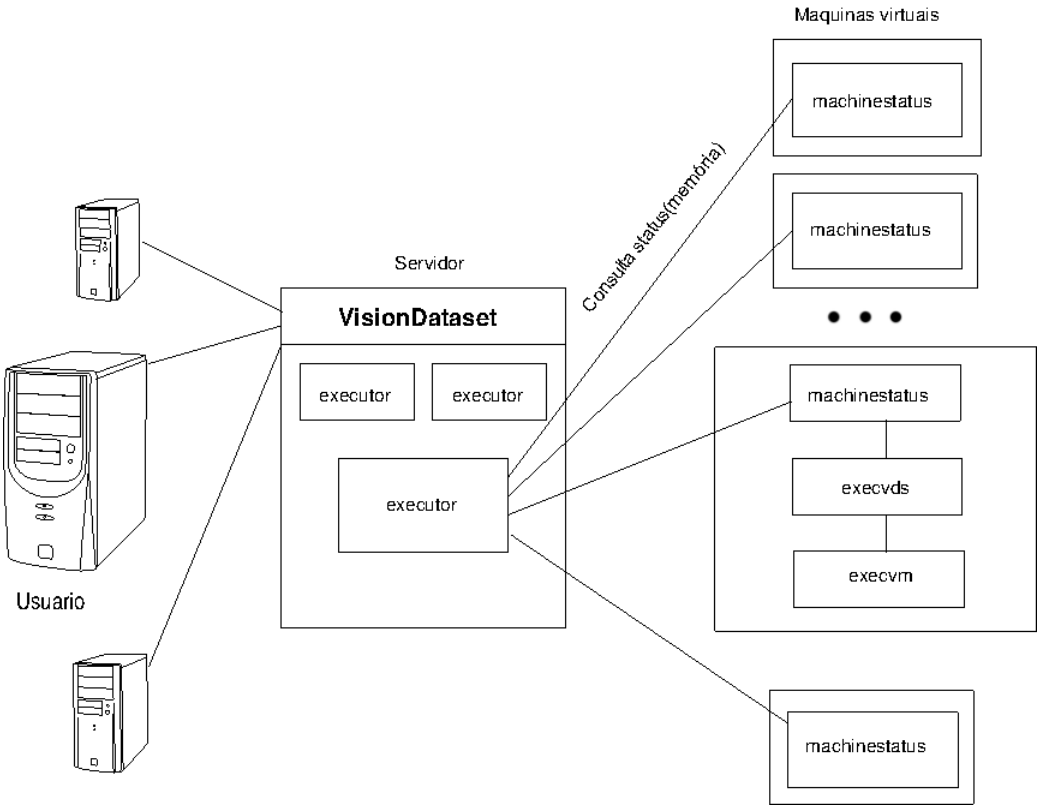

Figura 6 – Esquema com o funcionamento dos scripts.

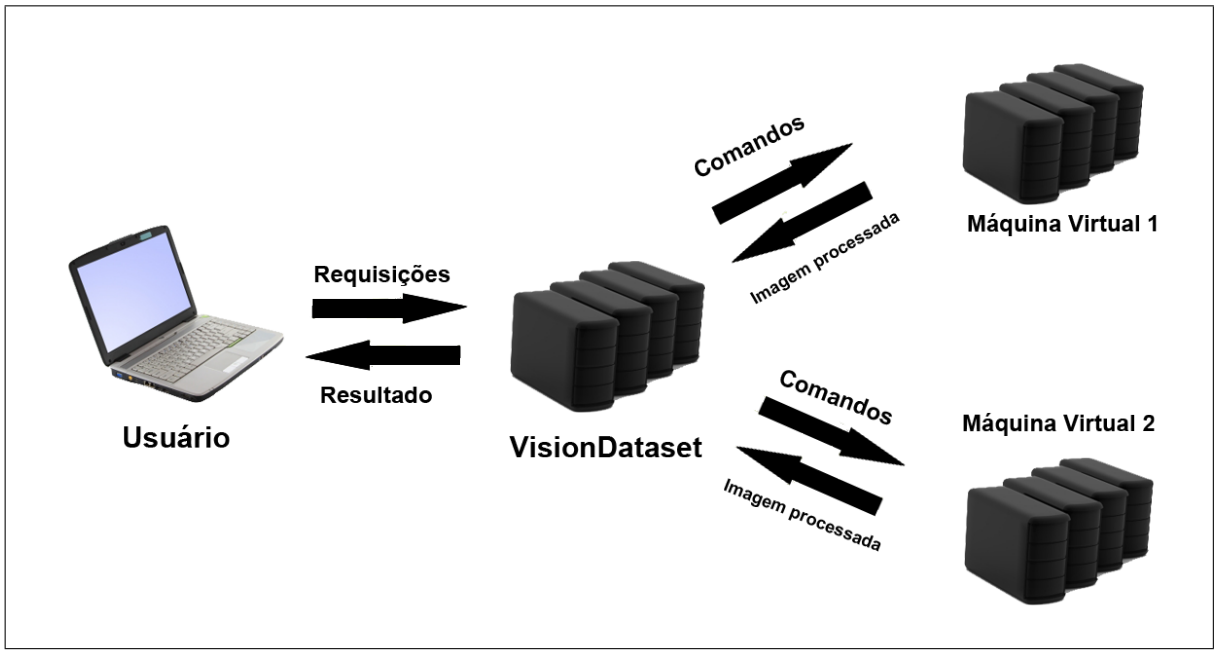

Figura 7 – Esquema que mostra o funcionamento da distribuição da carga.

3 Testes e resultados

Os testes foram realizados em máquinas virtuais que utilizam 4 GB de memória e 4 núcleos de um processador Intel Xeon de 2.4GHz. Essas máquinas ficam nos servidores da Rede eScience no IME-USP.

3.1 Entropia de 14 imagens de tamanhos variados em ordem crescente

Os primeiros testes realizados para testar a eficiência utilizaram a biblioteca scikit-image (http://scikit-image.org/) para encontrar a entropia de 14 imagens, com tamanhos variando de 1 a 15 megapixels.

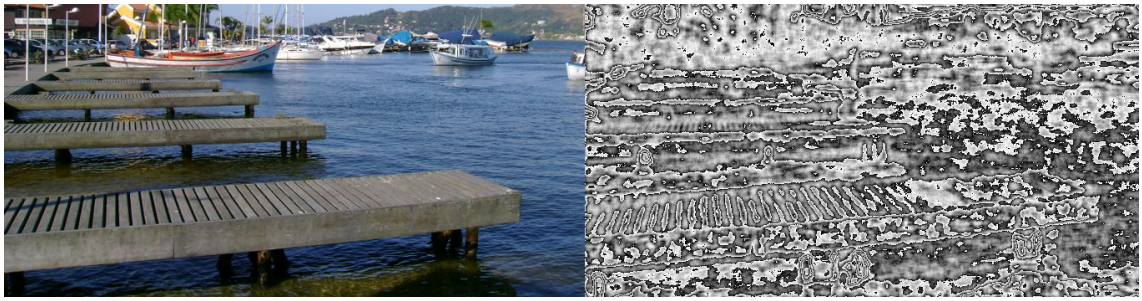

Figura 8 – Imagem antes e depois do algoritmo de entropia.

No primeiro gráfico temos os resultados obtidos utilizando a técnica de distribuição sequencial entre as máquinas:

Na primeira barra do primeiro gráfico temos a execução sendo feita no próprio servidor do VisionDataset, da forma mais simples possível, sequencialmente. Na segunda barra temos a execução em uma máquina virtual apenas, nela podemos observar um aumento no tempo mesmo realizando o processamento das imagens paralelamente, isso se deve ao fato de a memória da máquina ter esgotado e o *swap* começar a ser usado, o que resulta em uma drástica queda de eficiência. Na terceira barra, utilizando 2 máquinas virtuais, houve um a redução de quase 6 vezes no tempo necessário para processar todas as imagens, isso se deve ao fato de a memória não ter esgotado e todos os núcleos dos processadores estarem sendo utilizados. Na última barra, com 3 máquinas não temos tanto ganho porque as imagens menores foram processadas muito rapidamente e com 2 máquinas já era possível ter um núcleo para cada imagem cujo tempo de processamento era muito grande.o se

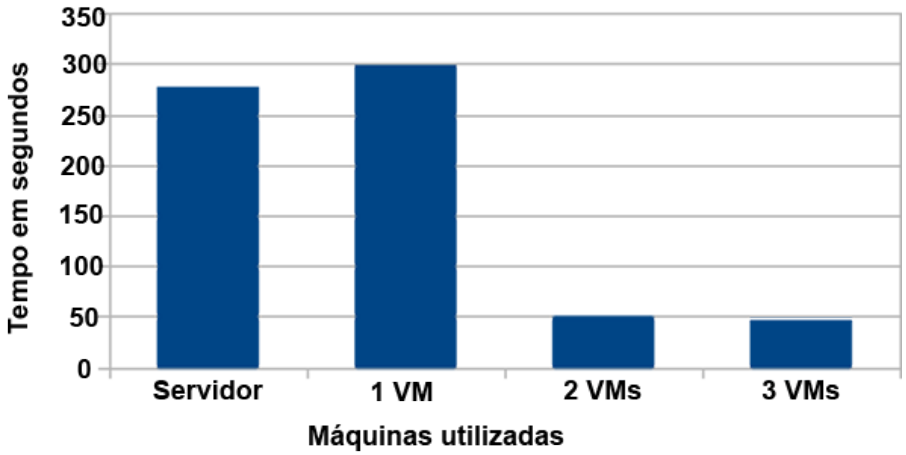

Figura 9 – Gráfico do número de máquinas pelo tempo, delegando o processamento sequencialmente.

No segundo, temos os resultados da técnica que atribui o processamento à máquina com mais memória disponível:

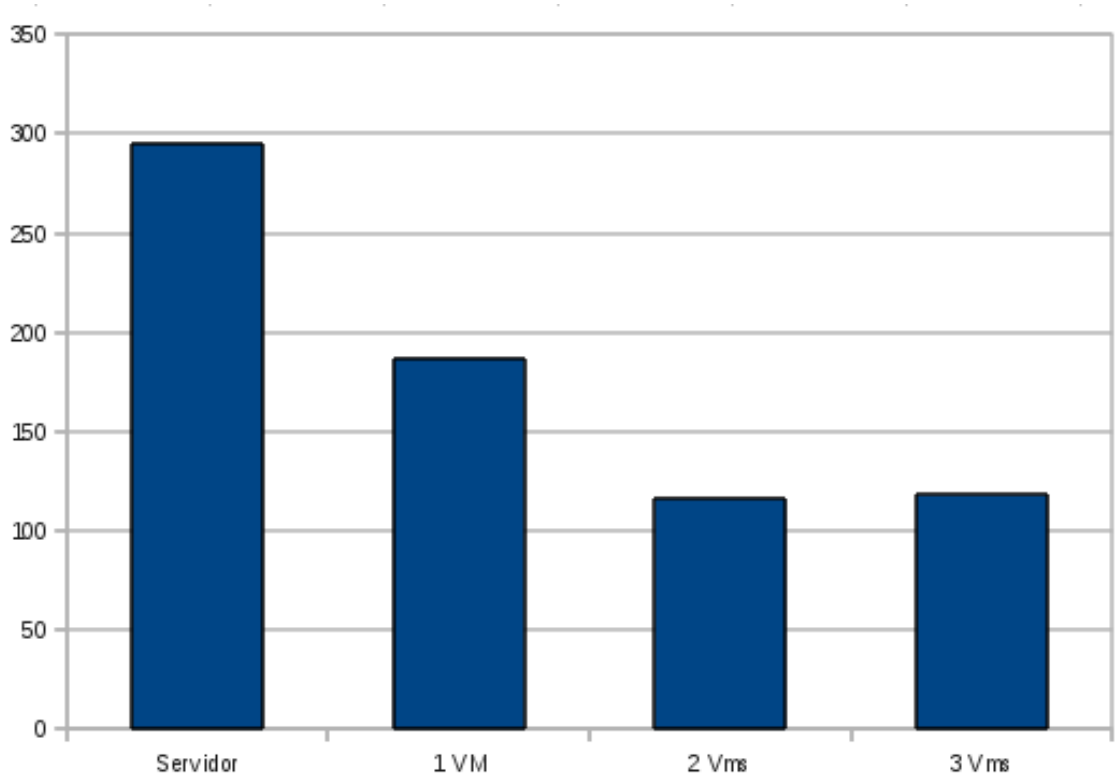

Figura 10 – Gráfico do número de máquinas pelo tempo, com o processamento sendo delegado à máquina com mais memória.

Podemos notar que a tendência de melhora na eficiência é parecida no segundo gráfico. Obtivemos um melhor resultado com uma máquina virtual apenas, já que existe um intervalo entre a delegação de processos de 3 segundos, esse tempo foi suficiente para

que alguma imagens menores fossem processadas e liberassem a memória. É também devido a este atraso que o desempenho do processamento realizado no servidor e com 2 ou mais máquinas virtuais ficou prejudicado. Como o tamanho das imagens é crescente, no método sequencial o processamento foi distribuído de uma forma boa.

3.2 Entropia de 14 imagens de dois tamanhos diferentes em ordem aleatória

Neste teste o número de imagens foi o mesmo, porém foram usadas 7 imagens de de 15 megapixels e 7 de 1 megapixel.

No primeiro gráfico temos os resultados da técnica que atribui o processamento à máquina com mais memória disponível:

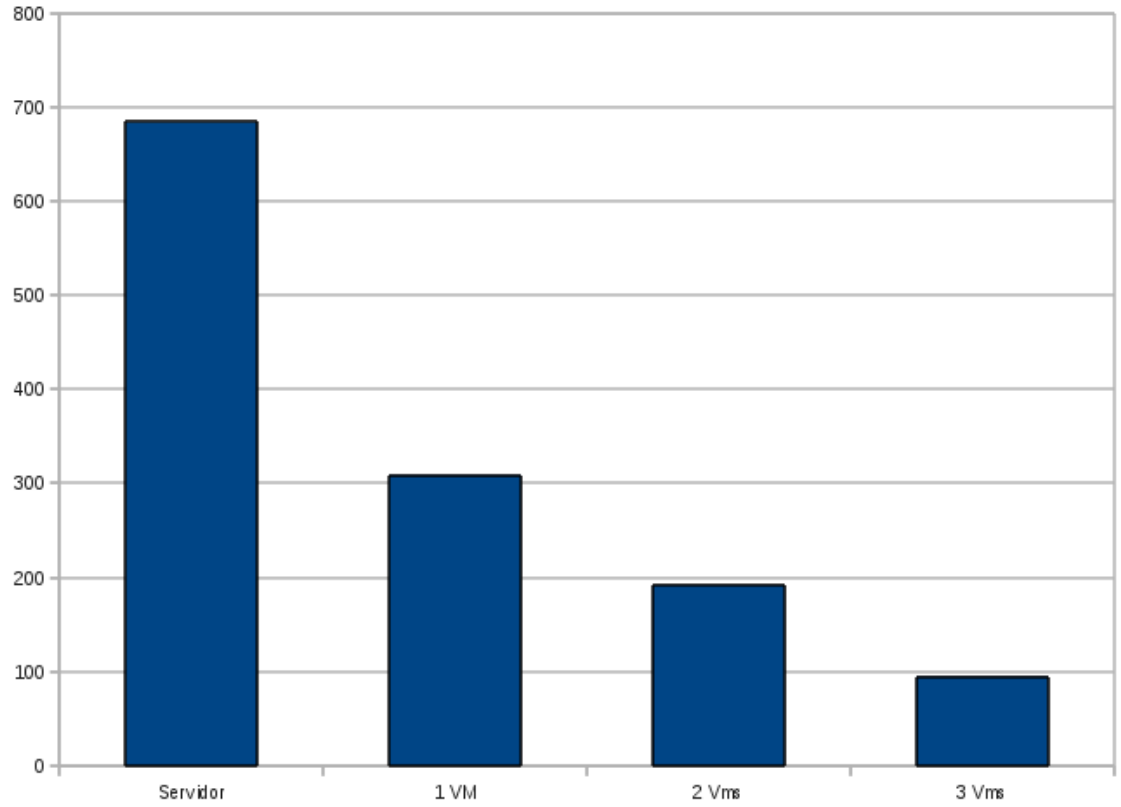

Figura 11 – Gráfico do número de máquinas pelo tempo, com o processamento sendo delegado sequencialmente.

Neste gráfico temos um resultado parecido com o outro, mas vemos um resultado muito melhor à partir de 1 máquina virtual, já que o consumo de memória foi menor.

No próximo gráfico temos os resultados obtidos delegando o processamento para

a máquia com mais memória:

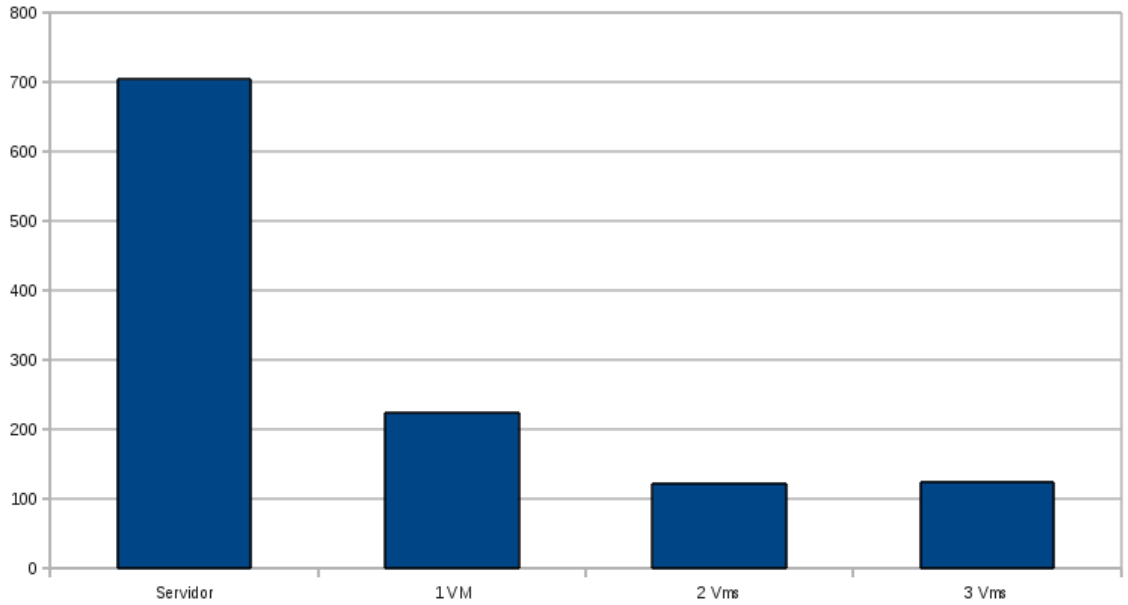

Figura 12 – Gráfico do número de máquinas pelo tempo, com o processamento sendo delegado à máquina com mais memória disponível.

Aqui podemos observar que o melhor desempenho no caso de duas máquinas virtuais é melhor do que o sequencial, isso aconteceu devido à melhor distribuição dos processamentos entre as imagens.

Conclusão

Executar scripts sobre as imagens dentro do próprio sistema VisionDataset facilita muito a colaboração, além de economizar muito o tempo dos usuários. Fazer tal execução de forma eficiente e segura, de forma que o usuário não quebre o sistema executando scripts errados, ao mesmo tempo que possui a liberdade para utilizar os recursos que precisar mostrou-se um bom desafio.

Não basta apenas que o sistema seja capaz de executar os scripts, é necessário também que o usuário consiga usá-los. Portanto, ao mesmo tempo em que preocupa-se em realizar as tarefas da forma mais eficiente possível, é necessário prover formas para que um usuário comum, que não trabalhou no desenvolvimento do sistema, consiga utilizá-lo da melhor forma possível.

Observando-se os resultado obtidos, pode-se chegar a algumas conclusões sobre vantagens e desvantagens de se distribuir o processamento entre máquinas virtuais. As desvantagens são a necessidade de utilizar a rede para realizar o acesso aos arquivos e a maior dificuldade para o administrador do sistema gerenciar todos os recursos. Porém, utilizando as máquinas virtuais, em todos os casos notou-se um excelente ganho no desempenho do processamento realizado, além de termos mais segurança. Quando usou-se as máquinas virtuais para realizar os testes não houve nenhuma alteração no desempenho do sistema.

Quanto às técnicas para a distribuição de carga, decidiu-se que delegar o processamento das imagens utilizando a memória como parâmetro de decisão é a melhor opção por enquanto, já que evita o uso do *swap* que mostrou-se um grande limitador de desempenho, além de garantir um razoável uso de todas as máquinas, sem sobrecarregar apenas algumas.

Referências Bibliográficas

- KLAVA, B. *Base de imagens do grupo de Visão Computacional do IME/USP*. 2010. 3 p. - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo.
- MATSUMOTO, F. T. *Ferramentas semelhantes ao VisionDataset*. 2012. 6 p. - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo.
- *Máquina Virtual*s. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2013. Disponível em: <http://pt.wikipedia.org/w/index.php?title=Máquina_virtual&oldid=36412439>. Acesso em: 20 ago. 2013.
- *Servidor Web*. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2005. Disponível em: <http://pt.wikipedia.org/w/index.php?title=Servidor_Web&oldid=1110359>. Acesso em: 20 ago. 2013.
- *Servidor Apache*. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2013. Disponível em: <http://pt.wikipedia.org/w/index.php?title=Servidor_Apache&oldid=36050277>. Acesso em: 2 set. 2013.
- LOTUFO, R. A.; MACHADO R. C. *Adessowiki*. 2009. Disponível em: <http://adessowiki.fee.unicamp.br/adesso/wiki/main/aw_slides/view>. Acesso em: 21 ago. 2013.
- *The Dataverse Network Project*. 2013. Disponível em: <http://thedata.org>. Acesso em: 21 ago. 2013
- Seshadri. G. *Understanding JavaServer Pages Model 2 architecture*. 1999. Disponível em: <http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html>. Acesso em 12 set. 2013.
- *JavaServer Pages*. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2013. Disponível em: <http://pt.wikipedia.org/w/index.php?title=JavaServer_Pages&oldid=36724950>. Acesso em: 12 set. 2013.
- *About PostgreSQL*, 2013. Disponível em: <http://www.postgresql.org/about/>. Acesso em: 12 set. 2013.
- *What is Maven?*, 2013. Disponível em: <http://maven.apache.org/what-is-maven.html>. Acesso em: 12 set. 2013.
- *socket — Low-level networking interface*, 2013. Disponível em: <http://docs.python.org/2/library/socket.html>.Acesso em: 25 set. 2013.

- Imagens da figura 7 cordialmente cedidas por SweetCrisis, Keerati e renjith krishnan/ freedigitalphotos.net

Parte II

Subjetiva

4 Desafios e frustrações

Quando o professor Roberto Hirata Jr. me mostrou o VisionDataset eu logo achei muito interessante e quis fazer meu trabalho baseado nele. A primeira grande dificuldade foi entender o funcionamento do sistema, que já era razoavelmente grande quando eu comecei. Nele é usando um framework que eu nunca havia visto e eu precisava adquirir mais conhecimentos de bancos de dados para conseguir compreender como tudo funcionava. Perdi praticamente o primeiro semestre inteiro nisso, além disso peguei muitas matérias e acabei me enrolando um pouco.

No segundo semestre, peguei matérias que me ajudariam a entender e desenvolver para o VisionDataset, como Redes e Laboratório de Banco de Dados. Tendo então o domínio de como o sistema funcionava, pude então começar a desenvolver com uma velocidade muito maior. Fiz a interface para execução de scripts, de edição e depois comecei a rodar os testes em servidores e a lidar com máquinas virtuais (a área que eu realmente gosto e sempre trabalhei). Estava tendo muitos avanços, foi quando acabei entrando em um processo seletivo de estágio em uma grande empresa, isso me tomou um certo tempo. Mas passado o processo, pude me dedicar mais ao trabalho e obter resultados muito promissores na distribuição do processamento.

Acredito que o principal desafio foi desenvolver sobre um sistema já pronto. Mesmo com toda a documentação e a ajuda do meu supervisor e dos alunos que desenvolveram, muitas vezes perdi muito tempo por não saber que algo já estava implementado ou onde colocar certa funcionalidade.

5 Disciplinas relevantes

Muitas disciplinas foram importantes para o desenvolvimento do meu trabalho. Mas algumas foram essenciais, e são as que destaco aqui.

5.1 MAC0242 - Laboratório de Programação II

Aprendi os fundamentos de orientação a objetos em linguagem Java, que foram essenciais para entender o VisionDataset.

5.2 MAC0426 - Sistemas de Bancos de Dados e MAC0439 - Laboratório de Bancos de Dados

Os conceitos de bancos de dados, linguagem SQL, como fazer consultas mais sofisticadas, esses conhecimentos foram essenciais para que eu pudesse gravar os scripts e os resultados no banco de dados usando o framework Java Struts.

5.3 MAC0417 - Visão e Processamento de Imagens

Apesar de não se tratar de um trabalho em que eu desenvolva um algoritmo para processamento de imagens, meu sistema é feito para isso. Então eu preciso entender as necessidades de quem deseja realizar tal processamento e também saber algoritmos para fazer os testes.

5.4 MAC0438 - Programação Concorrente

No meu sistema trabalhei com threads para processar várias imagens ao mesmo tempo. Este conhecimento adquiri nesta matéria.

5.5 MAC0448 - Programação para Redes de Computadores

Precisei dos conhecimentos adquiridos nesta matéria para realizar a comunicação entre o servidor do VisionDataset e as máquinas virtuais de processamento.

6 Próximos passos

Ainda há tarefas que podem ser realizadas sobre a execução de scripts no VisionDataset. Neste capítulo elas serão listadas.

6.1 Melhorar a interface para edição de scripts

Seria interessante colocar um editor de scripts em que houvesse destaque dos termos e correção de erros de código antes do envio para a máquina virtual. No momento o editor é apenas uma caixa de texto.

6.2 Melhorar o tratamento de erros

O usuário precisa ter certeza que o script está funcionando antes de mandá-lo rodar, ou então ele saberá do erro apenas quando descobrir que o resultado não foi gerado. Seria bom colocar algum tipo de tratamento de exceção que informasse o usuário de alguma forma que o processamento falhou.

6.3 Procurar e corrigir falhas de segurança

O fato de o processamento ser feito em máquinas virtuais já evita que usuários quebrem o sistema, mas ainda é preciso verificar até onde ele consegue chegar tendo a liberdade de executar scripts nas máquinas virtuais da maneira que desejar.

6.4 Melhorar eficiência e diferenciar usuários

Do jeito que está sendo feito, garante-se um bom uso das máquinas de forma eficiente, mas podem haver outros jeitos que consigam otimizar ainda mais, pode ser que um estudo mais aprofundado encontre essas maneiras, como dividir a imagem em várias partes e distribuir essas partes entre várias máquinas. Também seria interessante que alguns usuários tenham privilégios a mais e possam rodar seus scripts em máquinas que os outros não tem acesso.

7 Agradecimentos

Primeiramente gostaria de agradecer o meu supervisor, Prof. Dr. Roberto Hirata Jr., por sempre ter dado toda ajuda que precisei desde o final de 2012 e sempre ter se preocupado com o andamento do meu trabalho e com o meu aprendizado. Ele também deixou que eu usasse as excelentes máquinas da rede eScience para hospedar as minhas máquinas virtuais.

Gostaria também de agradecer a David da Silva Pires, administrador da Rede eScience, pois sempre que eu criava uma nova máquina precisava dele para adicionar o nome no servidor de DNS e para resolver problemas de infraestrutura.

Aos meus amigos, que estavam na mesma situação que eu e tornaram esta jornada muito mais fácil e divertida.

À minha família, que sempre me apoiou em todos os momentos da minha vida.