**DataViva**

Ambiente de Produção

V0.1

Núcleo de Sistemas e Gestão

Fevereiro 2013

Histórico de Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| DATA | VERSÃO | DESCRIÇÃO | AUTOR |
| 29/07/2013 | 1.0 | Criação do documento | Mario H. Mol |
| 31/10/2013 | 1.0 | Versão base | Mario H. Mol |

ÍNDICE

[1. INTRODUÇÃO 4](#_Toc371670101)

[2. SERVIDORES 5](#_Toc371670102)

[3. BANCO DE DADOS 6](#_Toc371670103)

[3.1. BASES DE DADOS 7](#_Toc371670104)

[4. CACHE 8](#_Toc371670105)

[5. APLICAÇÃO 9](#_Toc371670106)

[5.1. Servidor Web 10](#_Toc371670107)

[5.2. Integração Servidor Web com Python 12](#_Toc371670108)

[5.2.1. Outros Métodos 13](#_Toc371670109)

[5.2.2. NGINX 14](#_Toc371670110)

[5.3. Configurações de Ambiente 16](#_Toc371670111)

[6. PERFORMANCE 17](#_Toc371670112)

[6.1. Load Balance 18](#_Toc371670113)

INTRODUÇÃO

Está sendo implantado em Minas Gerais, em parceria com a *Growth Ventures (GV)* e o Governo do Estado de Minas Gerais, o Project Space (*PS*). Para saber mais sobre o objetivo deste projeto e outras informações veja o documento “Visão – DataViva”.

Este projeto irá disponibilizar informações que servem como ferramenta de análise para diversos públicos, tanto do meio público como privado. Sendo assim o projeto foi construído como um sistema Web, a ser acessado por qualquer pessoa através da internet.

Para viabilizar que todo cidadão possa acessar é necessário que este sistema seja instalado em uma infraestrutura de computadores, em um ambiente controlado, com acesso a internet, segurança, refrigeração e equipe especializada para eventuais manutenções. Este local a ser instalado é chamado de Data Center e, pelo fato de ser o local onde fica o site para acesso ao público, chamamos de ambiente de produção.

Este projeto é composto por um grande banco de dados, um sistema de cache e a aplicação que gera as páginas do sistema. O sistema de cache contém meios para armazenar as informações mais visualizadas pelos usuários de forma a acelerar o acesso a informação, aumentando muito a performance de acesso. Este tipo de recurso é necessário principalmente para este projeto, que contém diversos cálculos e muita informação.

No caso temos duas máquinas disponíveis para hospedar esta aplicação, uma com mais recursos computacionais onde está instalado o banco de dados e o sistema de cache, e outro mais leve com a aplicação.

O objetivo deste documento é descrever toda a infraestrutura utilizada para hospedar este projeto, servindo também de guia de instalação e manutenção. Sendo assim um profissional de Tecnologia da Informação (*TI*) poderá utilizar este documento para instalar novamente o *PS* nas maquinas que estão hospedando o sistema ou até mesmo em novas máquinas. Este mesmo conhecimento é necessário para que uma equipe de *TI* possa dar manutenção nestes servidores.

No futuro este ambiente de produção será substituído por outro, já que esta tecnologia será transferida para uma Secretaria do Estado, já que o Escritório de prioridades tem um prazo para terminar suas atividades. Sendo assim com este documento a nova equipe que receberá *PS* poderá então realizar a instalação em uma máquina na sua própria infraestrutura.

SERVIDORES

Temos à disposição duas máquinas a serem utilizadas para o ambiente de produção do PS, uma máquina bem robusta em que temos instalado os recursos mais pesado do sistema e outra mais simples onde temos a aplicação que é a parte mais leve.

O servidor mais robusto é uma máquina com o *Sistema Operacional (OS)* *Windows*, onde temos instalado o *Qlikview* utilizado pela Secretaria de Defesa Social do Estado de Minas Gerais e que está com recursos sobrando, pois o volume de usuários da defesa que acessam o *Qlikview* é bem abaixo da capacidade da máquina. As configurações desta máquina são:

* Nome: VM13525
  + Sistema Operacional: Windows 2008 R2 Standard 64 Bits
  + Memória RAM: 24 Gb
  + Disco: 500 Gb
  + IP: 200.198.9.184

O banco de dados utilizado é o *Mysql 5.5* e Está sendo implantado em Minas Gerais, em parceria com a *GV* e o Governo do Estado.

* Nome: VM13491
  + Finalidade: ESCRITORIO - Servidor WEB PHP
  + Sistema Operacional: Linux RedHat 6.1 64 Bits
  + IP: 200.198.9.180
  + Memória: 4 Gb
  + Disco: 30 GB
  + Tecnologias: Apache 2.2.15 - PHP 5.2.17
  + Processador: 2 núcleos - AMD Opteron(tm) Processor 6176 SE

Para instalação de todas as bibliotecas necessárias nesta aplicação vamos utilizar a ferramenta integrada ao Red Hat, chamada através Gerenciador de Instalações do SO. do comando “yum”, muito simples de utilizar. Existem alguns casos isolados onde teremos que instalar bibliotecas manualmente, ou seja, sem utilizar o Yum.

BANCO DE DADOS

O Sistema de Gerenciamento de Banco de Dados (SGBD) utilizado neste projeto é o *MySQL Community* *5.6* e a versão deve ser para *Windows 64 Bits*, afinal o servidor escolhido para ser o banco de dados é o VM13525, que tem este sistema operacional. A instalação do *MySQL* pode ser encontrado através do link:

* [www.mysql.com](http://www.mysql.com)
  + O instalador utilizado foi o: mysql-installer-community-5.6.10.0.msi

A sua instalação deve ser realizada de forma padrão, mantendo todas as configurações de fábrica e o diretório escolhido para instalação no servidor foi:

* C:\Program Files (x86)\MySQL

Desta forma a aplicação que quiser conectar ao *MySQL* deverá utilizar o IP deste servidor e a porta padrão de conexão com o *MySQL*, a porta 3306.

No caso só deverá haver conexão com o banco de dados através da máquina de aplicação, a *VM13491*. Como elas estão na mesma rede e sob o mesmo contrato, foi pedido a liberação de qualquer conexão entre estas máquinas, por isso não foi necessário requisitar a liberação da porta 3306 entre estes servidores.

Para a gestão do *SGBD*, das bases de dados e demais operações envolvendo o banco de dados utilizamos o aplicativo da própria fabricante do banco, o *MySQL Workbench 5.2 CE*, encontrado no mesmo endereço onde tem o instalador do *MySQL*.

Temos apenas uma particularidade em relação a instalação padrão, foi necessário alterar o diretório aonde o *MySQL* armazena as bases de dados. Isto porque o servidor *VM13525* tem uma partição pequena no *C*, aonde normalmente é instalado o sistema de banco de dados, e as bases de dados deste projeto são enormes. Portanto tivemos que mudar o local aonde as bases de dados são salvas para o diretório:

* E:\BancoDados\Mysql

Para realizar esta alteração é necessário acessar a pasta aonde os bancos de dados são instalados por padrão e copiar para a pasta na partição *E*, citada acima.

* C:\Program Files\MySQL\MySQL Server 5.6\data

A finalização desta configuração se dá pela alteração da pasta das bases de dados que fica no arquivo de configuração do *MySQL*, chamado my.ini e que fica localizado em:

* C:\ProgramData\MySQL\MySQL Server 5.6\my.ini

Ao abrir este arquivo com o editor de texto existe uma variável chamada basedir edeve ser alterada da seguinte forma:

* datadir="E:\BancoDados\Mysql\"

Para que esta alteração passe a valer basta reiniciar o MySQL, para isso abra a tela de serviços do Windows, encontrar o serviço chamado MySQL56 e então reiniciar.

* 1. BASES DE DADOS

Existem diversas tabelas que compõem o banco de dados, mas elas são normalmente agrupados em dois arquivos:

* DataMinas.sql.bz2
* mte\_tables.sql.bz2

Estas duas bases juntas e compactadas ocupam em torno de 1 Giga, mas após a sua descompactação e instalação elas passam a ocupar 20 Gigas. A expectativa é que a base completa, no final do projeto, deverá ocupar 70 Gigas.

A inclusão destas bases de dados no servidor MySQL é bem demorada, que está relacionado como seu tamanho. Para realizar esta operação, execute no terminal do Windows os comandos abaixo.

Base de dados da RAIS:

* mysql -uroot -p123456 ps < rais\_yb.sql
* mysql -uroot -p123456 ps < rais\_ybi.sql
* mysql -uroot -p123456 ps < rais\_ybio.sql
* mysql -uroot -p123456 ps < rais\_ybo.sql
* mysql -uroot -p123456 ps < rais\_yi.sql
* mysql -uroot -p123456 ps < rais\_yio.sql
* mysql -uroot -p123456 ps < rais\_yo.sql
* mysql -uroot -p123456 ps < secex\_yb.sql
* mysql -uroot -p123456 ps < secex\_ybp.sql
* mysql -uroot -p123456 ps < secex\_ybpw.sql
* mysql -uroot -p123456 ps < secex\_ybw.sql
* mysql -uroot -p123456 ps < secex\_yp.sql
* mysql -uroot -p123456 ps < secex\_ypw.sql
* mysql -uroot -p123456 ps < secex\_yw.sql
* mysql -uroot -p123456 ps < comtrade\_ypp.sql
* mysql -uroot -p123456 ps < comtrade\_ypw.sql
* mysql -uroot -p123456 ps < apps\_plan\_build.sql
* mysql -uroot -p123456 ps < apps\_plan\_title.sql
* mysql -uroot -p123456 ps < apps\_ui.sql
* mysql -uroot -p123456 ps < ask\_question.sql
* mysql -uroot -p123456 ps < ask\_reply.sql
* mysql -uroot -p123456 ps < ask\_reply\_flag.sql
* mysql -uroot -p123456 ps < ask\_status.sql
* mysql -uroot -p123456 ps < ask\_tag.sql
* mysql -uroot -p123456 ps < ask\_vote.sql
* mysql -uroot -p123456 ps < account\_starred.sql
* mysql -uroot -p123456 ps < account\_user.sql
* mysql -uroot -p123456 ps < apps\_app.sql
* mysql -uroot -p123456 ps < apps\_build.sql
* mysql -uroot -p123456 ps < apps\_build\_ui.sql
* mysql -uroot -p123456 ps < apps\_plan.sql
* mysql -uroot -p123456 ps < attrs\_bb.sql
* mysql -uroot -p123456 ps < attrs\_bra.sql
* mysql -uroot -p123456 ps < attrs\_bra\_pr.sql
* mysql -uroot -p123456 ps < attrs\_cbo.sql
* mysql -uroot -p123456 ps < attrs\_hs.sql
* mysql -uroot -p123456 ps < attrs\_isic.sql
* mysql -uroot -p123456 ps < attrs\_wld.sql
* mysql -uroot -p123456 ps < attrs\_yb.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2002.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2003.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2004.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2005.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2006.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2007.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2008.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2009.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2010.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2011.sql
* mysql -uroot -pmysql ps\_launch < rais\_ybio\_2012.sql

Aqui estão os comandos utilizados para atualizar as tabelas que sofrem alterações por interações com o usuário:

* mysql -uroot -pmysql ps\_launch < account\_starred.sql
* mysql -uroot -pmysql ps\_launch < account\_user.sql
* mysql -uroot -pmysql ps\_launch < ask\_question.sql
* mysql -uroot -pmysql ps\_launch < ask\_question\_tags.sql
* mysql -uroot -pmysql ps\_launch < ask\_reply.sql
* mysql -uroot -pmysql ps\_launch < ask\_reply\_flag.sql
* mysql -uroot -pmysql ps\_launch < ask\_status.sql
* mysql -uroot -pmysql ps\_launch < ask\_tag.sql
* mysql -uroot -pmysql ps\_launch < ask\_type.sql
* mysql -uroot -pmysql ps\_launch < ask\_vote.sql

CACHE

O sistema de cache utilizado no PS é o Redis e existe uma explicação mais detalhada da forma como ele foi utilizado no documento “Guia de Desenvolvimento – DataViva”. O endereço oficial do Redis é:

* http://redis.io

O Redis foi criado mais para rodar em servidor Linux, tanto que no site oficial existe as distribuições já compiladas e prontas para download apenas Linux, seja 32 ou 64 Bits. No caso do Windows eles têm um repositório não oficial, onde existe a compilação para Windows 32 e 64 bits

* <https://github.com/MSOpenTech/redis>
* https://github.com/rgl/redis/downloads

O arquivo de instalação do Redis que utilizamos neste ambiente foi o:

* redis-2.4.6-setup-64-bit.exe

Não houve qualquer alteração no padrão de instalação ou nas configurações do Redis, portanto basta instalar e então ele será iniciado como serviço. Para acessar este serviço o servidor de Aplicação deverá então conectar neste servidor através do IP e utilizando a porta padrão do Redis, no caso o 6379.

Um ponto interessante nesta instalação é que até o momento a aplicação ficava junto com o servidor do Redis. No nosso caso houve uma separação, em um servidor está o banco de dados e o Redis, e em outro servidor está a aplicação em Python. Sendo assim foi necessário realizar uma alteração no código para que ele utilize parâmetros diferentes do default para realizar a conexão com o Redis. Isto que dizer que, no código é necessário informar o IP e a porta do servidor do Redis para o Python se Conectar.

Outro ponto que poderia ser de alerta é relacionado à liberação desta porta para conexão entre os servidores, mas como estão sob o mesmo contrato e mesma rede, as portas de conexão entre eles já estão liberadas. Mas em casos onde não for este tipo de infraestrutura, a liberação da porta 6379 deverá ser levada em consideração.

Se houver qualquer biblioteca necessária durante a instalação do Redis, é possível utilizar um programa que pega todas as dependências de um programa:

* http://www.dependencywalker.com/

APLICAÇÃO

A aplicação foi construída utilizando em grande parte a linguagem de programação *Python* e o framework *Flask*. Existe mais detalhes sobre a linguagem e o framework no documento “*Guia de Desenvolvimento – DataViva*”, portanto nosso foco aqui será apenas na instalação.

Como tivemos problemas de limitação na biblioteca que realiza a exportação dos gráficos, no servidor Windows, estamos utilizando outro servidor Linux para hospedar a camada de aplicação.

No caso do nosso servidor temos o Sistema Operacional Red Hat, ele utiliza por padrão a versão do Python 2.6. Esta é a versão homologada para que todos os feitos em Python e que é utilizado por este Sistema Operacional. Por isso trocar a versão do Python pode ser perigosa neste caso, como alternativa podemos instalar o Python em outra pasta e utilizar o nome diferente para não comprometer o SO.

Como primeiro passo vamos instalar bibliotecas disponíveis no Yum do Red Hat e que serão úteis no futuro no sistema, para isso execute:

* yum install python-devel zlib-devel MySQL-python

Em casos de servidores com distribuição baseadas em Debian, possível realizar a instalação da seguinte forma:

* apt-get install python-mysqldb

Devemos então baixar os fontes do Python na página do fabricante, para então compilarmos e por instalar no servidor, para baixar acesse:

* <http://www.python.org/getit/>

Baixe o arquivo de fontes, que vem compactado, faça a extração dos arquivos em uma pasta local e entre nesta pasta:

* wget http://www.python.org/ftp/python/2.7.3/Python-2.7.3.tgz # Download
* tar xvfz Python-2.7.3.tgz # unzip
* cd Python-2.7.3 # go into directory

Execute o seguinte comando para fazer a sua configuração, modo de prepara-lo antes de Compilar:

* ./configure --enable-shared

No final do processo, se não ocorrer erros, execute os dois comandos abaixo, para criar realizar a compilação e por último instalar de forma alternativa, ou seja, sem prejudicar as configurações da instalação padrão do Python2.6:

* make
* make altinstall

Temos agora o Python 2.7 que pode ser acessado através do comando python2.7, e o mesmo acontece com o pip, que pode ser acessado através do pip2.7. Os links anteriores do Python 2.6 continuam funcionando, no caso através do comando python e também python2.6.

Para ver aonde fica a nova instalação do Python 2.7, acesse a pasta:

* /usr/lib/Python-2.7.3
* Caso queira conferir a versão, faça:
  + /usr/lib/Python-2.7.3/python –version

Existem algumas variáveis de configuração do PS que devem ser inicializadas no sistema operacional. No caso do Linux o comando para realizar tal operação chama export, e deve ser executado com todas as variáveis necessários, como no exemplo abaixo:

* export DATAMINAS\_DB\_NAME=ps
* export DATAMINAS\_DB\_PW=123456
* export DATAMINAS\_DB\_USER=root
* export DATAMINAS\_SECRET\_KEY=a
* GOOGLE\_OAUTH\_ID
* GOOGLE\_OAUTH\_SECRET

Utilizar o Guia de Desenvolvimento – DataViva para realizar a instalação de outros componentes do Flask e do Python, além de criação de variáveis de ambiente necessários para finalizar a instalação.

* 1. Servidor Web

A equipe da *GV* está utilizando o *NginX* para hospedar arquivos estáticos e o *Apache* para criar os arquivos dinâmicos. Isto quer dizer que imagens e outros arquivos que não mudam, quem fornece ao usuário é o NginX, mas no caso tabela e gráficos que são criados de forma dinâmica, de acordo com filtros e banco de dados, é fornecido pelo usuário através do Apache.

Aqui está a configuração do Apache da forma como a equipe da GV usa:

NameVirtualHost 127.0.0.1:8058

<VirtualHost 127.0.0.1:8058>

#Basic setup

ServerAdmin alex@growth-ventures.com

ServerName dataminas.growth-ventures.com

WSGIDaemonProcess dataminas user=hermes group=hermes threads=5

WSGIScriptAlias / /web/dataminas.growth-ventures.com/dataminas.wsgi

<Directory /web/dataminas.growth-ventures.com>

WSGIProcessGroup dataminas

WSGIApplicationGroup %{GLOBAL}

Order deny,allow

Allow from all

</Directory>

</VirtualHost>

Fizemos alguns testes e pesquisas para determinar se há realmente a necessidade de utilizar o NginX e o Apache desta forma. Vimos que pode não existir ganho e, caso exista, será muito pequeno, o que não justifica instalar dois sistemas e aumentar a dependência. Sendo assim optamos por realizar apenas o uso do Apache, tanto para acesso de arquivos estátivcos como dinâmicos. Para instalar o apache no sistema utilize o seguinte comando:

* yum install apache2

Após a instalação do Apache2 é possível encontrar o arquivo de configurações na pasta:

* /etc/httpd/conf/httpd.conf

Para garantir que a biblioteca certa do Python está disponível para uso pelo Apache, e principalmente pelo módulo WSGI do Apache, verifique se existe o arquivo da biblioteca no */usr/lib64/libpython2.7.so.1.0*, caso não exista crie um link da seguinte forma:

* *ln -s /usr/local/lib/libpython2.7.so.1.0 /usr/lib64/libpython2.7.so.1.0*

Abra o arquivo de configuração deste projeto no Apache deverá ficar como abaixo, considerando que para funcionar por completo é necessário terminar o procedimento descrito neste capítulo.

Listen 8080

<VirtualHost 200.198.9.180:8080>

DocumentRoot /web/productspace/dataminas

<Directory "/web/productspace/dataminas/static">

Options Indexes FollowSymLinks MultiViews ExecCGI

AllowOverride All

Order allow,deny

Allow from all

</Directory>

#MOD WSGI

SetEnv DATAMINAS\_DB\_NAME ps

SetEnv DATAMINAS\_DB\_PW 123456

SetEnv DATAMINAS\_DB\_USER root

SetEnv DATAMINAS\_SECRET\_KEY a

WSGIDaemonProcess dataminas user=apache group=apache threads=5 python-path=/usr/local/lib/python2.7/site-packages

WSGIProcessGroup dataminas

WSGIScriptAlias / /web/productspace/dataminas.wsgi

Alias /static /web/productspace/dataminas/static

<Directory /web/productspace>

WSGIProcessGroup dataminas

WSGIApplicationGroup %{GLOBAL}

Order deny,allow

Allow from all

</Directory>

</VirtualHost>

WSGISocketPrefix /web/productspace

WSGIPythonPath /usr/local/bin/python2.7

#WSGIPythonExecutable /usr/local/bin/python2.7

#WSGIPythonHome /usr/local/lib/python2.7/site-packages

* 1. Integração Servidor Web com Python

Temos então a aplicação em Python e Flask instalada, com conectividade ao banco de dado MySQL e o sistema de cache Redis. Agora precisamos fazer o integração entre esta aplicação em Flask com o sistema de servidor Web, através desta integração que será possível que usuários através da internet acessarão a aplicação.

Existem muitas formas de realizar esta integração, utilizando diferentes módulos do Apache ou até mesmo via um Proxy, que nada mais é que um redirecionamento do Apache para um servidor local em Python.

Nesta pesquisa encontramos muitas soluções e chegamos a implementar algumas, mas por questões de performance tivemos que utilizar a integração que podemos considerar a mais eficiente. A integração utilizando o módulo mod\_wsgi compatível com o Apache e faz uma conexão ao *Python* por meio do *WSGI*. Como o *Flask* é executado através da tecnologia *WSGI*, esta é a forma mais simples e crua para realizar a integração, por consequência a que apresenta a melhor performance.

O grande problema que tivemos neste caso foi o fato do nosso servidor ser Red Hat, e o padrão deste sistema operacional é utilizar Python 2.6. A aplicação foi construída em Python 2.7, sendo assim foi necessário encontrar a compilação desse Módulo WSGI para Python 2.7, compatível com o Red Hat 64 Bits. A página oficial do módulo, inclusive explicando como realizar a compilação com este SO:

* http://code.google.com/p/modwsgi

Existe uma documentação do Framework Flask que descreve como deve ser realizado a integração com o Apache através do MOD\_WSGI

* http://flask.pocoo.org/docs/deploying/mod\_wsgi/

Realizamos testes deste módulo WSGI no Windows 64 Bits e tivemos problemas para encontrar uma versão compatível para o Apache. No caso do IIS existe um módulo para WSGI, mas não funciona por completo e tem limitações com a maioria dos frameworks.

No caso do nosso SO, RedHat 64 Bits com Apache2, tivemos problema para ter uma versão compilada para o Python 2.7. A solução que encontramos foi pegar a versão compilada deste módulo da equipe GV, que por sorte foi compatível com nosso sistema operacional. No caso deles é um servido com SO Ubuntu Server 64 Bits, Apache2 e o Python 2.7 instalado como padrão.

Para habilitar este módulo adicione ou tire o comentário desta linha, no arquivo de configuração do Apache2:

LoadModule wsgi\_module modules/mod\_wsgi.so

* http://code.google.com/p/modwsgi/wiki/InstallationOnWindows
* 64 - http://www.lfd.uci.edu/~gohlke/pythonlibs/#mod\_wsgi

Normalmente estas bibliotecas do mod\_wsgi ficam nos seguintes diretórios:

* /usr/lib64/httpd/modules/mod\_wsgi.so (geralmente a versão que vem no servidor
* /usr/lib64/httpd/modules/mod\_wsgi.so-2.7

Para verificar se o modulo foi iniciado corretamente no apache e utilizando a versão do Python desejado, reinicie o apache e veja no arquivo de log do apache:

[Fri May 16 10:34:15 2014] [warn] mod\_wsgi: Compiled for Python/2.7.2+.

[Fri May 16 10:34:15 2014] [warn] mod\_wsgi: Runtime using Python/2.7.3.

[Fri May 16 10:34:15 2014] [notice] Apache/2.2.15 (Unix) DAV/2 mod\_wsgi/3.3 Python/2.7.3 configured -- resuming normal operations

Neste exemplo mostra exatamente q versão que o módulo foi compilado e em qual ele irá rodar, confirmando o uso da versão do Python 2.7.

* + 1. Outros Métodos

Não estamos utilizando qualquer framework para realizar a integração entre servidor Web e a aplicação, mas a título de documentação, vamos explicar aqui como é feita a integração por meio de um framework WSGI. Para isso é necessário instalar o CherryPy, que executa o projeto Flask em uma porta local do servidor e então faz-se um proxy entre o Apache o CherryPy. Neste caso utilizamos o mod\_proxy para realizar a integração com o servidor WSGI, mais informações sobre este módulo pode ser encontrado em:

* http://httpd.apache.org/docs/current/mod/mod\_alias.html

O arquivo de configuração do Apache, no local das configurações do Mod\_WSGI, é necessário colocar da seguinte forma:

Alias /static /web/productspace/dataminas/static

ProxyRequests off

ProxyPass /static/ !

ProxyPass / http://200.198.9.180:8080/ min=1 ttl=20 acquire=10000 timeout=60

ProxyPassReverse http://200.198.9.180:8080/ /

Abaixo temos as principais formas de integração e alguns comentários sobre pontos positivos e negativos de cada uma:

* FastCGI:
  + Existe uma integração nativa do IIS e módulos para o Apache, como o mod\_proxy\_fcgi. Foram realizados diversos testes, inclusive com a equipe do Flask e do Aphace, mas não foi possível realizar a integração com este framework.
    - <http://wiki.nginx.org/Configuration#Python_via_FastCGI>
  + O principal problema aqui foi conseguir compilar uma versão do Apache com este módulo para FCGI em Windows 7 64 Bits.
  + O problema relacionado ao Flask foi durante o uso do módulo Flup, utilizado para fazer a conexão com FCGI para o Python. Tentamos diversas configurações mas em todos os casos
    - http://wiki.nginx.org/PythonFlup
* Passenger
  + http://kbeezie.com/using-python-nginx-passenger/
* WSGI
  + Esta parece ser a solução mais recomendada para nosso caso, pois o Flask é um framework que utiliza a tecnologia WSGI, portanto seria uma ligação direta entre o servidor HTTP e a aplicação do Python.
  + Existem diversos frameworks que utilizam o WSGI para realizar a integração, entre eles: Mod\_WSGI, CheeryPy, uWSGI, gUnicorn, Waitress, Tornado entre outros
  + Na própria
    - <http://flask.pocoo.org/docs/deploying/mod_wsgi/>
    - http://flask.pocoo.org/docs/deploying/wsgi-standalone/

A equipe da GV está utilizando em ambiente Linux com Apache e o Mod\_WSGI, no nosso caso estamos utilizando Windows com a mesma tecnologia WSGI. Utilizando um Benchmark realizado na internet vimos que em termos de performance o Mod\_WSGI do Apache é similar ao CherryPy:

* + http://nichol.as/benchmark-of-python-web-servers
    1. NGINX

Existe a possibilidade de utilizar o *Nginx* *1.2.7* para realizar a requisição com o navegador, fornecendo apenas conteúdo estático e realizando um Proxy para arquivos dinâmicos para o *Apache* resolver. Para servidor de aplicação será utilizado o *NginX*:

* <http://nginx.org/en/download.html>

Baixe a versão para Windows estável mais atualizada, que na verdade é um zip com o *Nginx* pronto para execução. Faça então a extração dos dados para a seguinte pasta:

* *C:\Program Files\nginx-1.2.7\*

Faça o seguinte procedimento para iniciar:

* Entre na pasta do *Nginx* descompactado (a mesma citada acima)
* Dê permissão total ao seu usuário nesta pasta
* Execute o Comando: *start nginx*
* Confira que o processo está rodando: *tasklist /fi "imagename eq nginx.exe"*
* Veja dentro da pasta *nginx\logs\* se existe algum problema ao iniciar, principalmente no error.log

Alguns motivos que podem fazer não funcionar o *Nginx*:

* Verifique se não tem um serviço utilizando a porta 80 de sua máquina, se tiver o *IIS* instalado no *Windows* ele pode estar usando esta porta. Se desejar trocar a porta do *Nginx*, altere o arquivo conf\nginx.conf dentro da pasta onde está o Nginx.
  + server {
  + listen 8081;

Caso tenha iniciado sem mensagens de erro você poderá acessar a seguinte url:

* http://127.0.0.1/index.html

Veja que este arquivo *HTML* é o que fica na pasta nginx\html\index.html.

Com a instalação do Nginx funcionando vamos começar as configurações específicas do projeto. Para configurar o Nginx abra o arquivo nginx\conf\nginx.conf com um editor de texto.

Você encontrará uma configuração do servidor, que fica da seguinte forma

http {

server{

...

}

}

Adicione uma localização para fazer a chamada das página dinâmicas:

location / {

proxy\_pass http://localhost:8000;

proxy\_set\_header X-Real-IP $remote\_addr;

}

Para que os arquivos estáticos não sejam tratados no *sistema*, e assim ganhar performance, vamos configurar para que o Nginx faça a leitura destes arquivos diretamente do disco rígido:

location /media/ {

root html;

}

Lembrando que os arquivos estáticos estão na pasta WWW\HTML\, que fica dentro do repositório com o código fonte do sistema, deve estar na pasta do HTML do NGINX. Utilizando nosso padrão de instalação a pasta com os códigos fontes devem ficar em:

* C:\Program Files\nginx-1.2.7\html

Caso precise parar o Nginx para iniciar novamente, aplicando as alterações realizadas no nginx.conf, execute o comando abaixo dentro da pasta do Nginx:

* nginx –s stop
  1. Configurações de Ambiente

Existem algumas outras configurações para funcionar este aplicativo no sevirdor.

O fonte do sistema deve ter permissão 775 com dono adm:apache, caso não seja respweitado esta permissão muito provavelmente terá um erro na pasta search.db como no exemplo abaixo:

* OSError: [Errno 13] Permission denied: '/web/productspaceNew/search.db/Question'

PERFORMANCE

Com o intuito de garantir que este sistema seja disponibilizado ao seu público de forma a conseguir a atender as suas necessidades, é necessário fazer um trabalho para garantir a sua performance. Este trabalho abrande diversas áreas de estudo, desde a análise de comportamento dos usuários até testes de estresse no site para determinar sua capacidade.

Neste caso tratamos de um novo site a ser disponibilizado, por isso não temos um histórico de acessos nem algo que possamos trabalhar para tentar determinar quantos acessos teremos. O mais próximo que conseguimos chegar de uma métrica é considerando como ponto de referência o Atlas, <http://atlas.media.mit.edu/>, site disponibilizado pelo Media Labs do MIT. Esta é uma ferramenta foi utilizada como base para o sistema, inclusive pelos mesmo desenvolvedores.

Pegamos os três picos de acesso no histórico do Atlas, desde a sua criação até o ano de 2013, eles são:

* Dia 25 de Fevereiro de 2013, de 16:00 às 17:00:
  + 427 Visitantes Únicos
  + 1,304 Visualização de Páginas
  + 1,037 Páginas Únicas Visualizadas
  + 1:14 Tempo médio no site
  + 42.15% Taxa de desistência
* Dia 8 de Janeiro de 2013, de 17:00 às 16:00:
  + 266 Visitantes Únicos
  + 848 Visualização de Páginas
  + 715 Páginas Únicas Visualizadas
  + 2:46 Tempo médio no site
  + 52.26% Taxa de desistência
* Dia 5 de Novembro de 2013, de 10:00 às 11:00:
  + 172 unique visitors
  + 1,054 Visualização de Páginas
  + 646 Páginas Únicas Visualizadas
  + 1:38 Tempo médio no site
  + 55.41% Taxa de desistência

Considerando estas informações podemos agora montar um cenário de testes para garantir que o DataViva instalado em nossa infra irá suportar o mesmo volume de acessos. Vamos então pegar o volume de acessos do dia 25 de Fevereiro de 2013 e, se pegarmos o pico de Visitantes Únicos em um intervalo de 10 minutos, temos 427/6 = 71 Usuários simultâneos. Se pegarmos todos as visualizações de páginas e dividirmos pelo número de visitantes teremos quantas página em média eles acessam, no caso 1304/427 = 3 páginas por visitante.

Nosso teste deverá então simular pelo menos 70 usuários acessando 3 páginas, em um intervalo de 10 minutos. Durante esta simulação devemos entrar no sistema e realizar uma visitação exploratória, para garantir que ele continua acessível ou se devido ao número de acessos a página pare de responder.

Vamos testar:

* Visualização de Páginas por usuário: 3 ( 1304 Visualizações / 427 Visitantes )
* Visitantes: 71 Usuários únicos 71 ( 427 / 60 minutos por hora \* 10 Minutos)
* Tempo Total: 10 minutos

Temos ainda que considerar que a taxa de rejeição é de 50%, ou seja, somente metade destes usuários continua acessando o sistema fora a página inicial e por isso o cenário de testes acima irá exigir mais do que o real.

A ferramenta que utilizamos para estes é a *jMeter* e no documento “Ambiente de Desenvolvimento – DataViva” ensina como instalar, configurar e executar os testes. O script de teste foi salvo junto com o código fonte e neste documento iremos demonstrar apenas os resultados obtidos com os testes.

A ferramenta que iremos utilizar nestes testes é a *jMeter.* Baixe os fontes e realize a descompactação dos arquivos em uma pasta local, para inicar o jMeter basta executar o arquivo:

* C:\Users\m1165564\Documents\Projetos\apache-jmeter-2.9\bin\jmeter.bat

Descompacte o jmeter junto com os arquivos fontes de teste, configure o proxy ( no caso da Cidade Administrativa - <http://proxy.prodemge.gov.br/proxy.pac> ) e execute o comando:

* jmeter.bat -H proxy.prodemge.gov.br -P 8080

1. Teste Externo
   1. 17:27 a 17:35 = 8 min
   2. Media Inicial: 563
   3. Media Final 4000
2. Teste Local
   1. 37 – 54 = 17 min
   2. Media Inicial: 3525

* 1. Load Balance

Para termos então uma boa performance, é necessário criar um pool de servidores para responder. O melhor dos casos é ter um CDN, como se fosse um servidor em cada continente, assim não é preciso que um usuário dos EUA espere uma resposta de um servidor da Ásia, que demora bem mais do que uma resposta de um servidor dos EUA.

A realização deste balanceamento pode ser realizado através do apache, seguindo as seguintes configurações:

ANEXO 1 – Scripts Apache

Aqui temos os scripts mais utilizados pelo Apache para configuração do dataviva em diferentes modos.

Primeiro temos a configuração que apenas redireciona para o servidor dataviva.info, externo ao servidor do Escritório;

## PRODUCT SPACE TEMP INIT

<VirtualHost 200.198.9.180:80>

DocumentRoot /web/productspace/temp

ServerName dataviva.mg.gov.br

DirectoryIndex index.htm

RewriteEngine on

RewriteRule ^/(.\*) http://www.dataviva.info/$1 [R]

</VirtualHost>

<VirtualHost 200.198.9.180:80>

DocumentRoot /web/productspace/temp

ServerName www.dataviva.mg.gov.br

DirectoryIndex index.htm

RewriteEngine on

RewriteRule ^/(.\*) http://www.dataviva.info/$1 [R]

</VirtualHost>

###################

#TST

################

<VirtualHost 200.198.9.180:8081>

DocumentRoot /web/productspace/temp

ServerName www.escritorio.mg.gov.br

DirectoryIndex index.htm

#MOD WSGI

SetEnv DATAVIVA\_DB\_HOST 200.198.9.184

SetEnv DATAVIVA\_DB\_NAME ps

SetEnv DATAVIVA\_DB\_PW mysql

SetEnv DATAVIVA\_DB\_USER root

SetEnv DATAVIVA\_SECRET\_KEY a

SetEnv DATAVIVA\_PRODUCTION True

SetEnv DATAVIVA\_OAUTH\_TWITTER\_ID skzy1ia2XyLfwAmjAtrREg

SetEnv DATAVIVA\_OAUTH\_TWITTER\_SECRET GRsFBkYsU3JfxAoCTcmxDnrNsHF1WJSYrGzudYTOvM

SetEnv DATAVIVA\_OAUTH\_FACEBOOK\_ID 397693097001481

SetEnv DATAVIVA\_OAUTH\_FACEBOOK\_SECRET b03aece555743071070d69b890d4195e

SetEnv DATAVIVA\_MIRROR http://www.dataviva.info/

<Directory "/web/productspaceTst/dataviva/static">

Options Indexes FollowSymLinks MultiViews ExecCGI

AllowOverride All

Order allow,deny

Allow from all

AuthType Basic

AuthName "Restricted Files"

AuthBasicProvider file

AuthUserFile /etc/httpd/passwords

Require user dataviva

</Directory>

#dataviva.mg.gov.br:8080

SetEnv DATAVIVA\_OAUTH\_GOOGLE\_ID 521034971212-be0uppcqb8appn7f0pmk9555nd87plbu.apps.googleusercontent.com

SetEnv DATAVIVA\_OAUTH\_GOOGLE\_SECRET pRN-FJTP0H\_IsY2kUgjIAtFI

WSGIDaemonProcess dataminastst user=apache group=apache threads=5 python-path=/usr/local/lib/python2.7/site-packages

WSGIProcessGroup dataminastst

WSGIScriptAlias / /web/productspaceTst/visual.wsgi

ErrorLog /var/log/web/datavivaTst.log

</VirtualHost>