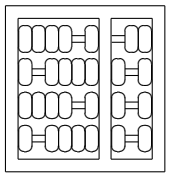

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Manual - Grupo 5
Projeto de Sistemas de Informação

Luiz Rodolfo Felet Sekijima *Klaus Rollmann*
Renan Camargo de Castro
Wendrey Lustosa Cardoso
Luis Felipe Hamada Serrano

Technical Report - IC-16-1 - Relatório Técnico

March - 2016 - Março

Manual para projeto de MC437
Projeto de Sistemas de Informação

Luiz Rodolfo Felet Sekijima Klaus Rollmann Renan Camargo de Castro
Wendrey Lustosa Cardoso Luis Felipe Hamada Serrano

Resumo

Este é um manual de como utilizar o projeto de sistema altamente disponível, da matéria MC437 (Projeto de Sistemas de Informação). Estão descritos os procedimentos para implementar as ferramentas de banco de dados (PostgreSQL), o servidor web (Tomcat), o aplicativo (TPC-W) e emulador de navegadores (RBE). Também estão incluídos os passos para execução dos testes e obtenção dos resultados.

1 Execução do Experimento

Nesta seção, será detalhado como executar o experimento que testa o funcionamento do sistema altamente disponível, com a injeção de duas falhas em momentos distintos.

1.1 Acesso ao Usuário e Máquina

A execução dos experimentos necessita do uso de duas máquinas da cbn5 e cbn2 e os servidores dbmaster2 e dbslave2. A primeira é a máquina usada pelo grupo, que irá conter todos os scripts e configurações. A segunda irá executar os RBEs e mostrar os resultados, e as outras são contém os bancos de dados.

O experimento necessita de um usuário com permissão sudo.

1.2 Configurações Banco, Tomcat, HAProxy

O Tomcat e HAProxy já estão configurados corretamente na cbn5. Os bancos Master e Slave também estão configurados corretamente nas máquinas dbmaster2 e dbslave2. Todas as configurações utilizadas estão em suas respectivas pastas dentro no Git.

1.3 Diretórios

Inicialmente, recomenda-se a criação de uma nova pasta para colocar os arquivos e scripts para execução do experimento. A pasta deve ser criada na máquina cbn5. Iremos supor que o nome da pasta seja *cbn5_exp*, criada da seguinte forma:

```
$ cd ~
$ mkdir cbn5_exp
$ cd cbn5_exp
```

Além disso, uma pasta deve ser criada na cbn2, para executar os rbes

```
$ ssh cbn2
$ cd ~
$ mkdir cbn2_exp
$ cd cbn2_exp
```

O restante do manual irá supor que todo o experimento será executado dentro dessas pastas.

1.4 Inicialização do Banco

Para executar o banco em uma cópia nova do banco, deve-se utilizar o script *reset_master_wrapper.sh*, que restaura o banco dbmaster2 com o dump fornecido e inicia dbslave2 como backup. As instruções são mostradas abaixo:

```
$ cp ~/grupo05/programas/bash/reset_master_wrapper.sh .
$ cp ~/grupo05/programas/haproxy/haproxy_master_slave.cfg .
$ sudo ./reset_master_wrapper.sh
```

1.5 Scripts e Arquivos Utilizados

Os scripts bash e python utilizados encontram-se na pasta */programas/bash* e */programas/python*, respectivamente, dentro do git. Os arquivos de configuração do haproxy estão em */programas/haproxy*. Os arquivos necessários para o experimento 3 são mostrados abaixo.

- fail-detector.py
- promote_master.sh
- promote_slave.sh
- reup_master.sh
- reup_slave.sh

- kill_master.sh
- kill_slave.sh
- exp3_kill_timer.sh
- reload_cfg_haproxy.sh
- haproxy_slave_master.cfg
- haproxy_master_slave.cfg

Esses scripts devem ser colocados na pasta criada na cbn5.
Para copiar rapidamente, pode-se fazer simplesmente

```
$ cp ~/grupo05/programas/bash/*.sh
$ cp ~/grupo05/programas/python/*.py .
$ cp ~/grupo05/programas/haproxy/*.cfg .
```

Já para a execução dos RBEs, é necessário acessar a cbn2 e copiar os seguintes arquivos, para uma pasta, aqui chamada cbn2_exp:

- analyze2.py
- exp3.sh

Para copiar, primeiro é necessário criar a pasta

```
$ ssh cbn2
$ mkdir cbn2_exp
```

Em seguida usar o comando *scp*, na cbn5, para copiar para cbn2.

```
$ scp ~/grupo05/programas/bash/exp3.sh cbn2:~/cbn2_exp/
$ scp ~/grupo05/programas/python/analyze2.py cbn2:~/cbn2_exp/
```

1.6 Execução do Experimento

Inicialmente, são necessários três terminais, um para executar os RBEs, outro para o fail-detector e o último para programar o timer que injeta as falhas.

O único cuidado que deve-se tomar é adicionar a flag-Y ao executar o ssh para cbn2 (e para cbn5), para que os gráficos sejam mostrados via ssh.

```
Terminal 1:
$ cd ~/cbn5_exp

Terminal 2:
$ cd ~/cbn5_exp
```

```
Terminal 3:
$ ssh -Y cbn2
$ cd ~/cbn2_exp
```

No terminal 1, será executado o script de detecção de falha *fail-detector*, que ira promover e restaurar os bancos no caso de falha. Para isso basta executar:

```
Terminal 1:
$ sudo ./fail-detector.py
```

No terminal 2, será executado o timer com injeção de falhas nos instantes 20s e 60s após a execução do script. O tempo das falhas pode ser alterado no script *exp3_kill_timer*. O comando só deve ser executado após rodar os RBEs no terminal 3.

No terminal 3, na cbn2, são executados os RBEs. O número de clientes, modo de operação e think time são parâmetros que podem ser modificados, basta alterar o cabeçalho do arquivo *exp3.sh*.

Após escolher os parâmetros, deve-se preparar os terminais 2 e 3, da seguinte forma.

```
Terminal 3 (cbn2):
$ ./exp3 <número do experimento> <mensagem de readme>

Terminal 2 (cbn5):
$ sudo ./exp3_kill_timer.sh
```

Os dois scripts devem ser executados logo em seguida, um em cada terminal.

1.7 Acompanhamento e Resultados

Podemos acompanhar a situação dos bancos master e slave pelo haproxy na página *http://cbn5.lab.ic.unicamp.br/admin?stats*

Ao final do experimento, um gráfico irá mostrar os resultados. A média de WIPS e WIRT estará no arquivo *results*, dentro da pasta nomeada pela data e número do experimento passado como parâmetro. A mensagem passada como parâmetro estará no arquivo *readme*.

2 Acesso à maquina

Primeiramente é necessário acessar a máquina do cluster utilizada pelo grupo 5. Para acessar, é preciso usar o comando ssh usando o usuario e senha. O ip da maquina em questão é cbn5.lab.ic.unicamp.br (IP público: 143.106.73.139). O comando utilizado, a partir de um dos computadores do IC, é

```
ssh [usuario]@cbn5.lab.ic.unicamp.br
```

3 Instalação PostgreSQL

Após acessar a máquina do cluster, é necessário instalar o banco de dados PostgreSQL na versão 9.5.1.

Essa versão não está disponível na versão do Ubuntu instalada, portanto é preciso adicionar um repositório extra.

3.1 Adicionando Repositório

Para adicionar o repositório basta adicionar o repositório do postgres nos *sources.list.d*. Para adicionar o repositório é preciso criar um novo arquivo, como mostrado no comando a seguir.

```
vi /etc/apt/sources.list.d/postgresql.list
```

Em seguida, a seguinte linha deve ser adicionada ao arquivo.

```
deb http://apt.postgresql.org/pub/repos/apt/ trusty-pgdg main
```

3.2 Instalação

Para instalar é necessário primeiramente dar update nos repositórios com o comando

```
sudo apt-get update
```

E em seguida pode-se instalar o postgresql versão 9.5.1 com o *apt-get*.

```
sudo apt-get install postgresql-9.5
```

3.3 Restauração do Dump

Para restaurar o banco, é preciso criar a role tpcw com senha 'tpcw_user' no postgres, isso pode ser feito usando o comando sql:

```
CREATE ROLE tpcw_user WITH PASSWORD 'tpcw' LOGIN;
```

Esse comando deve ser executado logando-se com o user do postgres com *su - postgres* e executando o utilitário psql. Após isso, basta restaurar o dump do banco com os comandos a seguir:

```
cp /nfs/postgres/tpcw-database-dump.tar .
pg_restore -d tpcw tpcw-database-dump.tar
```

Esse comando restaura o banco *tpcw-database-dump.tar* em um *db* chamado *tpcw*.

4 Configurando Servidor Web

O servidor web usado foi o Apache Tomcat na versão 7.x.

4.1 Instalação Tomcat 7

A instalação é feita usando o comando:

A instalação é feita no diretório salvo na variável *$CATALINA_BASE*, que representa o diretório */var/lib/tomcat7*.

4.2 Integração com o PostgreSQL

Para integrar com o PostgreSQL é necessário baixar o driver JDBC no link abaixo.

`https://jdbc.postgresql.org/download/postgresql-9.4.1208.jre7.jar`

O arquivo *jar* baixado deve ser colocado no diretório *$CATALINA_BASE/lib* Em seguida o *ResourceLink* abaixo precisa ser adicionado ao campo *Context* do arquivo *$CATALINA_BASE/conf/context.xml*

```
<ResourceLink type="javax.sql.DataSource"
              name="jdbc/TPCW"
              global="jdbc/TPCWPool"/>
```

Além disso, o *Resource* abaixo tem que ser adicionado ao *GlobalNamingResources* do *server.xml* encontrado em *$CATALINA_BASE/conf/server.xml*.

```
<Resource type="javax.sql.DataSource"
          name="jdbc/TPCWPool"
          factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
          driverClassName="org.postgresql.Driver"
          url="jdbc:postgresql://localhost:5432/tpcw"
          username="tpcw"
          password="tpcw_user"
          initialSize="10"
          maxActive="200"
          maxIdle="100"
          minIdle="10" />
```

5 Configurando Webapp - TPCW

Para configurar o webapp(TPCW), é preciso adicionar o usuário no grupo tomcat7, para poder fazer as alterações na pasta do tomcat7 como usuário comum. Em seguida, deve-se criar a pasta tpcw em *$CATALINA_BASE/webapps/*, onde *$CATALINA_BASE* é */var/lib/tomcat7/*. Depois, é necessário extrair o arquivo Images.tar.gz na pasta, com o comando

```
cp /nfs/servlet/Images.tar.gz .
tar xfvz Images.tar.gz
```

O arquivo tpcw-servlets.tar.gz deve ser copiado para uma pasta, extraído e executado o comando make para compilar os sources do tpcw. Então, deve-se copiar a pasta classes para */var/lib/tomcat7/webapps/tpcw/WEB-INF*, assim como o arquivo *web.xml*. Para finalizar, a pasta deve ter as permissões corretas e o servidor deve ser reinicializado.

```
chgrp tomcat7 . -R
chmod o-rwx . -R
invoke-rc.d tomcat7 restart
```

6 RBE - Cliente

Para realizar os testes no banco de dados, é usado o *Remote Browser Emulator*. A ferramenta gera um conjunto de clientes que acessam o servidor do TPC-W. Para rodar a ferramenta, deve-se extrair o programa RBE do repositório para uma pasta do cluster e o executar com os seguintes comandos:

```
java -cp /nfs/rbe/rbe.jar rbe.RBE -EB rbe.EBTPCW2Factory 1 -OUT
output1.m -CUST 14400 -ITEM 10000 -WWW http://cbn2/tpcw/
-RU 1 -MI 30 -RD 1 -DEBUG 10

java -cp /nfs/rbe/rbe.jar rbe.RBE -EB rbe.EBTPCW3Factory 30 -OUT
output2.m -CUST 14400 -ITEM 10000 -WWW http://cbn2/tpcw/
-RU 5 -MI 60 -RD 5  -TT 1.0
```

Esse comandos testam o funcionamento do RBE para alguns parametros genéricos. O comando para executar o RBE é da seguinte forma: Comando para executar o RBE:

```
java $CP ${ET[$J]} ${NB[$I]} $OUTPUT $DB $URL $TIME $TT
```

Em que:

```
CP (caminho da classe): -cp [caminho_classe]
ET (tipo de clientes): rbe.RBE -EB <TIPO>
<TIPO> pode ser um dos seguintes:
    rbe.EBTPCW1Factory - Clientes que procuram itens na loja (leitura)
    rbe.EBTPCW2Factory - Clientes que adicionam itens no carrinho (leitura e escrita)
    rbe.EBTPCW3Factory - Clientes que concluem a compra (escrita)
NB (número de clientes): [numero_inteiro]
OUTPUT (arquivo de saída): -OUT [arquivo_saida]
DB (informações do banco de dados): -CUST 14400 -ITEM 10000
URL (site do banco de dados): -WWW http://cbn5.lab.ic.unicamp.br:8080/tpcw/
TIME (tempo de inicio, execucao e finalizacao): -RU 5 -MI 180 -RD 5
TT (tempo de reflexao do cliente): -TT 1.0
```

Uma vez funcionando o RBE, é possível executar os scripts dos experimentos.

7 Experimentos

Os experimentos são divididos em experimento 1 e experimento 2, em que o experimento 1 varia o número de RBEs dos três tipos, para um TT de 1.0 e o segundo experimento varia tanto o número de RBEs quanto o TT de 0.1 até 0.9.

7.1 Dados experimentos

Para obter os dados do experimento 1 e 2, basta executar os scritps bash que se encontram na pasta */programas/bash* chamados *data_exp1.sh* e *data_exp2.sh*. Esses scripts executam os RBEs e obtém as informações para o experimento 1 e experimento 2 respectivamente. Esse script deve ser executado na mesma pasta dos RBEs, chamada (tpcw_rbe).

Esses comandos podem ser realizados usando:

```
./data_exp1.sh
./data_exp2.sh
```

Esses scripts geram os dados nas pastas *data_exp1* e *data_exp2* respectivamente.

7.2 Gerando Gráficos

Para gerar os gráficos, são feitas duas etapas. O primeiro script processa os dados e gera um arquivo com os pontos para se plotar no gráfico. O segundo script é para o gnuplot plotar os gráficos e gerar a imagem.

Lembrando que para executar esse comando é necessário possuir o gnuplot instalado. Caso não esteja instalado, basta realizar um

```
sudo apt-get install gnuplot
```

Na primeira etapa do experimento 1, basta mover os arquivos *exp1.py* e *exp1_runner.sh* que se encontram na pasta *programas/bash* para a pasta com os dados do experimento. Para isso deve-se executar os comandos dentro da pasta dos experimentos

```
cp ~/grupo5/programas/bash/exp1_runner.sh .
cp ~/grupo5/programas/python/exp1.py .
```

O primeiro script executa o segundo para todos os arquivos, sendo assim, só é preciso rodar o seguinte comando dentro da pasta onde se encontram os dados e os scripts:

```
./exp1_runner.sh
```

Com isso serão gerados arquivos *.data* que podem ser plotados em um gráfico. Esses arquivos são gerados na pasta *data_plot*

Para plotar e visualizar os gráficos, basta copiar o arquivo *plot* da pasta */programas/gnuplot* para a pasta em que estão os arquivos *.data* gerados. O comando para copiar deve ser executado dentro da pasta *data_plot*:

```
cp ~/grupo5/programas/gnuplot/plot .
```

Em seguida basta realizar o seguinte comando

```
gnuplot plot
```

Com isso deve-s gerar o conjunto de imagens no formato *.png* do experimento 1.

Para executar o experimento 2, os passos são muito semelhantes aos feitos no experimento 1.

Basta mover os arquivos *exp2.py* e *exp2_runner.sh* para a pasta com os dados do experimento 2 usando o comando abaixo dentro da pasta *data_exp2*.

```
cp ~/grupo5/programas/bash/exp2_runner.sh .
cp ~/grupo5/programas/python/exp2.py .
```

Então é preciso rodar o seguinte comando dentro da pasta onde se encontram os dados e os scripts:

```
./exp2_runner.sh
```

Para plotar e visualizar os gráficos, basta copiar o arquivo *plot2* da pasta */programas/gnuplot* para a pasta em que estão os arquivos *.data* gerados. O comando deve ser executado dentro da pasta *data_plot*:

```
cp ~/grupo5/programas/gnuplot/plot2 .
```

Em seguida basta realizar o seguinte comando

```
gnuplot plot2
```

Com isso deve-s gerar o conjunto de imagens no formato *.png* do experimento 2.

8 Replicação do Banco

Após os experimentos acima, é possível verificar o funcionamento do sistema com um único banco. Para configurar uma réplica do Banco, deve ser feita uma cópia dos arquivos do cluster *main* criado.

8.1 Criação do segundo cluster

Inicialmente é preciso logar com o usuário postgres.

```
sudo su - postgres
```

Em seguida deve-se copiar os arquivos do cluster para uma outra pasta com outro nome. Os comandos para isso são:

```
cd /etc/postgresql/9.5/
cp -r main second
```

Em seguida, as configurações do arquivo *postgresql.conf* do banco second (/etc/postgresql/9.5/second/postgres.conf) foram alteradas. As linhas a seguir foram trocadas pelo valor mostrado abaixo.

```
data_directory = '/var/lib/postgresql/9.5/second'
hba_file = '/etc/postgresql/9.5/second/pg_hba.conf'
ident_file = '/etc/postgresql/9.5/second/pg_ident.conf'
external_pid_file = '/var/run/postgresql/9.5-secondpidd.pid'
```

8.2 Configurando o modo Streaming Replication

Inicialmente deve ser criado o ROLE replication no banco *main*. Para isso, deve-se acessar o banco *main* e criar o ROLE. Para criar o ROLE:

```
psql tpcw -p 5432
CREATE ROLE replication WITH REPLICATION PASSWORD 'replication' LOGIN;
```

Após criar o role, deve-se permitir conexões feitas pelo role replication no banco *main*, para isso deve-se acessar a pasta abaixo com o usuário postgres.

```
cd /etc/postgresql/9.5/main
```

Em seguida a linha abaixo deve ser adicionada ao arquivo *pg_hba.conf* para permitir a conexão.

```
host  replication     replication     127.0.0.1/32          trust
```

Também é necessário alterar as configurações do banco *main* no arquivo *postgres.conf*. Para isso basta adicionar as seguintes linhas nesse arquivo.

```
wal_level = hot_standby
max_wal_senders = 32
wal_keep_segments = 32
```

9 Copiando para o banco secundário

O comando abaixo é usado para criar um backup do banco *main* e enviar ao banco *second*

```
pg_basebackup -h 127.0.0.1 -D second --xlog-method=stream --checkpoint=fast -R -U replication -W
```

Após criada a réplica, deve-se configurar as portas no arquivo de configuração */etc/-postgresql/9.5/second/postgres.conf* e verificar se contém as seguintes configurações.

```
data_directory = '/var/lib/postgresql/9.5/second'
hba_file = '/etc/postgresql/9.5/second/pg_hba.conf'
ident_file = '/etc/postgresql/9.5/second/pg_ident.conf'
external_pid_file = '/var/run/postgresql/9.5-secondpidd.pid'
port = 5433
hot_standby = on
```

Em seguida, basta subir o banco *second*, e ele estará funcionando como réplica do banco *main* no modo hot-standby com streaming replication.

Alguns testes podem ser feitos para verificar se tudo ocorreu corretamente. Para testar o funcionamento, pode-se criar uma tabela no banco *main* e verificar se a mesma foi replicada para o banco *second*.

10 Configurando HAProxy

Inicialmente, é preciso baixar a versão correta do HAProxy para o sistema operacional em que será executado. O sistema instalado é o Ubuntu Trusty (14.04 LTS) version 1.6-stable.

```
apt-get install software-properties-common
add-apt-repository ppa:vbernat/haproxy-1.6

apt-get update
apt-get install haproxy
```

Após instalar, é necessário colocar as configurações corretas do HAProxy. Para isso, deve-se entrar em */etc/haproxy/* e alterar o *haproxy.conf* para as configurações abaixo.

```
global

   daemon

   log     /dev/log local0

listen psql

   bind    *:5432

   mode  tcp

    log      global

   option  tcplog

   option  logasap

#   timeout server      (infinite when unset)

#   timeout client      (infinite when unset)

#   timeout connection  (infinite when unset)
```

```
option  pgsql-check user haproxy

    server  replicaOne  127.0.0.1:5432 check

server  replicaTwo  127.0.0.1:5433 backup check

listen stats

   bind    *:9000

   mode  http

   stats   enable

   stats   uri /admin?stats
```

Em seguida é possível acessar o HAProxy e ver o status dos bancos em *http://cbn5.lab.ic.unicamp.br:9000/admin?stats*

11 Configurando Banco Remoto

Nessa etapa, o banco *main* foi colocado no server *DBMaster* e posteriormente o *second* no *DBSlave*.

11.1 Configurando o DBMaster

Para configurar o banco na maquina do *DBMaster*, basta modificar as configurações do banco para obter o restore remoto do banco *main* na máquina cbn5. Para isso, o arquivo */etc/postgresql/9.5/grupo05/* deve ter as seguintes linhas:

```
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
#
host    all             all             10.1.2.2/32             md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
```

```
# replication privilege.
host  replication     replication     10.1.2.20/32         trust
```

Além disso, o banco *main* deve estar configurado para enviar a cópia para o *DBMaster*. A seguinte linha deve ser modificada no arquivo *pg_hba.conf* do banco *main*:

```
host  replication     replication     10.1.2.10/32         trust
```

Em seguida o comando abaixo foi usado para restaurar a cópia do *main* no *DBMaster*.

```
pg_basebackup -p 5432 -h 10.1.2.2 -D grupo05 --xlog-method=stream --checkpoint=fast -R -U replication -W
```

Agora, deve-se verificar se as configurações do *postgresql.conf* do *DBMaster* estão corretas. Para isso deve-se verificar as seguintes linhas no arquivo */etc/postgresql/9.5/grupo05/*

```
data_directory = '/var/lib/postgresql/9.5/grupo05'
hba_file = '/etc/postgresql/9.5/grupo05/pg_hba.conf'
ident_file = '/etc/postgresql/9.5/grupo05/pg_ident.conf'
external_pid_file = '/var/run/postgresql/9.5-grupo05.pid'
listen_addresses = '10.1.2.10'
port = 5435
wal_level = hot_standby
max_wal_senders = 32
wal_keep_segments = 32
```

As configurações do *pg_hba.conf* também devem ser modificadas, para que ele funcione como réplica primária. O arquivo deve conter:

```
host    all             all             10.1.2.2/32             md5
host    all             all             ::1/128                 md5
host  replication     replication     10.1.2.20/32         trust
host    tpcw            tpcw_user       10.1.2.2/32     trust
```

Como o banco *DBMaster* é primário, deve-se remover o *recover.conf* da pasta */var/lib/postgresql/9.5/grupo05/*

11.2 Configurando o Tomcat para Banco Remoto

Ao mudar o local do banco para a máquina *DBMaster*, é necessário trocar a url nas configurações do *tomcat*. Para isso deve-se acessar o arquivo */var/lib/tomcat7/conf/server.xml* e substituir a linha com a url pela abaixo:

```
url="jdbc:postgresql://10.1.2.10:5435/tpcw"
```

Em seguida é possível executar o experimento 1 para o banco remoto sem replicação.

11.3 Experimento no banco remoto

O primeiro experimento realiza o teste no banco remoto sem replicação, e é detalhado a seguir.

11.3.1 Dados experimentos

Para obter os dados do experimento 1 são usados os scripts que automatizam os parâmetros dos RBEs. O script usado é o *data_exp1_arco2.sh*. Para executar, deve-se colocar o caminho para uma pasta com os RBEs (tpcw_rbe).

```
cd CAMINHO
./data_exp1_arco2.sh
```

Com isso cada resultado de cada execução do RBE é colocado em uma pasta com o nome especificado no script.

11.3.2 Gerando os Dados dos Gráficos

Para juntar todos os resultados das execuções em uma mesma pasta, basta executar o comando abaixo colocando o caminho da pasta com os dados do experimento (setado no script).

```
cp CAMINHO/NOMEPASTA-*/*.m DESTINO
```

Para gerar os dados dos gráficos, basta mover os dados da execução do experimento para a pasta destino, usada anteriormente, e em seguida copiar os scripts *exp1_arco2_runner.sh* e *exp1_arco2.py* para o local.

Em seguida é possível executar o script para gerar os dados de plot executando a seguinte linha.

```
./exp1_arco2_runner.sh
```

Após executar, os dados para plotar os graficos serão gerados em uma pasta chamada *data_plot*

11.3.3 Gerando os gráficos

Para gerar as imagens, basta entrar na pasta *data_plot* e copiar o arquivo *plot_exp1_arco2* para a pasta e executar o comando:

```
gnuplot plot_exp1_arco2
```

Com isso as imagens dos gráficos para o experimento serão geradas na pasta.

12 Configurando Réplica Remota

Nessa seção, será configurado a réplica do banco em um outro servidor remoto. Os passos são basicamente os mesmos da replicação local, mudando apenas as configurações.

12.1 Configurando o DBSlave

Uma vez funcionando o DBMaster, pode-se configurar a réplica na maquina *DBSlave*. Uma descrição mais detalhada sobre como configurar uma réplica é feita nas seções 7 e 8.

O arquivo *postgresql.conf* do *DBSlave* deve conter a seguintes configurações:

```
data_directory = '/var/lib/postgresql/9.5/grupo05'
hba_file = '/etc/postgresql/9.5/grupo05/pg_hba.conf'
ident_file = '/etc/postgresql/9.5/grupo05/pg_ident.conf'
external_pid_file = '/var/run/postgresql/9.5-grupo05.pid'
listen_addresses = '*'
port = 5435
hot_standby = on
```

Em seguida, basta executar o seguinte comando para restaurar uma cópia da *DBMaster* na *DBSlave*

```
pg_basebackup -p 5435 -h 10.1.2.10 -D grupo05 --xlog-method=stream --checkpoint=fast -R -U replication -W
```

Em seguida o banco já está configurado como réplica primária do banco na *DBMaster*.

12.2 Configurando o HAProxy

Para que as requisições sejam redirecionadas em caso de falha, é necessário instalar e configurar o HAProxy. A instalação está descrita na secção 9.

Para os bancos remotos, o arquivo de configuração *haproxy.conf* em */etc/haproxy/* deve conter os seguintes parâmetros.

```
bind    *:5435
server  replicaOne  10.1.2.10:5435 check
server  replicaTwo  10.1.2.20:5435 backup check
```

Em seguida, o Tomcat deve ser configurado para receber as informações do banco no endereço local (cbn5) através da porta 5435.

12.3 Configurando o Tomcat para Banco Remoto

Ao mudar o local do banco para as máquinas dbmaster e dbslave, é necessário trocar a url nas configurações do tomcat.

Para isso deve-se substituir a linha do arquivo */var/lib/tomcat7/conf/server.xml* pela abaixo:

```
 url="jdbc:postgresql://127.0.0.1:5435/tpcw"
```

Algumas outras configurações do tomcat devem ser adicionadas, caso o número de RBEs utilizados seja muito alto. Mais detalhes são encontrados em:
http://www.tomcatexpert.com/blog/2010/04/01/configuring-jdbc-pool-high-concurrency

Em seguida é possível executar o experimento 2 para o banco remoto com replicação.

12.4 Experimento com Replicação Remota

O segundo experimento efetua o teste no banco com replicação para comparar o efeito da réplica nos tempos de WIPS e WIRT.

12.4.1 Dados do experimento

Para obter os dados do experimento 2, é usado o mesmo script usado no experimento anterior. O script é o *data_exp1_arco2.sh*. Para executar, basta colocar o caminho para uma pasta com os RBEs (tpcw_rbe).

```
cd CAMINHO
./data_exp1_arco2.sh
```

12.4.2 Gerando os gráficos

Para gerar os dados dos gráficos, também são feitos os mesmos passos do experimento anterior. Basta mover os dados do experimento para uma pasta e em seguida copiar os mesmos scripts usados anteriormente, chamados *exp1_arco2_runner.sh* e *exp1_arco2.py* para o local.

Em seguida deve-se executar o script para gerar os dados de plot.

```
./exp1_arco2_runner.sh
```

Após executar, os dados para plotar os graficos serão gerados em uma pasta chamada *data_plot*

Para gerar as imagens, basta entrar na pasta *data_plot* e copiar o arquivo *plot_exp1_arco2* para ela, e em seguida executar o comando:

```
gnuplot plot_exp1_arco2
```

Com isso as imagens dos gráficos do experimento serão geradas na pasta.

13 Injeção da falha no banco primário

Nessa seção, estão os passos para injetar uma falha no banco primário para testar a alta disponibilidade do sistema.

13.1 Configurando chave SSH

Para executar os comandos no dbmaster2 e dbslave2 via ssh, é necessário configurar uma ssh na cbn5. Para isso deve-se criar uma chave publica com o RA do líder do grupo. Basta executar os comandos abaixo, em que XXXXXX é o RA do líder do grupo, ou usuário com acesso aos servidores dbmaster2 e dbslave2.

```
ssh-keygen -t rsa

cat .ssh/id_rsa.pub | ssh raXXXXXX@10.1.2.20 'cat >> .ssh/authorized_keys'

cat .ssh/id_rsa.pub | ssh raXXXXXX@10.1.2.10 'cat >> .ssh/authorized_keys'
```

13.2 Otimização Tomcat

Para aguentar mais conexões, o Tomcat deve ser usado com as configurações abaixo.

```
    <Resource type="javax.sql.DataSource"
              name="jdbc/TPCWPool"
              factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
              driverClassName="org.postgresql.Driver"
              url="jdbc:postgresql://127.0.0.1:5435/tpcw"
              username="tpcw_user"
              password="tpcw"
              testOnBorrow="true"
              validationQuery="SELECT 1"
              validationInterval="1000"

             maxActive="2000"
             />
  </GlobalNamingResources>
```

13.3 Otimização HAProxy

Outra configuração utilizada para aguentar mais clientes e diminuir o tempo de indisponibilidade foi colocada no haproxy. O arquivo */etc/haproxy/haproxy.cfg* deve ter as seguintes configurações.

```
   timeout server 1s

   timeout client 1s

   timeout connect 1s

    timeout check 5s

    option  pgsql-check user haproxy

   server  replicaOne  10.1.2.10:5435 check inter 1s

   server  replicaTwo  10.1.2.20:5435 backup check inter 1s
```

```
maxconn 200
```

13.4 Experimento com injeção de falha

Esse terceiro experimento realiza os testes com a injeção de falha no banco primário.

13.5 Dados experimentos

Para obter os dados desse experimento, é usado o script *data_exp3_arco2.sh*. Esse script deve ser modificado para os parâmetros desejados para os RBEs.

Para executar esse experimento, deve-se colocar o script no mesmo diretório dos RBEs, chamado tpcw_rbe. O script *analyze.py* pode ser colocado na pasta para já gerar os dados durante a execução do experimento.

```
cd CAMINHO
./data_exp3_arco2.sh
```

Durante a execução do experimento, deve-se injetar a falha no servidor primário. Isso é feito executando o script *kill_master.sh*.

Esse script pode ser programado para ser executado em um tempo programado, para isso basta executar o seguinte comando, em que X é o tempo para injetar a falha no banco primário.

```
sleep X; ./kill_master.sh
```

Esse script deve ser usado pelo líder, pois requer uma conexão ssh do usuário com acesso aos bancos dbmaster2 e dbslave2

13.6 Gerando os gráficos

Para gerar o gráfico, basta executar o script *analyze.py* e em seguida executar o gnuplot com o script *plot_exp3_arco2.plot*.

```
python analyze.py
gnuplot < plot_exp3_arco2.plot
```

O gráfico será gerado na pasta.

13.7 Reiniciando o Experimento

Para reiniciar o experimento pode-se utilizar o script *reup_master.sh* executando o seguinte comando:

```
./reup_master.sh
```

Com isso o banco primário estará restaurado como primário e o secundário como réplica deste banco, sendo possível refazer o experimento.

14 Detecção e correção automática de falha

Nessa seção, estão explicitados os mecanismos utilizados para implementar um sistema composto por 2 instâncias do postgres que se alternam entre primário e secundário, detectando e se recuperando automaticamente de falhas.

14.1 Detecção de falhas

Para a detecção de falhas, foi utilizado um script em python que monitora o estado dos bancos através do file socket linux do HAProxy. Esse script python se conecta ao socket, e faz polling do estado dos bancos. Caso seja detectada falhas, chama os scripts bash de acordo com a necessidade. Para executá-lo é necessário ter poderes de sudo:

```
sudo ./fail-detector.py
```

O programa acessa o haproxy se conectando com:

```
s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
s.connect("/var/run/haproxy.sock")
```

Dado que o arquivo de socket do haproxy está em "/var/run/haproxy.sock". Para verificar o estado, manda o comando "show stat"para o socket e faz o parse dos dados CSV retornados pelo haproxy.

14.2 Correção de falhas

A correção de falhas se divide em várias etapas. O fail-detector, ao detectar uma falha, age de acordo com a configuração atual do cluster. A única falha corrigível e detectável pelo script é a falha de um banco primário, o banco de backup (ou em hot-standby) não é monitorado contra falhas.

14.2.1 Falha no banco primário

Caso seja o dbmaster2 que falhou é chamado o script: promote_slave.sh, ou o promote_master.sh, caso contrário. Esse script basicamente coloca um arquivo no /tmp sinalizando que o banco secundário deve se tornar primário. Então, roda-se o script reup_master.sh e logo em seguida reup_master_sudo.sh.

14.2.2 Postgres rewind/checkpoint

Os comandos de rewind e checkpoint são utilizados nos arquivos reup_master.sh. Para utilizar o rewind, foi necessário definir algumas configurações no postgres.conf:

```
full_page_writes = on # recover from partial page writes
wal_log_hints = on # also do full page writes of non-critical updates
```

Feito isso, no script de reup, é feito um start/stop no banco que sofreu queda para poder estar no estado clean. Então, é feito um comando de checkpoint no banco que não sofreu queda com o seguinte comando:

```
export PGPASSWORD='1234'; psql -U ra147775 -p 5435 -d tpcw -c 'CHECKPOINT;'
```

Logo na sequência é feito um rewind do banco que caiu com o banco que virou principal com o seguinte comando:

```
/usr/lib/postgresql/9.5/bin/pg_rewind --source-server=\"user=ra147775 password=1234 host=10.
```

Por fim, se coloca um recovery.conf no server que caiu, pois ele será o backup do banco promovido agora. Isso é feito com o comando:

```
cat << EOF > /var/lib/postgresql/9.5/grupo05/recovery.conf
standby_mode = 'on'
recovery_target_timeline='latest'
primary_conninfo = 'user=replication password=replication
host=10.1.2.20 port=5435 sslmode=prefer sslcompression=1
krbsrvname=postgres'
trigger_file = '/tmp/postgres9.5-grupo05'
EOF
```

14.2.3 Trocando papéis no HAProxy

Para trocar os papéis dos bancos no haproxy, isso é, passar de principal para backup e vice-versa, foi utilizado um jeito de não precisar restartar o server toda vez. Dessa forma, nós carregamos em tempo de execução uma nova configuração no HAProxy. Isso é feito da seguinte forma:

```
cp haproxy_second.cfg /etc/haproxy/haproxy.cfg
haproxy -f /etc/haproxy/haproxy.cfg \
-p /var/run/haproxy.pid -sf $(cat /var/run/haproxy.pid)
```

onde haproxy_second.cfg é um arquivo já preparado que está na pasta do script.

14.2.4 Permissões dos scripts

Como o script fail-detector.py deve ser executado como sudo e os scripts auxiliares são chamados à partir dele, todos os scripts seriam executados como sudo. Isso não é interessante, pois o ssh nas máquinas dbmaster2 e dbslave2 só são dadas à partir do usuário ra147775, para contornar isso, os scripts que precisam executar comandos pelo ssh na máquina são executados com:

```
sudo su - ra147775 -c "comando"
```

Isso garante que serão executados como ra147775, desde que o usuário que executa o fail-detector tenha permissões de usar sudo.