UNICAMP

Universidade Estadual de Campinas

Manual

Autores

Gabriel Bueno de Oliveira 139455
Joao Guilherme Daros Fidelis 136242
Lucas Henrique Morais 136640
Matheus Yokoyama Figueiredo 137036
Pedro Rodrigues Grijo 139715

1 Setup do Cluster

1.1 Acesso

O acesso ao cluster pode ser feito de fora e de dentro do IC (Instituto de Computação - UNICAMP).

1.1.1 Interno ao IC

Para acceso interno ao ic, basta acessar por ssh o endereço cbn6.lab.ic.unicamp.br. No linux, isso pode ser feito através de:

```
# ssh cbn6.lab.ic.unicamp.br
```

Após a conexão ser feita, insira sua senha.

1.1.2 Externo ao IC

Para o acesso externo, é necessário fazer uma conexão de duas fases ou usar um tunnel por uma máquina do ic. Para acesso remoto a uma máquina do ic, no linux, utilize **um** dentre os seguintes comandos:

```
# ssh -l raXXXXXX ssh.students.ic.unicamp.br
# ssh -l raXXXXXX ssh2.students.ic.unicamp.br
```

Após a conexão ser feita, insira sua senha. Após feita a credencial, prossiga do mesmo modo como se tivesse acessando de dentro do IC.

2 Instalação e Configuração

2.1 PostgreSQL

2.1.1 Instalação

A versão utilizada do PostgreSQL é a versão 9.5, que pode ser obtida seguindo os seguintes comandos:

```
# sudo su
# cat /etc/apt/sources.list.d/postgresql.list
# deb http://apt.postgresql.org/pub/repos/apt/ trusty-pgdg
main
# sudo apt-get install wget ca-certificates
# wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc
| sudo apt-key add -
# sudo apt-get update
# sudo apt-get upgrade
# sudo apt-get install postgresql-9.5
```

2.1.2 Configuração

Para configurar o acesso ao banco de dados deve-se alterar o arquivo de configuração principal. Nele são definidos toda a configuração de acesso ao banco de dados e os diretórios de dados que são respectivamente:

```
hba_file = '/etc/postgresql/9.5/main/pg_hba.conf'
data_directory = '/var/lib/postgresql/9.5/main'
```

O arquivo pg_hba.conf define regras para a conexão ao banco. Por padrão, são só permitidas conexões locais: o usuário administrador do banco (postgres, quando instalado por gerenciador de pacotes) tem acesso via socket e sem senha. os demais usuários, via interface local e com senha:

```
local   all         postgres            peer

host    all         all         127.0.0.1/32            md5
```

Para criar o primeiro usuário siga os seguintes comandos:

```
# sudo su
# su postgres
# psql
=# CREATE ROLE <usr_name> with superuser createdb createrole
login;
=# ALTER ROLE <usr_name> PASSWORD '<password>';
```

2.1.3 Carregando o Dump

Para carregar o ddump no banco deve-se criar o banco de dados (tpcw) e um usuário tpcw-user.

```
# pg_restore -d tpcw tpcw-database-dump.tar
```

2.2 Tomcat

2.2.1 Instalação

Para instalar o Tomcat (usaremos o tomcat7), precisamos instalar o pacote do tomcat e suas dependencias. Para isso, no linux, podemos usar o apt-get:

```
# sudo apt-get install tomcat7
```

A instalação irá criar um usuário tomcat7 dentro de um grupo, tomcat7. Os arquivos estão em $CATALINA_BASE, que na instalação do Ubuntu é /var/lib/tomcat7.

Para teste, acesse em uma máquina do IC o cbn6.lab.ic.unicamp.br:8080.

```
<Resource type="javax.sql.DataSource"
              name="jdbc/TPCWPool"
              factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
              driverClassName="org.postgresql.Driver"
              url="jdbc:postgresql://localhost:4002/tpcw"
              username="tpcw_user"
              password="123456"
              initialSize="10"
              maxActive="500"
              maxIdle="100"
              minIdle="10"
              validationQuery="SELECT 1"
              logValidationErrors="true"
              logAbandoned="true"
              validationQueryTimeout="5"
              validationInterval="500"
              removeAbandoned="true"
              removeAbandonedTimeout="5"/>
```

Figura 1: Configurações do Tomcat no Arco 03

2.2.2 Integração com o PostgreSQL

Para integração com o PostgreSQL é necessário fazer o download do .jar do PostgreSQL JDBC para ser adicionado a pasta $CATALINA_BASE/lib. Após fazer o download, modifique os seguintes arquivos com o conteúdo indicado:

- $CATALINA_BASE/conf/context.xml: http://www.ggte.unicamp.br/eam/pluginfile.php/188108/mod_wiki/attachments/132/tomcat7-context.xml
- $CATALINA_BASE/conf/server.xml: http://www.ggte.unicamp.br/eam/pluginfile.php/188108/mod_wiki/attachments/132/tomcat7-server.xml. Para este, modifique os campos de username, password e possivelmente url.

Após essas configurações, reinicie o tomcat:

```
# invoke-rc.d tomcat7 restart
```

2.2.3 Arco 03

Devido a problemas relacionados à pool de conexões quando há uma queda de um dos bancos, houve a necessidade de se checar a validade de cada conexão do pool. Para isto, novas configurações foram usadas para o Tomcat JDBC, como podem ser vista na Figura 1 acima. Nota-se que as flags de log foram usadas apenas por motivos de depuração.

2.3 TPC-W

As aplicações implantadas no Tomcat são configuradas em $CATALINA_BASE/webapps. Para ter permissão de criar aplicações nesta pasta, o usuário tem que ser do grupo tomcat7, para fazer isso, no linux:

```
# adduser [usuário_comum] tomcat7
```

Para instalação default do TPC-W no tomcat7, crie a seguinte árvore:

```
\$CATALINA_BASE/webapps

    tpcw

        Images

        WEB-INF

            classes

            web.xml
```

Na pasta tpcw, rode os seguintes comandos:

```
# sudo cp /nfs/servlet/Images.tar.gz .
# sudo tar -xzf Images.tar.gz
# sudo cp /nfs/servlet/tpcw-servlets.tar.gz
# sudo tar -xzf tpcw-servlets.tar.gz
# cd tpcw-servlets.tar.gz
# sudo make
```

No fim, copie todas as *.class geradas para a pasta WEB-INF/classes e copie o web.xml para pasta WEB-INF/. Reinicie o Tomcat.

2.4 RBE

O RBE (Remote Browser Emulator) é o programa empregado para gerar carga para o TPC-W. É um aplicativo que emula um conjunto de clientes que acessam o lado servidor do TPC-W, que implementa uma loja de livros. Os clientes operam como se fossem navegadores web. O código fonte do RBE, assim como arquivo README e o script em Python que realiza a análise das saídas estão disponíves na pasta do RBE.

2.4.1 Automatização

Para facilitar a coleta dos dados foi implementado um script em bash que executa o rbe com variando os parâmetros e as saídas, já executa o script analyse.py e gera os gráficos. Para executar o script basta executar:

```
# ./script.sh
```

3 Entrando e conhecendo as novas máquinas

Obtivemos acesso a duas máquinas diferentes do cluster que contém uma instância do postgres cada, assim, uma máquina conterá o banco primário e a outra o secundário.
O banco primário fica na máquina dbmaster2. Essa máquina tem IP 10.1.2.10 e o banco fica na porta 5434.
Já o banco secundário fica na máquina dbslave2. Essa máquina tem IP 10.1.2.20 e o banco fica na porta 5434.

Para acessar o dbmaster2, basta enquanto logado no cluster cbn6, dar ssh no seu endereço, assim:

```
# ssh dbmaster2
```

Para acessar o dbslave2, basta enquanto logado no cluster cbn6, dar ssh no seu endereço, assim:

```
# ssh dbslave2
```

As máquinas já vieram com a instalação do PostgreSQL 9.5.1.

As pastas de dados para os bancos de dados em ambas as máquinas fica em `/var/lib/postgresql/9.5/grupo06/`.

As pastas de configurações (como postgres.conf e pg_hba.conf), ficam em `/etc/postgresql/9.5/grupo06/`.

Também é necessário configurar e iniciar um banco de dados inicial no banco primário em dbmaster2.

Faça login na máquina dbmaster2 e crie um usuário "tpcw-user"como é ensinado na seção 2.1.2 e faça uma restauração do dump como na seção 2.1.3.

4 Instalando a Replicação

Era necessário instalar o serviço de Streaming Replication do banco primário para o secundário, para que mudanças feitas no banco primário sejam copiadas para o secundário. Para isto, seguimos tutoriais sugeridos pelo professor. Agora, temos dois bancos de dados em duas máquinas diferentes.

4.1 Editar configurações no mestre

Na máquina do banco primário, dbmaster2, vamos criar um usuário "replication"no postgres, que é um usuário com privilégios de replicação. O usuário terá a senha 123456. Para isso, execute os comandos:

```
# sudo su
# su postgres
# psql
# CREATE ROLE replication WITH REPLICATION PASSWORD ‘123456’
LOGIN
```

Agora, vamos editar o arquivo /etc/postgresql/9.5/grupo06/postgresql.conf da máquina dbmaster2. Na seção **CONNECTIONS AND AUTHENTICATION**, adicione a linha:

```
listen_addresses = '*'
```

Na seção **WRITE AHEAD LOG**, adicione as linhas:

```
wal_level = hot_standby
```

Em **REPLICATION**, coloque as linhas:

```
max_wal_senders = 5
wal_keep_segments = 32
```

Finalmente, em **ARCHIVING**, adicionamos as linhas:

```
archive_mode = on
archive_command = 'test ! -f /var/lib/postgresql/9.5/grupo06/archivedir/\%f
&& cp \%p /var/lib/postgresql/9.5/grupo06/archivedir/\%f'
```

Depois, crie a pasta do comando acima com:

```
# mkdir /var/lib/postgresql/9.5/grupo06/archivedir/
```

Veja que a pasta /var/lib/postgresql/9.5/grupo06/archivedir/ será criada e nela ficará contido todos os arquivos log gerados pelos bancos de dados. Agora, temos que editar o arquivo /etc/postgresql/9.5/grupo06/pg_hba.conf. Adicione a linha no final:

```
host    replication     replication     10.1.2.20/32    md5
```

Essa linha permite que o banco em standby consiga se conectar com o primário para fazer a replicação.

4.2 Fazer base backup

Devemos fazer um base backup do servidor mestre para o escravo. Utilizaremos o comando pg_basebackup.
Para isso, faça login na máquina do escravo, a dbslave2.
Primeiro, a pasta de dados do banco de dados que receberá o base backup deve estar vazia. Logo, apagaremos tudo dentro dela. Depois, utilizaremos o comando de base backup para se conectar no banco de dados mestre e fazer a cópia.
Rode o comando:

```
rm /var/lib/postgresql/9.5/grupo06/* -r
pg_basebackup -h 10.1.2.10 -p 5434 -D /var/lib/postgresql/9.5/grupo06
-P -U replication --xlog-method=stream -R
```

Ao ser pedido uma senha, digite 123456.
A flag -R no final, faz com que o arquivo recovery.conf seja gerado automaticamente no processo.

4.3 Configurar o slave

Na máquina escrava, edite o arquivo `/etc/postgresql/9.5/grupo06/postgresql.conf`, adicionando as mesmas linhas que fez na máquina mestre, como nos passos anteriores. Isso é necessário pois quando o primário falhar, o secundário deve se comportar exatamente como o primário. Entretanto, é necéssario adicionar mais uma linha, na seção **STANDBY SERVERS**, adicione a linha:

```
hot_standby = on
```

Isso faz com que o banco escravo seja read-only e aceite apenas comandos de leitura.

4.4 Arco 03

Para as configurações de replicação do postgres, foi adicionado ao arquivo recovery.conf a seguinte linha:

```
recovery_target_timeline = 'latest'
```

Assim, após a queda, promoção e possíveis correções de consistência pelo `pg_rewind`, o banco que agora seria o secundário seguiria pegando na mais nova timeline criada pelo banco promovido, agora master.

Para habilitar o `pg_rewind`, precisamos fazer modificações no arquivo `/etc/postgresl/9.5/grupo06/postgresql.conf` das duas máquinas, dbmaster2 e dbslave2. Adicionamos na seção WRITE AHEAD LOG a linha:

```
wal_log_hints = on
```

5 Configuração do HAProxy

5.1 Instalação

Para instalar o HAProxy, tivemos apenas que adicionar ao nosso ambiente de execução o repositório apt-get relacionado e rodar o comando apt-get install apropriado, como segue:

```
# sudo add-apt-repository ppa:vbernat/haproxy-1.6
# sudo apt-get update
# sudo apt-get install haproxy
```

Podemos verificar que a versão correta foi então instalada com o seguinte comando:

```
# haproxy -v
```

5.2 Configuração

Uma vez instalado o HAPROXY, temos ainda de configurá-lo de modo a:

1. Informá-lo da existência de dois endereços de destino distintos
2. Permitir que seu backend escolha o endereço de destino adequado de acordo com o resultado de um health-check periódico de cada um dos alvos
3. //Requisitar que o HAProxy escreva mensagens de log relevantes em /var/log

Para tal, é necessário criar em `/etc/haproxy` um arquivo haproxy.cfg com parâmetros adequados. A seguir, comentamos o conteúdo de um arquivo mínimo desse tipo que é capaz de prover o requerido em (1) e (2):

```
 listen psql

bind    *:4002

mode  tcp

option  pgsql-check user haproxy

server  replicaOne  10.1.2.10:5434 check

server  replicaTwo  10.1.2.20:5434 check backup
```

A linha 02 garante que toda a comunicação ocorrendo através da porta 4002 deverá ser interceptada.

A linha 03 indica que o haproxy deverá funcionar como um tcp proxy – isto é, transparentemente interceptando e redirecionando pacotes do tipo TCP.

A linha 04 configura o haproxy para fazer contatos de health-check com os bancos replicados usando o usuário haproxy.

As linhas 05 e 06 definem as réplicas do banco como os dois destinos possíveis para os pacotes TCP interceptados. Além disso, a opção "backup" da linha 06 garante que o servidor replicaTwo não seja contatado antes que replicaOne falhe.

Por fim, as opções "check" ocorrendo nas duas linhas indicam que haproxy deverá verificar periodicamente a disponibilidade dos dois bancos, alternando o destino dos pacotes interceptados de acordo com essa avaliação.

5.3 Startup, restart e shutdown

Para realizar o startup, restart e shutdown do HAProxy, respectivamente, basta usar o comando "service" do Linux, como segue:

```
sudo service haproxy start
sudo service haproxy restart
sudo service haproxy stop
```

5.4 Arco 03

Como a detecção de falhas foi passada para um novo script(descrito na seção de Detector de falhas abaixo), o check do haproxy se tornou obsoleto, servindo apenas para checagem na GUI no site disponibilizado pelo haproxy.

Então as configurações de haproxy foram separadas em dois arquivos, onde um deles aponta para o dbmaster2 e outro para dbslave2. Quem ficou resposável pela troca de configuração e religar o haproxy também ficou por parte do detector de falhas.

6 Detector de falhas

Para a implementação do detector de falhas bolamos um script em bash que executa uma consulta em loop dos status dos banco de dados. Ao detectar que algum dos bancos de dados caiu ele automaticamente, se for o primário, altera as configurações do haproxy e realiza a promoção do banco de dados secundário para primário e restaura o banco que estava indisponível como secundário. Para a execução do script deve-se estar logado com um usuário pertencente ao grupo e executar o seguinte comando:

```
# ./health_checker.sh
```

7 Automatização da coleta de dados

Todos os scripts mencionados nessa seção estão na pasta `scripts`.

A segunda parte do experimento foi dividida em 3 partes. Para executar os testes da primeira parte, foram gerados os scripts `arc2_exp1_script.sh` e `arc2_exp_analise.sh` que devem ser executados da seguinte forma:

```
# ./arc2_exp1_script.sh
# ./arc2_exp_analise.sh 1
```

O argumento passado para o segundo script é simplesmente para geração de arquivos com o nome desejado.

Para executar os testes da segunda parte, foi gerado o script `arc2_exp2_script.sh`. Seguem os comandos para executar:

```
# ./arc2_exp2_script.sh
# ./arc2_exp_analise.sh 2
```

Para executar os testes da terceira parte, foi gerado o script `arc2_exp3_script.sh`. Seguem os comandos para executar

```
# ./arc2_exp3_script.sh
# mkdir ../rbe/arc2/exp3/analise
# python ../rbe/ananlyse.py <outs_rbe>
```

7.1 Arco 03

O programa necessário para executar o experimento se encontra em /ra137036/grupo06/programas e para a execução verifique que não há nenhum `health_checker.sh` rodando.
Para executar o experimento do arco 3 basta executar o seguinte comando:

```
# ./arc3.sh
```

O script se encarrega de incializar os bancos de dados nas configurações iniciais, chamar o `health_checker.sh` (Script que verifica os status do banco e realiza a promoção automática) e o `killdb.sh` (Script que mata o primário 2 vezes) para a realização dos testes. Os arquivos de saída do rbe são encontrados na pasta /ra137036/arc3/ da máquina cbn7. Vá para o diretório /ra137036/grupo06/resultados e execute o seguinte script:

```
# ./runAnalyse.sh
```

Para gerar os gráficos execute:

```
# gnuplot plot_graphs.plot
```

Os gráficos estarão disponíveis na pasta arc3/graphs.