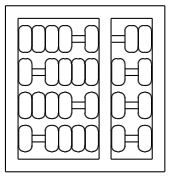

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

MC437 - Grupo06 - Relatório 3

Gabriel Oliveira *João Fidélis* *Lucas Morais*
Matheus Figueiredo *Pedro Grijó*

Technical Report - IC-16-03 - Relatório Técnico

June - 2016 - Junho

MC437 - Grupo06

Gabriel Bueno de Oliveira João Guilherme Daros Fidélis
Lucas Henrique Morais Matheus Yokoyama Figueiredo
Pedro Rodrigues Grijó

Resumo

Este é o relatório final do projeto da disciplina MC437 (Projeto de Sistemas de Informação), projeto cujo objetivo foi a resolução de um problema de replicação de banco de dados. Foram utilizados dois bancos de dados diferentes, um atuando como primário e outro como secundário. O primário é replicado para um secundário em hot-standby. Ao inserirmos uma falha neste, promovemos o secundário para primário e fazemos o antigo primário voltar como secundário, sendo que o objetivo é deixar o menor tempo de indisponibilidade possível no sistema. Utilizamos o benchmark TPC-W, que modela uma livraria online, através de um ambiente controlado, para simular atividades num servidor WEB. Em conjunto com o simulador RBE, que gera três diferentes perfis de carga (Shopping, Ordering e Browsing), pudemos checar o desempenho do servidor instalado num cluster no IC, ao verificarmos o número de WIPS (WEB Interactions per Second) e WIRT (WEB Interaction Response Time) gerados por diferentes cargas.

1 Introdução

Este é o relatório final do projeto da disciplina MC437 - Projeto de Sistemas de Informação. No projeto, preparamos uma máquina virtual remota com um servidor web Apache Tomcat, utilizamos duas outras máquinas remotas com instâncias de bancos de dados PostgreSQL, uma com um banco atuando como primário e outra com um banco secundário inicialmente em hot-standby. Ao integrarmos essas funcionalidades, conseguimos emular um site de compras com dados gerados aleatoriamente.

Na máquina do web server foi instalado o aplicativo TPC-W, um benchmark de transações web. Para fazer uso do TPC-W, foi utlizado o RBE (Remote Browser Emulator), que emula conjuntos de clientes que acessam o lado servidor do TPC-W, sendo que este implementa uma livraria digital.

O RBE é um simulador escrito completamente em Java que simula o tráfego HTTP gerado por um usuário que estivesse acessando o site através de um navegador.

Também utilizamos o HAProxy, que atua como um proxy para aplicações baseadas em TCP e HTTP. Ele oferece alta disponibilidade e balanceamento de carga para servidores web. Neste trabalho, sua função é de detectar falha do banco primário e redirecionar as requisições feitas para o banco secundário, que deve ser promovido a banco primário.

O TPC-W gera um valor que indica a quantidade de WIPS (Web Interactions per Second). O fluxo de trabalho é gerado pelo RBE e pode ser de três perfis. O perfil de

compras (*shopping*), onde 80% das ações são de leitura e 20% de escrita no banco de dados. O perfil de navegação (*browsing*), tem 95% das ações de leitura e 5% de escrita. Já o perfil de compras (*ordering*) tem 50% de suas operações de leitura e 50% de escrita.

Rodamos o RBE com os três perfis variando o parâmetro Think-Time. O Think-Time é o tempo que um usuário simulado pelo RBE gasta "pensando", ou seja, um menor tempo de Think-Time faz com que mais requisições sejam enviadas ao servidor pelos usuários no mesmo período de tempo de experimento.

2 Condições Experimentais

Nesta seção serão descritas as configurações de hardware e software utilizadas nos experimentos.

Nesta fase do trabalho contamos com quatro máquinas, todas com a mesma configuração de hardware, fornecidas pelo Instituto de Computação.

A primeira, denominada CBN6, é onde estão instalados o HAProxy 1.6, Apache Tomcat versão 7 e TPC-W. A máquina CBN6 atua como servidor web.

Duas máquinas, a dbmaster2 e dbslave2, é onde ficam os bancos de dados primário e secundário, respectivamente. Nelas está instalado o PostgreSQL versão 9.5.1.

Já o RBE foi rodado a partir de outra máquina, a CBN7.

Todas as máquinas tem a mesma configuração: sistema operacional Ubuntu 14.04, CPU Intel(R) Core(TM)2 Quad CPU Q8400 2.66GHz e memória RAM de 4GB e 1333 MHz.

3 Metodologia de Pesquisa

No início, após todo o sistema estar rodando de maneira estável, realizamos testes variando as cargas para determinar uma carga que o sistema consegue atender.

Foram executados vários experimentos, mudando o valor de Think-Time, com o objetivo de vermos o desempenho do sistema ao lidar cada vez mais requisições.

A carga escolhida e fixada foi de 1000 clientes, pois essa carga foi testada e analisamos

No primeiro experimento, o RBE foi utilizado com apenas o banco primário ligado. No segundo experimento, estavam ligados o banco primário e o banco secundário em hot-standby.

Esses dois experimentos foram feitos para caracterizar o servidor. O objetivo foi medir a maior carga para a qual ele se comporta de forma consistente.

Para isso, variamos os valores de carga entre 2000 e 4000 clientes, variando de 1000 em 1000 para cada iteração para cada perfil de usuário descrito anteriormente. Assim, gráficos foram gerados para demonstrar até qual carga o servidor se manteve num bom nível de WIPS ao longo de todo o teste (que dura 100s no total).

O valor encontrado foi utilizado no RBE para simular essa "carga ótima"no servidor, causando um failover manualmente (isto é, matando o banco de dados primário), enquanto o banco de dados secundário era promovido, para ver em quanto tempo o servidor conseguia se recuperar e voltar a ficar ativo.

É importante ressaltar que o RBE sempre foi rodado de outra máquina remota (CBN7), diferente de da máquina do servidor(CBN6), e que os parâmetros utilizados foram Ramp-Up Time: 5s, Ramp-Down Time: 5s e Measurement Interval: 90s. O valor de máximo número de erros tolerados foi colocado em 0. Todos os outros valores são os padrões do RBE.

4 Análise e Resultados

Seguem os gráficos gerados pela execução do RBE para o experimento 1.

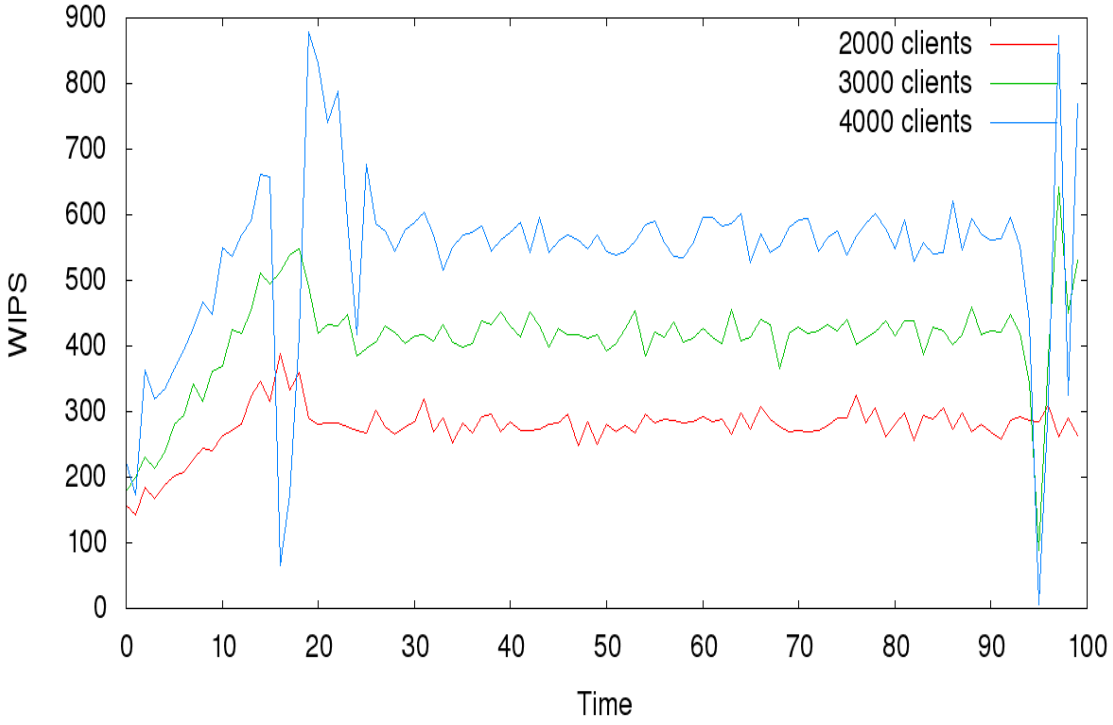

Figura 1: WIPS por tempo(s) para cada carga simulada pelo RBE no perfil Browsing do experimento 1

Nota-se que o experimento com 4000 clientes teve uma boa consistência entre 30 e 90 segundos, porém houve uma grande queda por volta dos 15s e a partir dos 90s. O mesmo ocorreu no caso para 3000 clientes a partir de 90s.

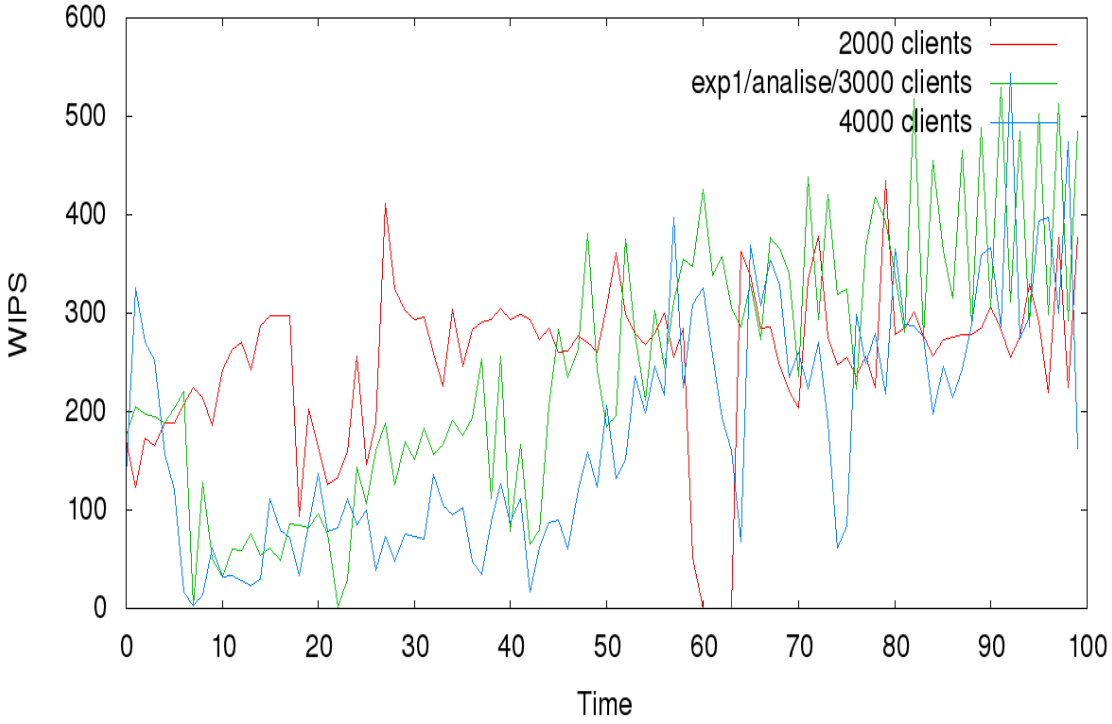

Figura 2: WIPS por tempo(s) para cada carga simulada pelo RBE no perfil Ordering do experimento 1

Pelo gráfico, é visível que o teste com 3000 clientes obteve o melhor desempenho no geral. Até 50s, o desempenho era melhor com 2000 clientes, porém após esse tempo o teste com 3000 clientes obteve uma quantidade maior de WIPS e se manteve consistentemente melhor que os testes com 2000 e 4000 clientes.

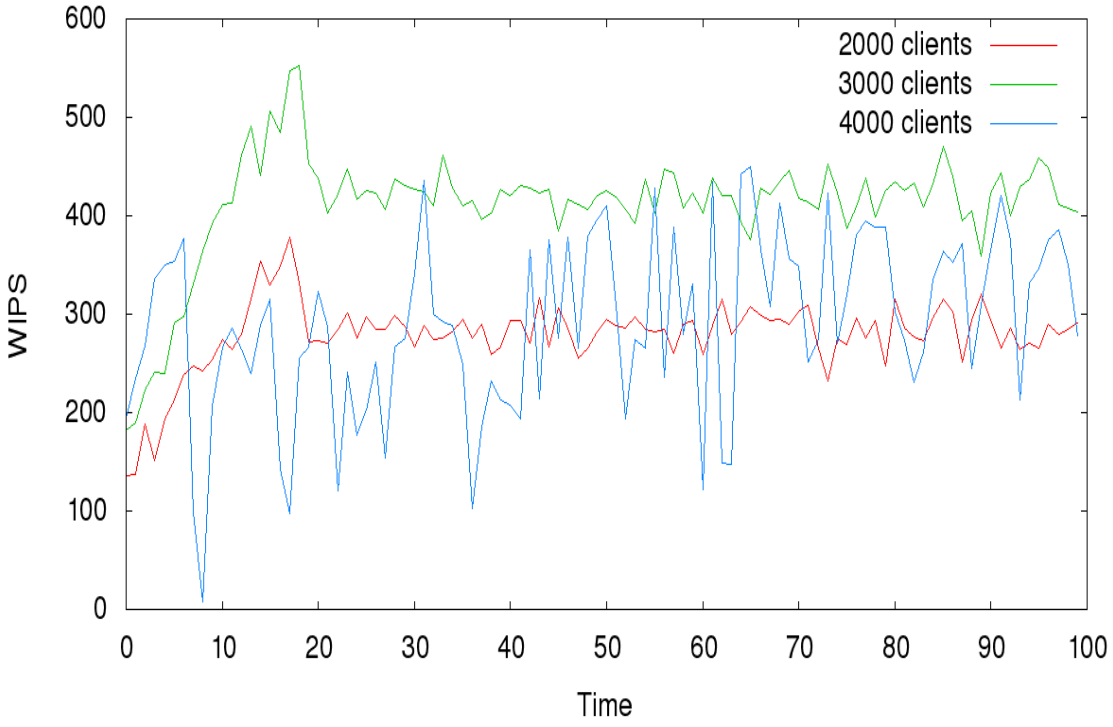

Figura 3: WIPS por tempo(s) para cada carga simulada pelo RBE no perfil Shopping do experimento 1

Neste gráfico, ficou evidente o melhor desempenho da simulação com 3000 clientes sobre as duas outras. Uma melhor taxa de WIPS foi obtida durante praticamente todo o experimento.

Com a combinação dos resultados desses gráficos, temos indícios de que 3000 usuários talvez seja o melhor número a ser utilizado, pois tem taxas de WIPS mais altas e estáveis que as outras quantidades de usuários testadas.

Agora, analisaremos os dados do experimento 2.

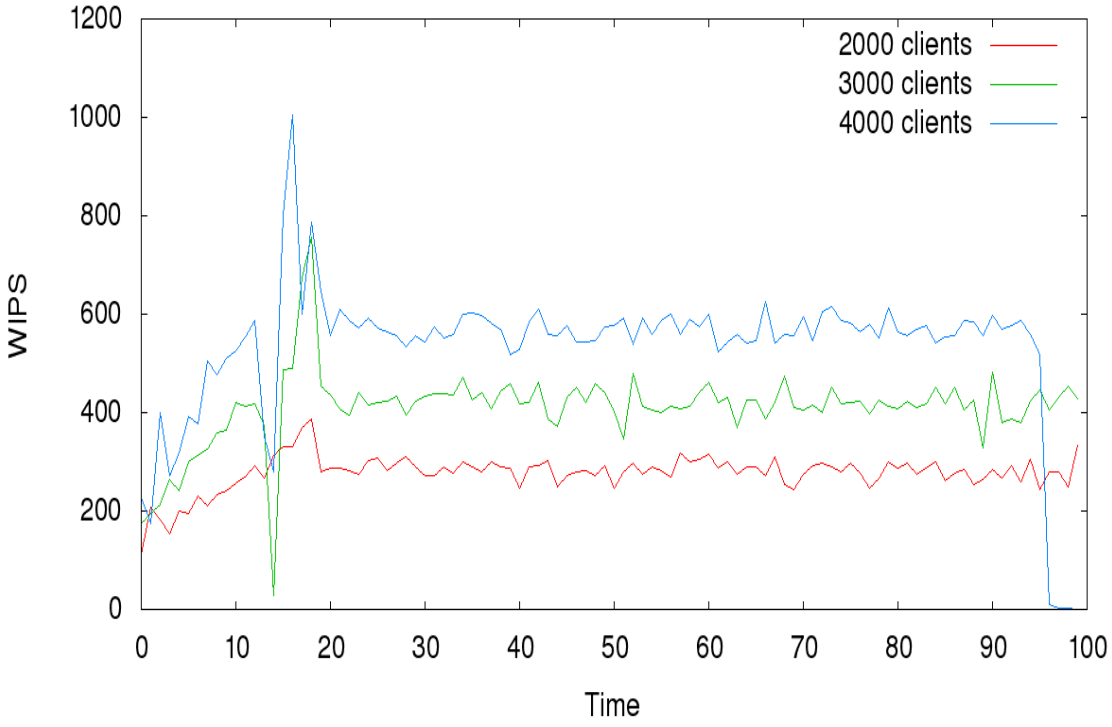

Figura 4: WIPS por tempo(s) para cada carga simulada pelo RBE no perfil Browsing do experimento 2

O gráfico indica um melhor desempenho geral para 4000 clientes. Entretanto, no final do experimento há uma grande queda, que indica problemas devido a alta carga. Nesse caso, 3000 clientes apresenta melhor estabilidade em relação a 4000 e maior quantidade de WIPS em relação a 2000 clientes

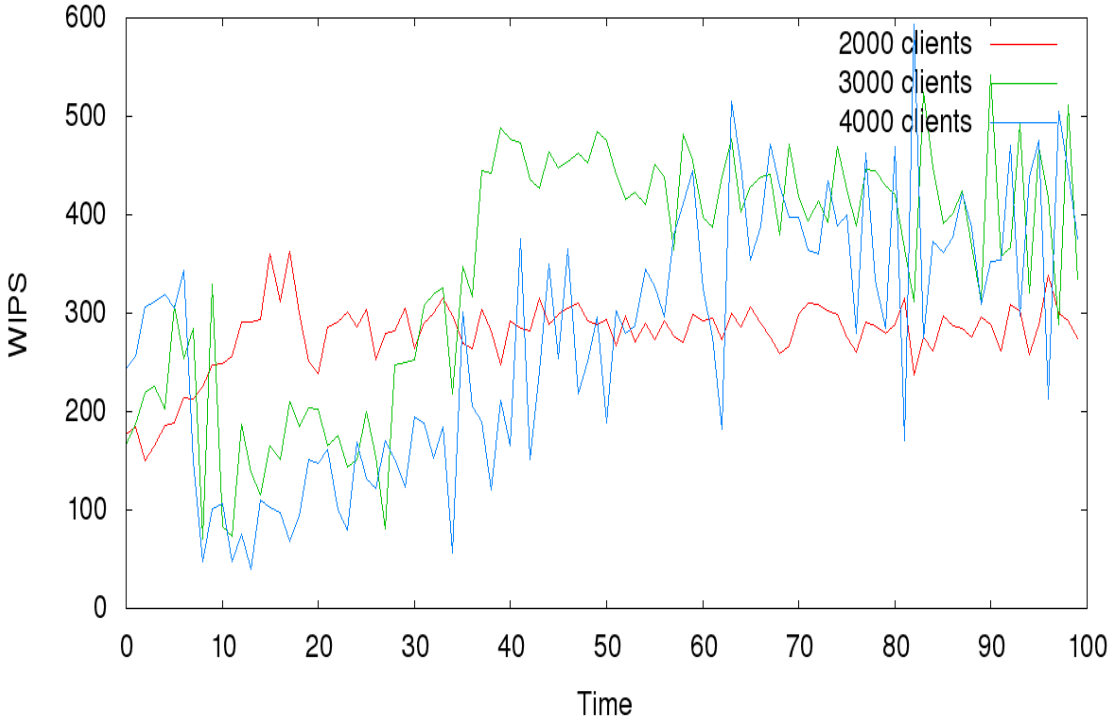

Figura 5: WIPS por tempo(s) para cada carga simulada pelo RBE no perfil Ordering do experimento 2

O melhor desempenho geral foi para 3000 clientes, pois essa carga manteve uma boa média durante todo o experimento e se manteve no mesmo nível de WIPS para 4000 clientes na segunda metade do experimento.

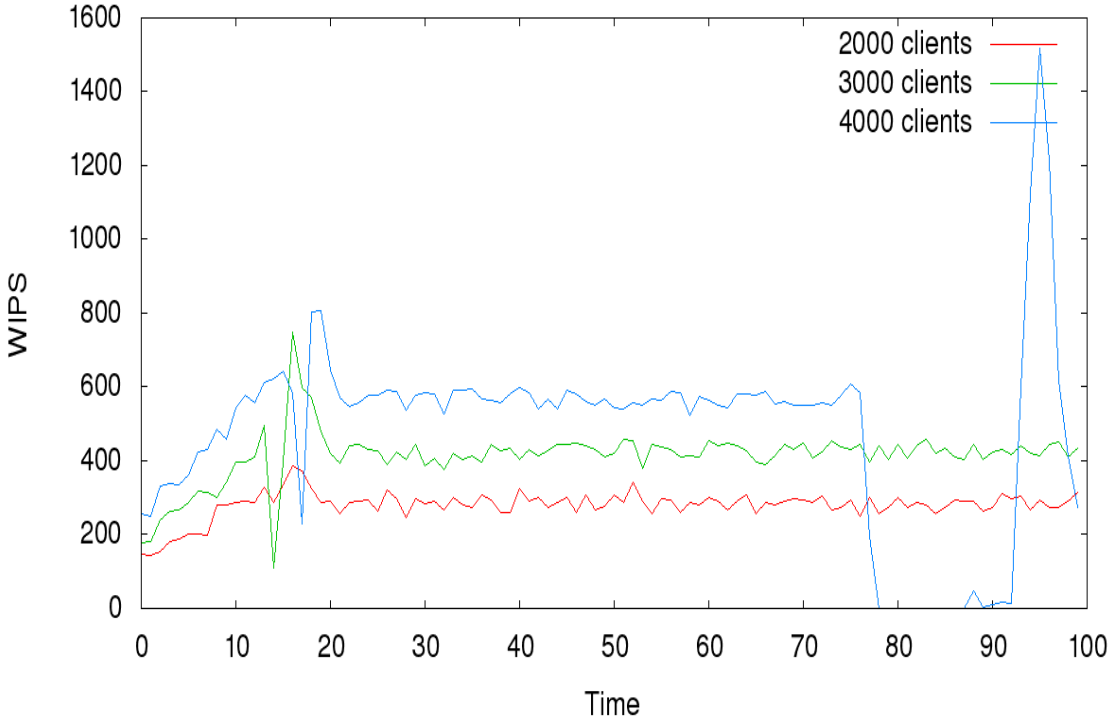

Figura 6: WIPS por tempo(s) para cada carga simulada pelo RBE no perfil Shopping do experimento 2

Este resultado se assemelha ao do perfil Browsing. 4000 clientes possui bom desempenho até aproximadamente 70s do experimento, quando sofre queda de WIPS, o que indica que esta alta carga sobrecarrega o servidor. Como no perfil Browsing, a carga mais estável é de 3000 clientes.

Definimos então rodar o experimento três (com failover) utilizando uma carga de 3000 clientes.

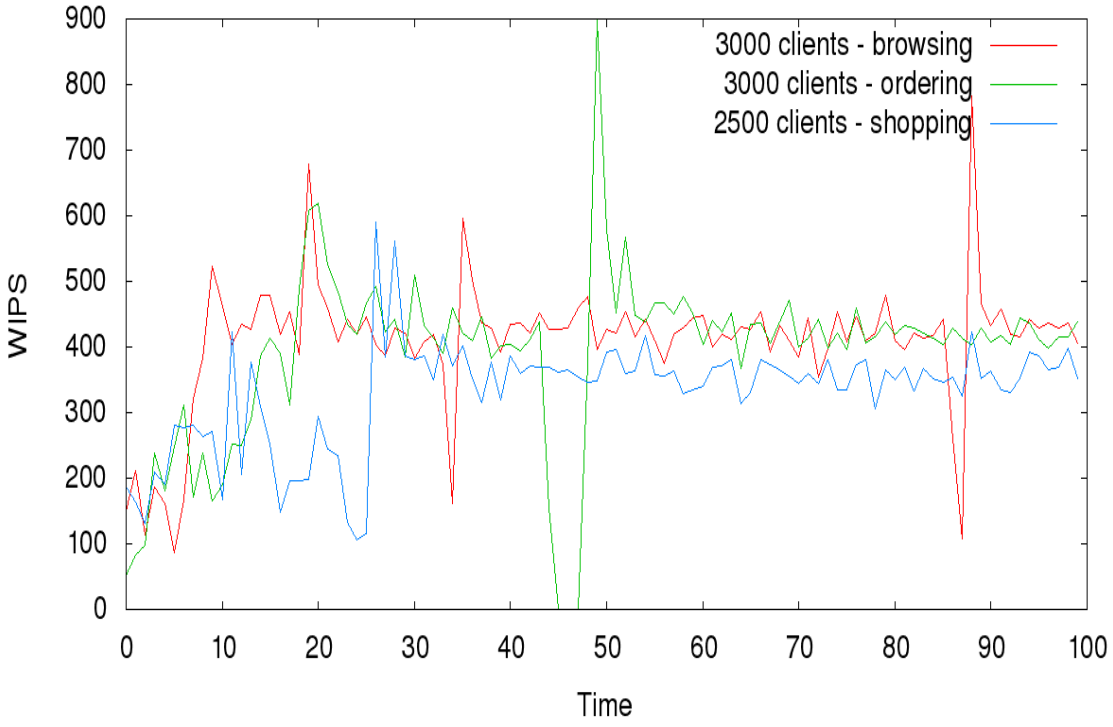

Figura 7: WIPS por tempo(s) para cada perfil simulado pelo RBE no experimento 3

Este gráfico indicas alguns resultados interessantes. Podemos ver que não houve queda de desempenho nos perfis de browsing e shopping, mesmo após o failover. Isso ocorreu provavelmente pois nosso script finaliza o processo do banco de dados primário (causando o failover) e logo em seguida já promove o secundário que rapidamente assume como primário, pois o HAProxy é rápido em notar que o primário falhou e já redireciona as requisições para o secundário. Mas notamos que no perfil Ordering, onde tem muito mais instruções de escrita, que são muito custosas, a carga de 3000 causou falhas no servidor. Por, utilizamos uma carga de 2500 e conseguimos resultados satisfatórios. Vemos que o failover ocorre perto da marca de 45s e que o servidor se recupera por volta de 49s, o que da um pouco menos de quatro segundos de recuperação.

5 Conclusão

Concluímos que os experimentos foram bem sucedidos.

Os experimentos um e dois, conseguiram prever com certa precisão a melhor carga a ser utilizada no experimento três, de 3000 clientes.

O experimento três teve bastante sucesso nos perfis de browsing e shopping, onde o failover não fez cair o número de WIPS do servidor, pois a troca de banco de dados foi quase que imediata. Já no perfil de ordering, houve por volta de quatro segundos de tempo de recuperação para o servidor voltar a responder requisições, o que consideramos um tempo bastante razoável e que podemos tentar melhorar no futuro.

Infelizmente, tivemos problemas no experimento três e não tivemos tempo de rodar o experimento várias vezes para compararmos os resultados, mas ficamos bastante satisfeitos com o resultado demonstrado.

Tabém falhamos em conseguir apresentar os dados de WIRT (WEB Interaction Response Time) o que seria um dado muito bom de ser demonstrado, mas nossa coleta de dados falhou em consegui-los, obtínhamos sempre os mesmos pontos. Pretendemos consertar isso para o próximo estágio do projeto.