Balanceamento de Carga Orientado a Fluxos de Dados: Otimizando a Utilização de Servidores Web

Daniel Stefani Marcon, Leonardo Richter Bays
Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
{dsmarcon,lrbays}@inf.ufrgs.br

***Resumo*—A expansão da *Internet* provocou um aumento do número de usuários requisitando serviços através da rede, bem como do número de servidores e da quantidade de serviços oferecidos. Com o objetivo de minimizar esse problema, servidores *web* começaram a ser implementados com uma arquitetura distribuída, porém com um única interface externa para o recebimento de requisições de usuários. Nesse artigo, é proposta uma abordagem para o balanceamento de carga orientada a fluxos de dados, com a utilização da tecnologia OpenFlow. Nesse contexto, são avaliadas três técnicas: escolha aleatória, escolha baseada em fatias de tempo e balanceamento ponderado.**

I. Introdução

A expansão da *Internet* provocou um aumento do número de usuários requisitando serviços através da rede, bem como do número de servidores e da quantidade de serviços oferecidos. Além disso, os usuários esperam respostas em um curto intervalo de tempo, sem levar em consideração a quantidade de requisições que os servidores *web* precisam atender em um determinado instante. Esse novo cenário acarreta problemas ao modelo cliente/servidor utilizado na *Internet*, já que o nível de escalabilidade de um único servidor *web* é relativamente baixo [1], [2].

Com o objetivo de minimizar esse problema, servidores *web* começaram a ser implementados com uma arquitetura distribuída, porém com um única interface externa para o recebimento de requisições de usuários. Desse modo, são necessários mecanismos de balanceamento de carga para propagar as requisições entre os diversos nodos oferecendo o serviço requisitado [3]. Assim, a escalabilidade do serviço sendo ofertado é proporcional à eficiência do algoritmo de balanceamento de carga usado para prover uma utilização ótima dos recursos disponíveis.

Recentes avanços em tecnologias de gerência de rede permitem a reprogramação dinâmica de dispositivos orientada a fluxos de dados. O protocolo OpenFlow[1] permite tal reprogramação em *switches* de rede através de um controlador externo, que concentra as operações de gerência.

Nesse artigo, é proposta uma abordagem para o balanceamento de carga através de fluxos de dados, com a utilização da tecnologia OpenFlow. Dessa forma, cada fluxo de dados é direcionado a um servidor, de acordo com a política sendo utilizada. Para avaliar a eficácia dessa abordagem no contexto de balanceamento de carga, são utilizadas três políticas distintas: escolha aleatória, escolha baseada em fatias de tempo e balanceamento ponderado.

[1]http://www.openflow.org/

O restante desse artigo está organizado da seguinte forma: a Seção II descreve as melhores técnicas de balanceamento de carga encontradas na literatura, bem como a tecnologia OpenFlow. A Seção III apresenta a solução proposta para o balanceamento de carga e os experimentos realizados para avaliar as técnicas de balanceamento de carga utilizadas na proposta. A Seção IV apresenta os resultados obtidos a partir dos experimentos realizados. Por fim, a Seção V apresenta as conclusões obtidas sobre o trabalho realizado.

II. Trabalhos Relacionados

Nesta Seção, são apresentadas as melhores propostas de balanceamento de carga encontradas na literatura, além de ser introduzida a tecnologia OpenFlow, utilizada pela solução apresentada nesse artigo para realizar o controle de fluxos de dados.

A. Balanceamento de Carga

O balanceamento de carga é um tópico com diversos estudos presentes na literatura. Em [2], foi proposta a união da técnica de *round robin* com o esquema de TTL (*time-to-live*) adaptativo, os quais levam em consideração a distribuição desigual das taxas de requisições de clientes e a heterogeneidade dos servidores *web* para cada mapeamento de requisição de endereço.

Já em [4], é proposto um algoritmo de balanceamento de carga em tempo real baseado nas estatísticas do servidor (*Real Time Server-statistics based Load Balancing* - RTSLB). Essa técnica leva em conta três fatores principais: a utilização da CPU do servidor *web*, a sua taxa de resposta e o número de requisições sendo atendidas pelo servidor. Em [5], o tráfego *web* é modelado através de processos estocásticos e alguns algoritmos de balanceamento de carga baseados em DNS (*Domain Name Server*) são comparados através de simulações.

Outra solução proposta baseia-se na técnica de *round robin* com DNS, com a adição de duas novas funcionalidades. Primeiramente, é descrito um novo algoritmo para a indexação de desempenho e de carga dos nodos. Em segundo lugar, é proposto um escalonador para a distribuição de tarefas no servidor DNS, que alocará um servidor baseando-se nos índices de desempenho e de carga. Para validar a proposta, os autores realizaram uma simulação, comparando essa abordagem com a técnica convencional de *round robin*.

Em [6], é proposta uma técnica, denominada *Random Early Detection* (RED), para detectar servidores *web* que ficarão sobrecarregados em um futuro próximo. De

acordo com o RED, a probabilidade de um servidor ficar sobrecarregado está relacionada à sua carga atual. Essa técnica tem o objetivo de mitigar o problema que técnicas baseadas em DNS apresentam de oscilação de carga entre os nodos oferecendo o serviço.

B. OpenFlow

A tecnologia OpenFlow possibilita a execução de testes de protocolos experimentais em redes reais, concomitantemente com o tráfego de produção. OpenFlow é uma tecnologia para abstração e virtualização de redes, possibilitando o controle de tráfego da rede através de fluxos de dados [7].

O OpenFlow permite o gerenciamento de *switches* programáveis, permitindo separar diferentes tipos de tráfegos (por exemplo, tráfego de produção de tráfego experimental), através de um controlador externo. Isso é possível devidos a *switches* e roteadores possuírem tabelas de fluxo (*flow tables*) utilizadas para implementar NAT (*network address translation*), QoS (*quality of service*) e outras funcionalidades. Apesar dessas tabelas variarem entre os diferentes fabricantes, há um protocolo padrão que provê um conjunto de funções idênticas em todos os dispositivos.

A utilização de uma plataforma baseada em fluxos de dados como o OpenFlow traz vantagens no contexto de balanceamento de carga. Ao contrário das soluções apresentadas acima, não é necessária nenhuma ação por parte dos servidores. O controlador externo possui conhecimento sobre a rede, incluindo a quantidade de fluxos ativos por servidor, o que permite que o balanceamento de carga seja realizado por este de forma autônoma.

III. Solução Proposta

Esta Seção apresenta a solução proposta para o balanceamento de carga, através da utilização da tecnologia OpenFlow. Por fim, o ambiente de testes e os experimentos realizados são descritos.

A. Balanceamento por Fluxos de Dados

Para realizar o balanceamento de carga entre diversos servidores dentro de uma rede, é proposta uma abordagem orientada a fluxos de dados. Um fluxo de dados é estabelecido no momento em que um cliente envia uma requisição para um servidor. O fluxo permanece ativo enquanto houver comunicação entre as duas partes, expirando após um determinado tempo de inatividade.

Os diferentes fluxos de dados são divididos entre os servidores de forma transparente para o cliente. Os fluxos são gerenciados pelos *switches* da rede, que redirecionam os pacotes de forma com que sejam entregues a um servidor escolhido por uma política de balanceamento de carga. Nesse contexto, são propostas três políticas para distribuir os fluxos entre os servidores da rede, com diferentes níveis de complexidade:

- **Escolha aleatória:** a cada fluxo criado, o controlador seleciona um servidor de forma aleatória. Essa é a política mais simples, onde não é necessário armazenar nenhum dado, porém a carga atual de cada servidor não é considerada.
- **Escolha baseada em fatias de tempo:** a cada intervalo de tempo, o controlador seleciona um servidor responsável por atender as requisições durante o período. A seleção do servidor a cada intervalo de tempo é feita de forma sequencial (*round robin*). Dessa forma, nenhum servidor receberá mais fatias de tempo para atender em relação aos demais servidores. Nessa política, o único dado armazenado é o servidor responsável naquele período, porém a carga atual de cada servidor também não é considerada.
- **Balanceamento ponderado:** o controlador possui um controle geral de quantos fluxos estão sendo redirecionados para cada um dos servidores. Dessa forma, para realizar a escolha de um servidor na criação de um novo fluxo, é escolhido sempre o servidor com menor nível de carga, isto é, o servidor que atualmente está se comunicando com o menor número de clientes. A cada fluxo criado para um servidor, o contador de fluxos desse servidor é incrementado, e o mesmo é decrementado a cada fluxo removido.

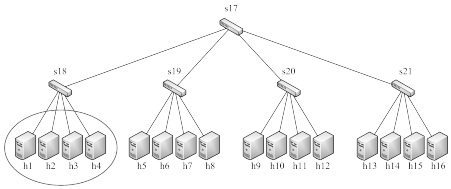

Figura 1. Topologia de rede utilizada, com os servidores em destaque.

B. Experimentos

Para realizar os experimentos, primeiramente foi criada uma rede virtual com *switches* que suportam o protocolo OpenFlow através da aplicação Mininet[2]. A rede criada possui uma topologia em forma de árvore, conforme apresentado na Figura 1. A topologia conta com um total de 16 *hosts* e cinco *switches*. Desses 16 *hosts* presentes na rede, quatro foram utilizados como servidores (hosts conetados ao *switch* **s18**, em destaque na figura), e os demais operam como clientes. Todos os clientes acessam os servidores através do endereço IP do servidor principal (**h1**), porém, os pacotes são modificados antes de chegarem ao destino, para que seja executado o balanceamento de carga entre os quatro servidores da rede (**h1**, **h2**, **h3** e **h4**).

Para realizar o balanceamento de carga entre os servidores presentes na rede, foi criado um *script* na linguagem Python utilizando a API do controlador NOX [3]. A cada pacote recebido por um *switch*, o mesmo verifica se já há um fluxo definido para o pacote. Caso contrário,

[2]http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet
[3]http://noxrepo.org/wp/

esse avisa o controlador, que decidirá qual ação tomar. O controlador provê o comportamento padrão para todos os *switches* da rede. Isto é, para todos os *switches*, há o comportamento de aprendizado de endereços MAC e suas respectivas portas no *switch*.

Além disso, o controlador provê ao switch **s17**, que interconecta os demais *switches* presentes na rede, a capacidade de realizar o balanceamento de carga entre os quatro servidores presentes. O balanceamento de carga é realizado através da criação dinâmica de fluxos nesse *switch*, que encontra-se no nível superior da rede. Ao receber um pacote endereçado para o servidor principal (**h1**), é criado um fluxo para redirecionar os pacotes para um dos quatro servidores, de acordo com a política de balanceamento definida. Dessa forma, os pacotes subsequentes recebidos a partir da mesma origem nessa conexão serão redirecionados para esse mesmo servidor.

O cenário descrito foi instanciado para as três políticas de balanceamento de carga por fluxos de dados propostas na Seção III-A: escolha aleatória, escolha *round robin*, com uma fatia de tempo de cinco segundos, e balanceamento ponderado.

Os fluxos criados possuem, também, um tempo definido de *timeout*. Ou seja, após um certo tempo de inatividade, o fluxo é removido. Isso garante que fluxos inativos não serão utilizados por outra conexão após o estouro do *timeout*. Dessa forma, caso ocorram mudanças na atividade entre clientes e servidores ao longo do tempo, é possível adaptar o balanceamento de carga.

Para simular o comportamento dos clientes, foi criado um *script*, que é executado continuamente em cada um destes. O *script* é executado durante 30 minutos (1800 segundos). Nesse período, o cliente envia pacotes do tipo *ICMP echo request* continuamente para o servidor, por um período com duração entre 60 e 300 segundos, permanece inativo inativo entre 15 e 45 segundos e, após, reinicia o ciclo.

IV. Resultados

Conforme descrito na seção anterior, os testes foram realizados em uma rede virtual criada através da ferramenta Mininet. Os dados foram coletados através de registros gerados pelo próprio controlador implementado. Os registros são atualizados a cada evento de criação ou remoção de fluxos, e mostram a quantidade exata de fluxos ativos na rede durante o período de testes.

Nos testes realizados utilizando a política aleatória de balanceamento, pode-se notar diferenças nos níveis de carga ao longo da execução. A série temporal exibida na Figura 2 mostra que houve uma distribuição de carga entre os servidores. Porém, é possível visualizar períodos em que certos servidores tiveram cargas superior ou inferior aos demais. O servidor **h4**, por exemplo, foi o servidor com maior nível de carga no período entre aproximadamente 6 e 11,5 minutos da execução. Isso significa um período de 5 minutos e meio consecutivos, ou cerca de 18% do tempo de execução do experimento.

A política de escolha *round robin* mostra um comportamento similar durante a maior parte da execução.

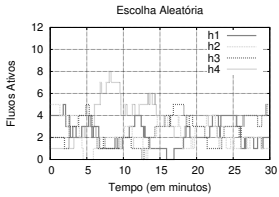

Figura 2. Nível de carga dos servidores ao longo do tempo utilizando a política de escolha aleatória.

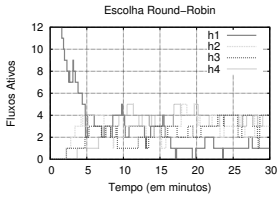

Figura 3. Nível de carga dos servidores ao longo do tempo utilizando a política de escolha *round robin*.

Porém, nos resultados dos experimentos realizados, exibidos na Figura 3, pode-se observar que houve um grande pico de conexões a um único servidor no período inicial dos testes. Isso se deve ao fato de que todos os clientes começam a enviar pacotes exatamente ao mesmo tempo, e nesta política de escolha, um único servidor é escolhido durante uma determinada fatia de tempo. No restante da execução, houve um balanceamento similar ao da política aleatória. Porém, os resultados obtidos mostram que essa política é inadequada em situações de *flash crowd*, onde há um grande aumento de usuários na rede em um curto espaço de tempo.

Em contraste com os resultados das duas primeiras políticas, a escolha ponderada mostrou-se capaz de prover um balanceamento muito próximo do ideal durante toda a execução. Como pode ser observado na Figura 4, o nível de carga permanece equilibrado entre todos os servidores, evitando que hajam servidores com cargas demasiadamente altas ou baixas.

Observando os gráficos é possível notar diferenças

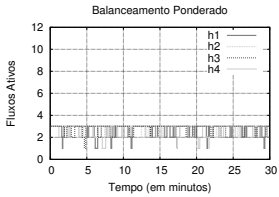

Figura 4. Nível de carga dos servidores ao longo do tempo utilizando a política de balanceamento ponderado.

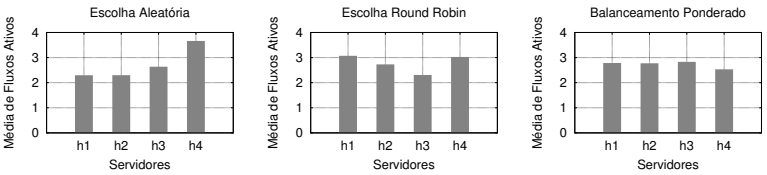

Figura 5. Média de carga dos servidores durante os experimentos.

significativas nos picos de carga atingidos pela utilização de diferentes políticas. Na política de escolha aleatória, os picos mantiveram-se entre 5 e 8 fluxos ativos por servidor. Já na política de escolha *round robin*, os níveis máximos observados encontram-se entre 4 e 12 fluxos ativos. O *flash crowd* presente no início do experimento levou à sobrecarga de um dos servidores, que atendeu às requisições de todos os clientes da rede simultaneamente.

No entanto, na política de balanceamento ponderado, nenhum dos servidores teve um número de fluxos ativos superior a 3. Isso mostra que a política obteve um sucesso muito maior ao balancear a carga da rede, pois nenhum dos 4 servidores, em momento algum, atendeu a uma quantidade de clientes maior que ¼ dos presentes na rede.

Comparando a média de carga dos servidores durante o tempo de execução dos experimentos, é possível visualizar que não houve uma grande diferença entre as três políticas. Utilizando a política de escolha aleatória, a menor média de fluxos ativos obtida foi de ≈2,3, e a maior média, de ≈3,6. Na política *round robin*, as médias mais baixa e mais alta foram de, respectivamente, ≈2,3 e ≈3,1. Já no balanceamento ponderado, os valores ficaram entre ≈2,5 e ≈2,8.

A comparação pode ser visualizada na Figura 5, e mostra que o balanceamento ponderado obteve um maior equilíbrio entre as médias de carga dos servidores. No entanto, julga-se que tal diferença foi demasiadamente baixa para ser levada em consideração. Também é possível observar que, apesar do pico inicial de carga no servidor **h1** na política *round robin*, as médias de carga mantiveram-se balanceadas entre os quatro servidores.

V. Considerações Finais

Balanceamento de carga é um tópico de grande importância para garantir a utilização eficiente dos recursos disponíveis, não se limitando a servidores *web*. Entretanto, além de garantir que a política otimize a utilização dos recursos disponíveis, também deve-se levar em consideração a complexidade do código que implementa a política.

Nesse sentido, o OpenFlow possibilita a criação de novas técnicas de balanceamento de carga, com maior flexibilidade e controle. Por isso, foram propostas três políticas de balanceamento de carga orientadas à fluxos de dados. Nas medições realizadas, a política do balanceamento ponderado obteve os melhores resultados em detrimento das outras políticas. Entretanto, a complexidade do código dessa política é linear em relação ao número de servidores ($O(n)$), o que pode torná-la ineficiente caso haja um grande número de nodos atendendo às requisições de um mesmo serviço.

Como trabalhos futuros, pretende-se realizar medições das propostas de balanceamento de carga em redes de produção e compará-las com as melhores propostas encontradas na literatura. Além disso, pretende-se combinar esquemas de balanceamento de carga com técnicas de *failover* para aumentar a disponibilidade de sistemas *web*.

Referências

[1] T. Kwan, R. McGrath, and D. Reed, "NCSA's World Wide Web server: design and performance," *Computer*, vol. 28, no. 11, pp. 68 –74, nov 1995.

[2] M. Colajanni and P. S. Yu, "Adaptive TTL schemes for load balancing of distributed web servers," *SIGMETRICS Perform. Eval. Rev.*, vol. 25, pp. 36–42, September 1997. [Online]. Available: http://doi.acm.org/10.1145/262391.262401

[3] J. Aweya, M. Ouellette, D. Y. Montuno, B. Doray, and K. Felske, "An adaptive load balancing scheme for web servers," *Int. J. Netw. Manag.*, vol. 12, pp. 3–39, January 2002. [Online]. Available: http://dx.doi.org/10.1002/nem.421

[4] D. Shadrach, K. Balagani, and V. Phoha, "A weighted metric based adaptive algorithm for web server load balancing," in *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*, vol. 1, nov. 2009, pp. 449 –452.

[5] Z. Xu, R. Huang, and L. Bhuyan, "Load balancing of dns-based distributed web server systems with page caching," in *Parallel and Distributed Systems, 2004. ICPADS 2004. Proceedings. Tenth International Conference on*, july 2004, pp. 587 – 594.

[6] C.-C. Yang, C. Chen, and J.-Y. Chen, "Random early detection web servers for dynamic load balancing," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, dec. 2009, pp. 364 –368.

[7] Y. Kanaumi, S. Saito, and E. Kawai, "Toward large-scale programmable networks: Lessons learned through the operation and management of a wide-area openflow-based network," in *Network and Service Management (CNSM), 2010 International Conference on*, oct. 2010, pp. 330 –333.