Solução eficiente para balanceamento de carga em servidores de aplicação java

Efficient load balancing solution for java application servers

DOI:10.34115/basrv5n4-014

Recebimento dos originais: 06/07/2021
Aceitação para publicação: 06/08/2021

José Renato Castro Milanez
Mestre em Engenharia Elétrica pela Universidade Federal de Itajubá
Universidade Federal de Itajubá
Endereço: Av. BPS, 1303, Bairro Pinheirinho, Itajubá – MG
E-mail: jrcmilanez@unifei.edu.br

Pablo Marques de Oliveira
Mestre em Ciência e Tecnologia da Computação pela Universidade Federal de Itajubá
Universidade Federal de Itajubá – UNIFEI
Endereço: Av. BPS, 1303, Bairro Pinheirinho, Itajubá – MG
E-mail: pablo@unifei.edu.br

Edmilson Marmo Moreira
Doutor em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo
Universidade Federal de Itajubá – UNIFEI
Endereço: Av. BPS, 1303, Bairro Pinheirinho, Itajubá – MG
E-mail: edmarmo@unifei.edu.br

Thatyana de Faria Piola Seraphim
Doutora em Física Aplicada pela Universidade de São Paulo
Universidade Federal de Itajubá – UNIFEI
Endereço: Av. BPS, 1303, Bairro Pinheirinho, Itajubá – MG
E-mail: thatyana@unifei.edu.br

Enzo Seraphim
Doutor Ciências da Computação e Matemática Computacional pela Universidade de São Paulo
Universidade Federal de Itajubá – UNIFEI
Endereço: Av. BPS, 1303, Bairro Pinheirinho, Itajubá – MG
E-mail: seraphim@unifei.edu.br

RESUMO
Atualmente, o desenvolvimento de aplicações para internet é a melhor forma de oferecer facilidade de acesso aos usuários a partir de um computador ou de um dispositivo móvel, pois não é necessário instalar nenhum software adicional. Dentre as diversas soluções para o desenvolvimento de aplicações, a plataforma Java tem se tornado a procurada por ser independente de ambiente computacional, oferecendo uma variedade de bibliotecas, e projetada para executar código com segurança. Assim, aplicações baseadas em Java passaram a automatizar atividades operacionais das áreas administrativa e acadêmica das

universidades federais. Em certas ocasiões, estas aplicações podem sofrer uma grande demanda de requisições simultâneas provocando baixos tempos de resposta ou até a interrupção dos serviços. Uma solução para atender a grande demanda de requisições simultâneas é utilizar várias máquinas na rede atuando como um único servidor, através de um balanceamento de carga. Este trabalho apresenta um estudo comparativo entre os balanceadores de carga Apache Server com Tomcat Connector mod_jk e o HAProxy dos servidores de aplicação Java que hospedam os Sistemas Institucionais Integrados de Gestão (SIG/UFRN) da Universidade Federal de Itajubá.

Palavras-Chave: Balanceamento de Carga, Alta Disponibilidade, Benchmark.

ABSTRACT
Currently, the development of internet for applications is the best way to provide easy access to users from a computer or a mobile device, it is not necessary to install any additional software. Among the various solutions for application development, the Java platform has become the sought to be independent computing environment, offering a variety of libraries and designed to run with security code. Thus, Java-based applications started to automate operational activities of the administrative and academic areas of federal universities. Sometimes, these applications can suffer a great demand for simultaneous requests causing low response times or even service interruption. A solution to meet the high demand of simultaneous requests is to use multiple machines on the network acting as a single server through a load balancing. This work presents a comparative study of the Apache Server load balancers with Tomcat Connector mod_jk and HAProxy Java application servers that host the Integrated Institutional Management Systems (GIS / UFRN) of the Federal University of Itajubá.

Keywords: Load Balance, High Availability, Benchmark.

1 INTRODUÇÃO

O constante crescimento da Internet provoca muitos problemas de desempenho como, tempos de resposta baixos, congestionamento da rede e interrupção dos serviços, causados por uma normal sobrecarga do sistema ou por ataque distribuído de negação de serviço (Distributed Denial of Service – DDoS). Assim, uma única máquina servidora não é suficiente para lidar com o grande volume de tráfego.

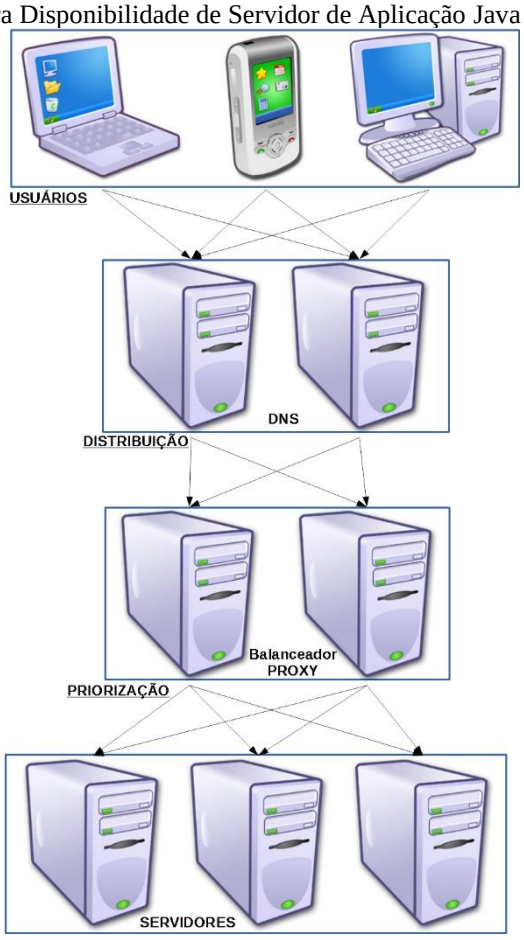
A solução mais utilizada para minimizar ou resolver esses problemas é utilizar várias máquinas na rede atuando como um único servidor, através de um balanceamento de carga. O balanceamento de carga divide a quantidade de trabalho de um computador entre dois ou mais computadores para que mais trabalho seja feito na mesma quantidade de tempo. Essa divisão de trabalho também pode ser descrita como um processo de distribuição de solicitações de serviços por um grupo de servidores.

Os serviços ou aplicações podem ser instalados nos vários servidores que estão configurados para compartilhar a carga de trabalho. Esta configuração é conhecida como

cluster com balanceamento de carga, onde, a carga é dividida de modo a equilibrar o desempenho dos programas baseados em servidores, como um Servidor Web. Esse balanceamento é realizado através da distribuição de pedidos de clientes para vários servidores.

A solução para um efetivo balanceamento de carga requer que as requisições sejam distribuídas e priorizadas entre os vários servidores, através de hardware ou de software. Existem soluções específicas que utilizam hardwares que não são tratados neste trabalho [1]. A figura 1 apresenta a arquitetura baseada em software para a solução que oferece disponibilidade para servidor de aplicação Java baseada em distribuição por DNS (Domain Name System) e priorização por balanceador Proxy.

Figura 1. Arquitetura para Disponibilidade de Servidor de Aplicação Java com Balanceador Proxy.



Para garantir disponibilidade do serviço, a distribuição de requisições aproveita-se da natureza distribuída do sistema de nomes de domínio [2] (Domain Name System – DNS) com comportamento do algoritmo Round-Robin para distribuir respostas de um único registro de endereço entre diversos servidores [3]. O algoritmo de Round-Robin é um método de atendimento usado na teoria das filas da área de probabilidade [4]. Por

exemplo, um domínio que tem três sites idênticos hospedados em três servidores com três endereços IP diferentes. Ao acessar o site através do domínio, o DNS Round-Robin responde ao usuário o primeiro endereço IP. O segundo usuário que acessa o site será enviado para o segundo endereço IP, e o terceiro usuário será enviado para o terceiro endereço IP. Em cada caso, uma vez que o endereço de IP é entregue, ele vai até ao final da lista de IPs cadastrados para o domínio. Assim, o próximo usuário vai ser enviado para o primeiro endereço de IP, e assim por diante.

Essa técnica chamada de DNS Round-Robin é uma forma de balanceamento estático, pois não considera a questão de prioridade de resposta de cada servidor já que o algoritmo Round-Robin distribui as requisições levando-se em consideração somente qual é o próximo recurso de determinada sequência em que a requisição deve ser encaminhada. Existem soluções específicas que possuem versões modificadas do algoritmo de Round-Robin, com o objetivo de prover distribuição do uso de recursos com prioridade que não são tratados neste trabalho [5].

A priorização de requisições pode ser estabelecida com o uso de um balanceador Proxy como:

- **mod_proxy** – módulo do Apache que implementa Proxy/gateway para vários protocolos populares bem como apresentam diferentes algoritmos de balanceamento de carga [6].
- **mod_jk** – módulo do Apache que gerencia a comunicação entre o Tomcat e o Apache [7].
- **HAProxy** – software de alta disponibilidade, balanceamento de carga e Proxy para protocolo TCP e aplicações baseadas em HTTP [8].

Os balanceadores Proxy citados também fazem distribuição, no entanto, não garantem a disponibilidade do serviço.

Este trabalho tem por objetivo a comparação entre Apache Server com Tomcat Connector mod_jk e o HAProxy para balanceamento de carga em servidores de aplicação Java. Ambas as soluções usam software livre e devem atender as seguintes características:

- **Aumento da escalabilidade.** Quando as requisições não são mais atendidas por um único servidor, é necessária flexibilidade para adicionar mais servidores de forma rápida e transparente aos usuários.
- **Alto desempenho.** O melhor desempenho é alcançado quando o poder de processamento dos servidores é usado inteligentemente. Uma infraestrutura avançada de

balanceamento de carga pode direcionar as requisições do usuário para os servidores menos ociosos e, capazes de fornecer o tempo de resposta mais baixo.

• **Alta disponibilidade e recuperação de desastres.** O terceiro benefício do balanceamento de carga é a capacidade de melhorar a disponibilidade de serviço. Se um serviço ou servidor falhar, o balanceamento de carga pode automaticamente redistribuir as requisições do usuário para outros servidores em um cluster de servidores ou para servidores em outro local.

2 METODOLOGIA

A Universidade Federal de Itajubá (UNIFEI) adotou o Sistema Integrado de Gestão (SIG) desenvolvido pela Universidade Federal do Rio Grande do Norte (UFRN) que é baseado em servidores de aplicação Java. O SIG foi desenvolvido com o intuito de informatizar as atividades operacionais das áreas administrativas, acadêmica e de recursos humanos. Dessa maneira, é fundamental que o sistema possua alta disponibilidade e esteja preparado para receber grandes quantidades de requisições simultâneas.

A alta disponibilidade é um fator extremamente desejado, uma vez que a indisponibilidade do SIG pode causar uma série de transtornos e prejuízos para a universidade.

Já a preocupação com as grandes quantidades de requisições simultâneas está ligada diretamente ao módulo de Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA), que ocorre durante o período de matrículas. Neste período, pode-se ter o pior cenário onde todos os alunos da instituição acessam o sistema simultaneamente, gerando lentidão e/ou até mesmo, indisponibilidade.

Para atender as grandes quantidades de requisições simultâneas, pode-se utilizar o escalonamento vertical ou horizontal.

No escalonamento vertical, o único servidor deve ser substituído ou melhorado seu hardware (adicionando mais memória RAM e processadores). Porém, quando a capacidade de hardware é atingida e a quantidade máxima de requisições simultâneas desejadas não foi atendida, é necessário adotar o escalonamento horizontal, que é a distribuição das requisições simultâneas em vários computadores servidores.

No escalonamento horizontal precisamos garantir que as requisições sejam distribuídas uniformemente entre os computadores servidores, de modo a impedir o congestionamento em um único equipamento e evitar uma possível indisponibilidade.

Para garantir a alta disponibilidade e as grandes quantidades de requisições simultâneas, é necessário que o SIG seja disponibilizado aos usuários através do escalonamento horizontal. A seguir serão descritos como foi realizado o escalonamento horizontal usando Apache Server com Tomcat Connector mod_jk e usando HAProxy.

2.1 APACHE SERVER COM TOMCAT CONNECTOR MOD_JK

A infraestrutura computacional proposta e usada atualmente pela Universidade Federal do

Rio Grande do Norte (UFRN), desenvolvedora do sistema, é baseada nas aplicações Apache Server [6] e Apache Tomcat Connector mod_jk [7]. Esta solução atua como balanceador de carga dos servidores de aplicação do SIG, distribuindo as requisições ao Apache Server [6] entre os servidores de aplicação do SIG.

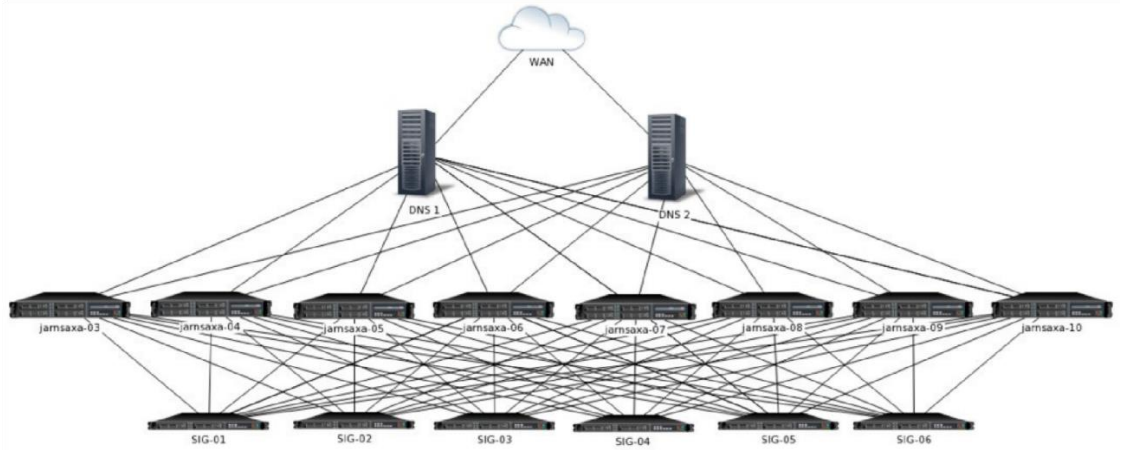
Porém, esta solução não apresenta alta disponibilidade, já que o sistema SIG não foi projetado para tirar proveito do benefício de clusterização do JBoss, que é o seu servidor de aplicação. Dessa forma, ao acessar o SIG, o usuário é designado a um servidor de aplicação e permanece lá até que sua conexão seja encerrada ou até que o servidor de aplicação do SIG pare de responder por um motivo qualquer. Quando o usuário navegar no sistema SIG e o servidor para o qual foi designado não estiver respondendo, o mesmo será desconectado do sistema e uma nova conexão deverá ser feita. Este é um grande incomodo para os usuários, pois os computadores servidores precisam executar atualizações de softwares do sistema operacional para melhorias, correções de erro e segurança. Em alguns momentos, a reinicialização do servidor é necessária para que as atualizações entrem em vigor.

A infraestrutura aplicada para a interface Web do SIG na UNIFEI é apresentada pela Figura 2, baseada na proposição da UFRN. Ela foi concebida através da ferramenta Load Balancer Configuration Tool [9], baseando-se na quantidade de Apache Servers [6] com mod_jk [7], servidores de aplicação SIG e a quantidade de núcleos de CPU disponíveis para cada um deles. No total, são 16 computadores servidores envolvidos, com a capacidade de atender até 19.200 requisições simultâneas ao SIG.

No primeiro nível de computadores da Figura 2 tem-se dois servidores DNS, responsáveis por retornar a pesquisa ao endereço sig.unifei.edu.br. Cada requisição ao endereço é direcionada para uma das 8 máquinas no segundo nível, fazendo com que o DNS atue como balanceador de carga através do algoritmo Round-Robin. No segundo nível, tem-se 8 servidores Apache Server [6] com o Apache Tomcat Connector mod_jk

[7]. A requisição ao SIG é recebida pelo Apache Server [6] e o balanceamento de carga é realizado pelo mod_jk [7] entre os 6 servidores de aplicação do SIG no terceiro nível, que processam e retornam as requisições aos clientes.

Figura 2 - Infraestrutura para a interface web do SIG

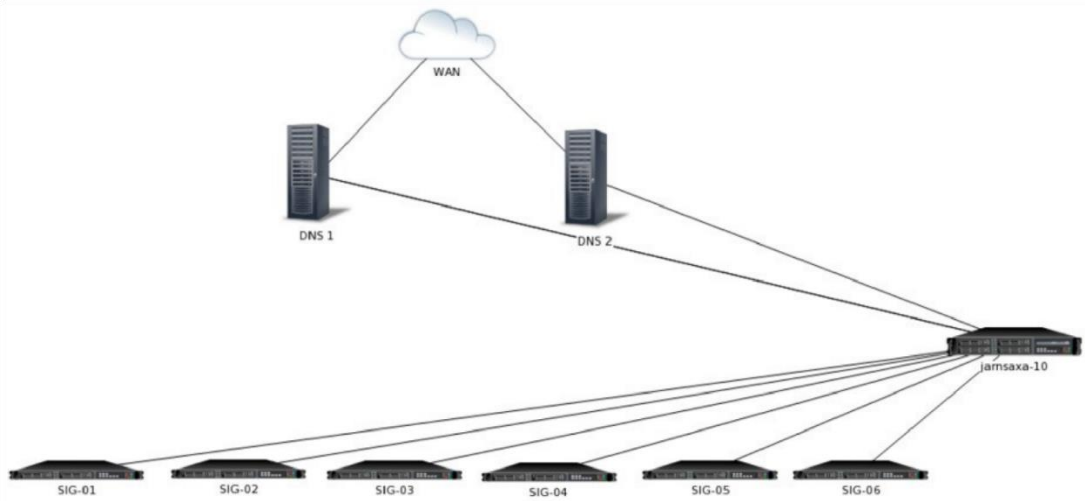


2.3 HAPROXY

Na busca de outras soluções para o balanceamento de carga e alta disponibilidade, foi encontrado o software HAProxy [8], que é amplamente utilizado na Internet para o balanceamento de carga em páginas com grandes quantidades de requisições simultâneas. Por tratar-se de uma implementação completamente diferente da solução proposta pela UFRN, realizou-se um estudo comparativo entre as soluções.

Este trabalho abordará apenas a interface web do SIG. O balanceamento de carga e a alta disponibilidade do banco de dados do SIG será alvo de um próximo trabalho.

Figura 3 - Balanceador de carga sem DNS Round-Robin.



3 RESULTADOS

Para a realização de testes, quatro cenários foram utilizados:

- Apache Server [6] com mod_jk [7] e DNS Round-Robin (Figura 2)
- Apache Server [6] com mod_jk [7] sem DNS Round-Robin (Figura 3)
- HAProxy [8] sem DNS Round-Robin(Figura 3)
- HAProxy [8] com vários núcleos de CPU (nproc) sem DNS Round-Robin (Figura 3)

Para os quatro cenários, foi utiliza a configuração dos servidores apresentada na tabela 1.

Tabela 1 - Configuração dos servidores

Servidor	Tipo	CPU	Memória RAM	Interface de Rede
Gerador de Carga	Virtualizado	6 x vCPU Xeon E7-4807 1.87 GHz	8GB	4 x Gigabit
DNS	Virtualizado	2 x vCPU Xeon E7-4807 1.87 GHz	2GB	1 x Gigabit
Apache Server com mod_jk / HAProxy	Físico	2 x Xeon E5420 2.5 GHz	32GB	4 x Gigabit
Aplicação SIG	Físico	2 x Xeon E5420 2.5 GHz	64GB	2 x Gigabit

No gerador de carga foi instalado o software Apache HTTP server benchmark tool (AB) [10]. Os parâmetros utilizados para o teste de carga foram: 10.000 requisições no total; 200, 400, 600, 800 e 1000 requisições em paralelo.

Com esta metodologia, pretende-se analisar o impacto no desempenho ao utilizar o DNS Round-Robin, bem como analisar o desempenho do HAProxy [8] utilizando somente um e todos os núcleos de CPU disponíveis no servidor. Já o Apache Server [6] não possui esta possibilidade de variação de núcleos de CPU, uma vez que foi desenvolvido para tirar proveito da arquitetura de multiprocessamento.

Os resultados obtidos são apresentados nas figuras a seguir.

Figura 4. 10000 requisições com 200 requisições em paralelo.

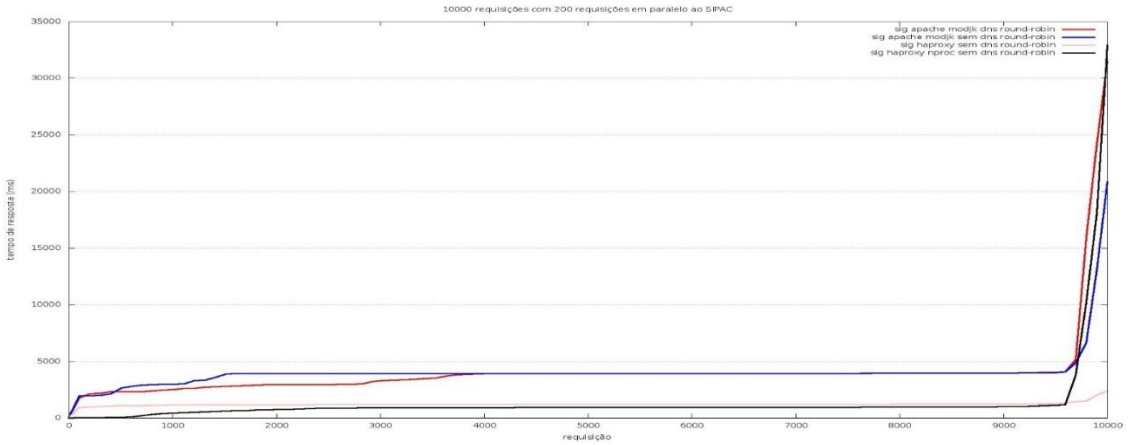


Figura 5. 10000 requisições com 400 requisições em paralelo.

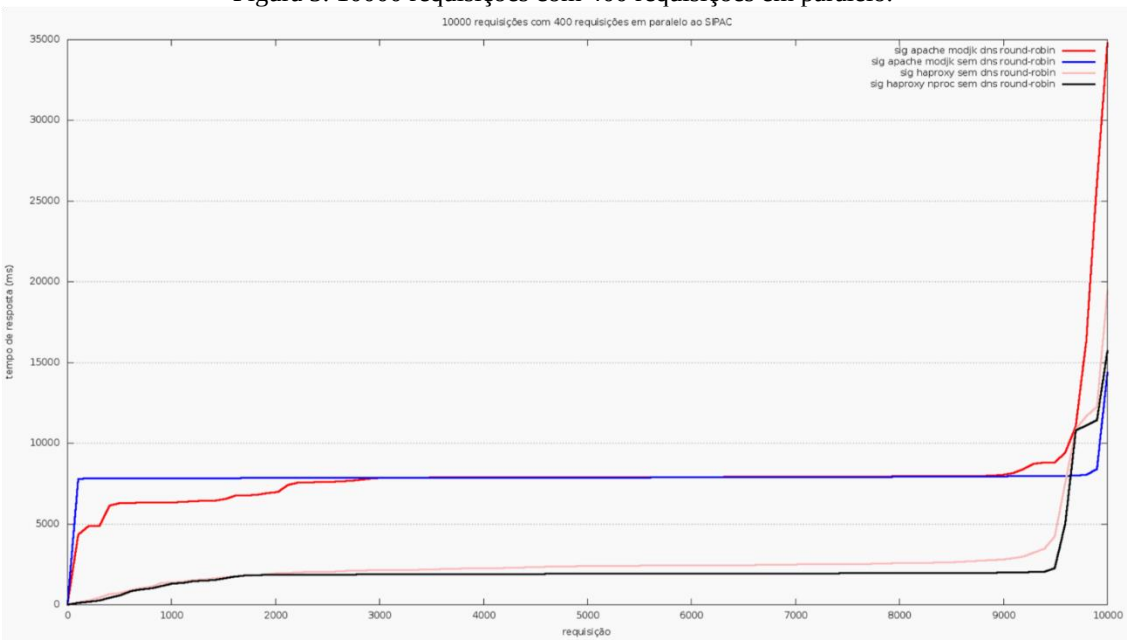


Figura 6. 10000 requisições com 600 requisições em paralelo.

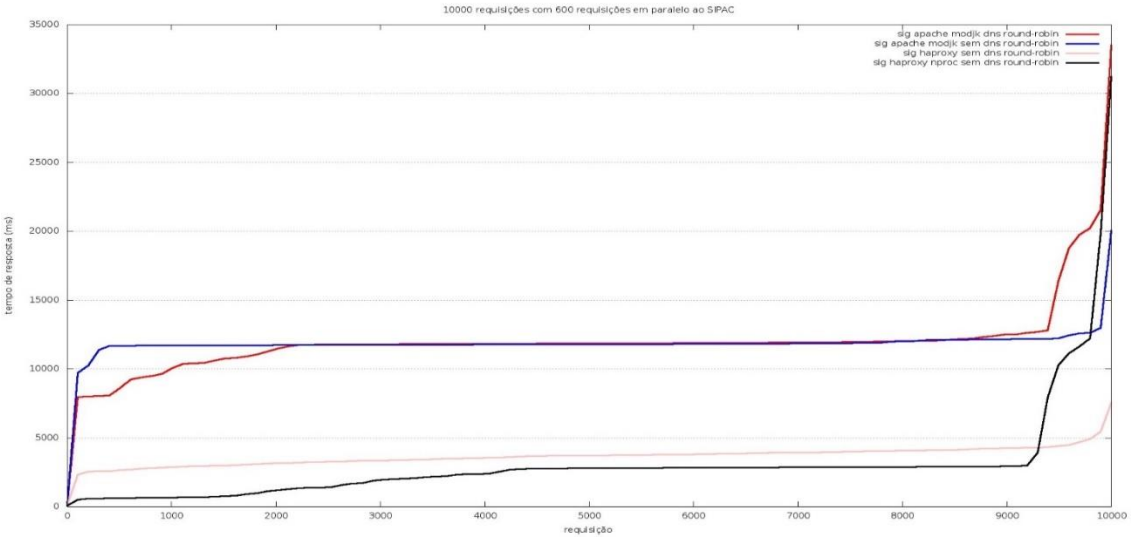


Figura 7. 10000 requisições com 800 requisições em paralelo.

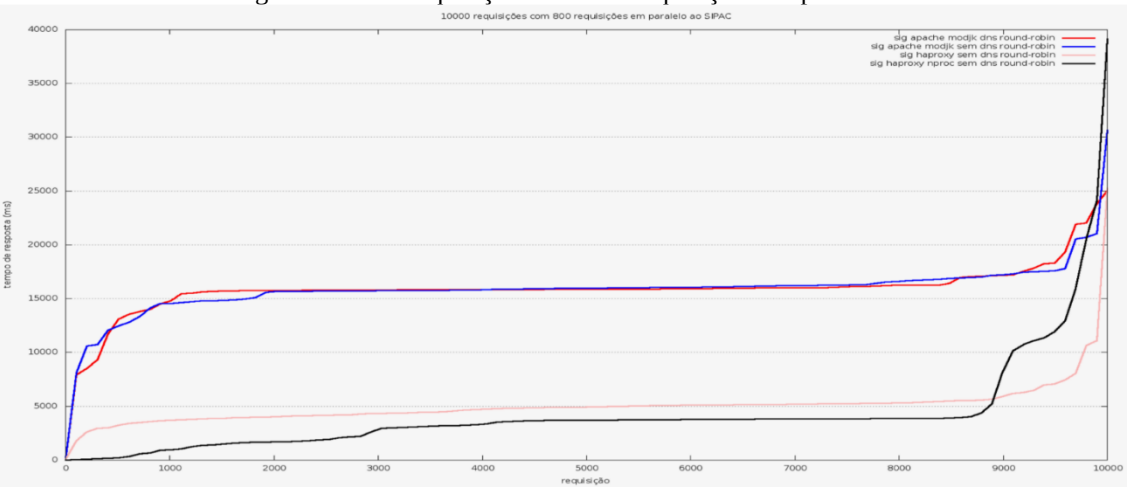
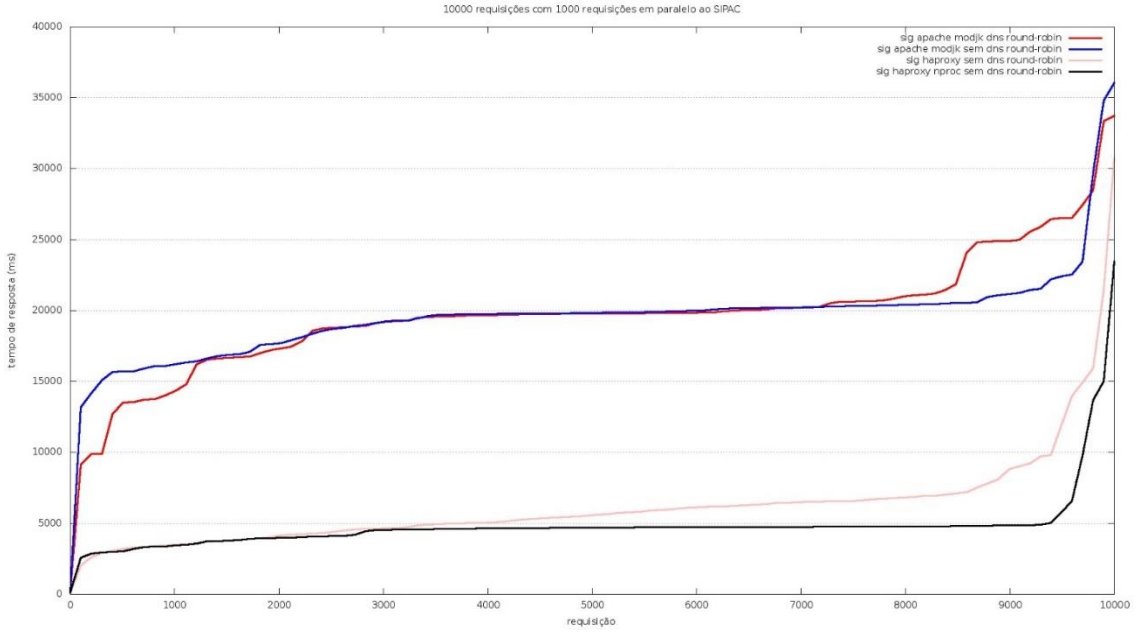


Figura 8. 10000 requisições com 1000 requisições em paralelo.



É possível verificar nas figuras apresentadas que o HAProxy [8] possui melhor desempenho às requisições, com uma pequena melhoria ao operar com multiprocessamento.

Notou-se também que o DNS Round-Robin não apresentou impacto nas requisições atendidas pelo Apache Server [6] com mod_jk [7], como se esperava. Isto se deve ao fato de que o AB [10] executa apenas uma requisição ao DNS antes de iniciar o teste de carga.

4 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma comparação entre Apache Server com Apache Tomcat Connector mod_jk e o HAProxy para balanceamento de carga em servidores de aplicação Java. Tal comparação é oportuna e necessária, uma vez que o HAProxy necessita de menos requisitos de hardware e apresenta desempenho superior, de acordo com seu desenvolvedor. De modo a verificar a veracidade da informação, bem como o desejo de diminuir os custos de implementação da infraestrutura computacional necessária ao SIG, realizou-se cinco testes de carga diferentes em quatro infraestruturas diferentes, totalizando 20 testes.

Verificou-se que a melhor opção foi, sem dúvida, a utilização do software HAProxy [8] para o balanceamento de carga entre os servidores de aplicação do SIG. Porém, como medida de alta disponibilidade, recomenda-se o uso de pelo menos, dois

servidores HAProxy [8] com os domínios do SIG configurados como Round-Robin no DNS.

Uma vez que os testes sobre o impacto do DNS Round-Robin não foram realizados devido à implementação do software AB [10], é prudente aprofundar os estudos em busca de ferramentas de teste de cargas capazes de realizar uma consulta no DNS para cada requisição a ser disparada.

Recomenda-se também configurar o HAProxy [8] para utilização de vários núcleos de CPU para máximo desempenho. Porém, é imprudente utilizar todos os núcleos do servidor: reserve pelo menos dois núcleos para todas as outras atividades do computador servidor. Desta forma evita-se o congestionamento de tarefas no servidor no momento em que o HAProxy [8] receber grandes quantidades de conexões simultâneas.

Sobre a dificuldade de implementação dos balanceadores de carga, o HAProxy [8] também foi melhor ao compará-lo ao Apache Server [6] com mod_jk [7]: sua configuração é armazenada em um único arquivo e com poucas linhas. Também não necessita a instalação de módulos adicionais para o seu funcionamento, como é o caso do Apache Server [6].

Finalmente, temos economia. Enquanto a estrutura oficial da UNIFEI utiliza 8 servidores balanceadores Apache Server [6] com mod_jk [7], equipados com 16 processadores (64 núcleos de CPU) e 256 GB de RAM para oferecer 19.200 requisições, temos que o HAProxy [8] pode encaminhar 20.000 requisições para cada GB de RAM. Ainda, é otimizado para apenas um núcleo de processamento. Desta forma, pode-se utilizar uma solução de balanceamento de carga a baixo custo.

REFERÊNCIAS

[1]Tony Bourke, “Server Load Balancing”, "O'Reilly Media, Inc.", 2001 - 175 páginas.

[2]Liu, C., Albitz, P., “DNS and BIND”, 5th Edition, O'Reilly, 2006.

[3]RFC 1794 – DNS Support for Load Balancing, Abril 1995. Disponível em: tools.ietf.org/html/rfc1794.

[4]Tavares, Diego M. L., Bessa, Tiago M., Paiva, Lais Sousa de., Vilhena, Andressa P. N. M., Sousa, Adilson da Silva. Revisão Sistemática de Publicações Brasileiras Associadas à Teoria das Filas e Sistemas de Processos de Filas. Brazilian Applied Science Review. Curitiba, v.5, n.2, p. 1273-1285 mar./abr. 2021.

[5]Barazandeh, I., Mortazavi, S. S., “Two Hierarchical Dynamic Load Balancing Algorithms inDistributed Systems”, ICCEE '09 Proceedings of the 2009 Second International Conference on Computer and Electrical Engineering – Volume 01, 2009.

[6]Apache Server Project. Disponível em: httpd.apache.org. Acesso em 16 de março de 2016.

[7]Apache Tomcat Connectors / mod_jk. Disponível em: tomcat.apache.org/connectors-doc. Acesso em 16 de março de 2021.

[8]HAProxy. Disponível em: www.haproxy.org. Acesso em 16 de março de 2021.

[9]Load Balancer Configuration Tool. Disponível em: access.redhat.com/labs/lbconfig. Acesso em 16 de março de 2016.

[10] Apache HTTP server benchmarking tool. Disponível em: httpd.apache.org/docs/2.4/programs/ab.html. Acesso em 16 de março de 2021.