Variações de teste de desempenho aplicados à ferramenta de recuperação de fórmulas matemáticas *SearchOnMath*

Felipe A. Freire
Universidade Federal de Alfenas
Alfenas, Minas Gerais
felipea.freire@bcc.unifal-mg.edu.br

Thiago M. Meloto
Universidade Federal de Alfenas
Alfenas, Minas Gerais
thiago.meloto@bcc.unifal-mg.edu.br

Flavio B. Gonzaga
Universidade Federal de Alfenas
Alfenas, Minas Gerais
fbgonzaga@bcc.unifal-mg.edu.br

ABSTRACT

With the amount of data on the Internet growing continuously, the demand for efficient information retrieval systems increases. Even though the area of mathematical information retrieval is relatively new, the amount of data offered about it is significant. SearchOnMath, a pioneering tool in the subject, has thousands of formulas indexed in its database. Since there are many mathematical problems that have not yet been solved, the mathematical formulas recovery branch can be quite demanding to aid in the algorithms that propose solutions to such problems. Aiming at a possible expansion of the use of the tool, the present work is based on analyzing the behavior of the system, through two variations of performance test; The load test and the server stress test. For this, scenarios that simulate light, intermediate and heavy loads of concurrent users are applied to the application server.

KEYWORDS

Load Test, Stress Test, JMeter, SearchOnMath

ACM Reference format:
Felipe A. Freire, Thiago M. Meloto, and Flavio B. Gonzaga. 2017. Variações de teste de desempenho aplicados à ferramenta de recuperação de fórmulas matemáticas *SearchOnMath*. In *Proceedings of , , ,* 6 pages.
https://doi.org/

1 INTRODUÇÃO

O aumento expressivo de informação em toda a Internet passou a oferecer um número crescente de coleções de documentos disponíveis para acesso. Essas coleções de documentos envolvem, na maioria das vezes, dados não estruturados, o que exige sistemas de recuperação de informações de textos completos, com acesso concorrente, coordenado e distribuído[3].

Ferramentas tradicionais de recuperação de informação, como Google e Yahoo, seguem o método clássico de busca por palavras, mas o foco das empresas vem se concentrando em buscas de conteúdo específico. Um exemplo é a busca por artigos científicos, oferecida pela Google através da ferramenta Scholar [1]. Seguindo esse paradigma, a SearchOnMath[2], que teve sua primeira versão lançada em 2013, é uma ferramenta de busca usada para encontrar páginas que contenham fórmulas matemáticas semelhantes às determinadas pelos usuários[2].

O ramo de recuperação de fórmulas matemáticas ainda está em processo de evolução e possui aplicações a curto, médio e longo prazo, podendo no futuro obter algoritmos que proponham soluções para problemas matemáticos ainda não resolvidos[7]. Por isso, a SearchOnMath, que já vem se expandindo no decorrer dos últimos anos, e que ainda pretende um crescimento futuro, encontra a necessidade de explorar as capacidades do sistema, e para obter resultados é preciso realizar testes na aplicação. Testes de desempenho, serão aplicados, com o intuito de refletir diretamente o comportamento da aplicação em diferentes cenários de uso.

2 METODOLOGIA

Com base no trabalho desenvolvido como defesa de conclusão de curso[4], a configuração ideal para o sistema distribuído aplicado à *SearchOnMath*, seria a que abriga 17 máquinas, sendo 16 *slaves* e 1 máquina denominada *master*. O presente estudo realiza então testes com o objetivo de medir a capacidade dessa configuração em atender uma demanda crescente de usuários.

Existem algumas ferramentas de código aberto para testes de serviços web disponíveis no mercado de software. As finalidades podem ser parecidas, porém elas diferem em usabilidade e recursos. O JMeter[3] é uma ferramenta de teste de código aberto, desenvolvida pela Apache, que tem como função principal testar o carregamento de uma aplicação cliente/servidor[1]. A ferramenta oferece interface gráfica do usuário (GUI) amigável, exigindo pouco esforço para a construção de um plano de teste. Além disso, é bastante recomendada para testes de performance, que no caso deste estudo, se concentrou em duas variações, o teste de carga e o de estresse de servidor[6]. Foi utilizada a versão mais recente da ferramenta (3.2) na data de realização do presente trabalho.

Das variações de testes de performance, consideramos utilizar duas delas: o teste de carga e o teste de estresse de servidor. O teste de carga fornece informações sobre o funcionamento do sistema na condição diária, ou seja, ele tem como objetivo reproduzir o comportamento do usuário no mundo real, incluindo atrasos no tempo de reflexão. Já o de estresse, analisa como o sistema irá se comportar com os piores cenários possíveis, supondo grande número de usuários simultâneos [5].

Tanto para o teste de carga, quanto para o de estresse, os cenários são definidos determinando primeiramente o número de usuários simultâneos. Em ambos, é preciso começar com uma carga menor de usuários, e então, gradativamente, aumentá-la. Sendo assim, os

[1] https://scholar.google.com.br/
[2] http://www.searchonmath.com/
[3] http://jmeter.apache.org/

, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/

cenários foram divididos em seis casos: as menores cargas foram definidas como sendo 50 e 100 usuários simultâneos; as intermediárias com 500 e 1 000; e as maiores com 2 000 e 5 000. A execução simula um usuário que abra a página inicial da ferramenta, e em seguida, cole uma fórmula a ser buscada. Logo, cada usuário emitirá duas requisições: uma de acesso da página inicial da SearchOnMath, referenciada como *home*; e outra de submissão de busca, referenciada como *search*. Ao executar o JMeter, o mesmo dispara uma *thread* para cada usuário que se deseja simular. Assim, para 50 usuários, 50 *threads* serão disparadas. Na Figura 1 é possível enxergar como esse intervalo funciona. No caso do teste de performance, considerou-se a taxa de chegada de usuários como sendo igual a dois por segundo em média. Logo, para o caso de 50 usuários, gasta-se um intervalo de 25 segundos para que todas as threads sejam iniciadas. Já no teste de estresse, todas as *threads* são iniciadas ao mesmo tempo.

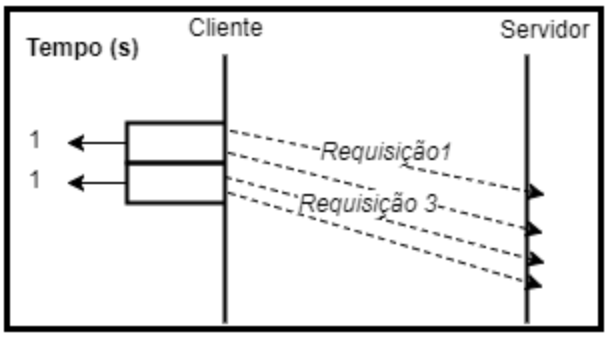

Figura 1: Ilustração do intervalo de inicialização das *threads*
Fonte: Própria

Para a escolha das fórmulas aplicadas aos referidos testes, foram feitos levantamentos baseados no banco de dados da SearchOnMath. A base de dados possui 1 905 358 fórmulas indexadas. Após a análise de todas elas, foram selecionadas uma fórmula do menor tamanho, uma do tamanho médio, e a maior existente na base de dados. A essas três fórmulas chamou-se de Menor, Médio, e Maior caso respectivamente, conforme pode-se observar na Tabela 1.

Tabela 1: Fórmulas Utilizadas

Caso	Fórmula
Menor	e_j
Médio	$Q(t^2) = f + gt^2 + ht^4$
Maior	**Quantum Mechanical*[4]

As fórmulas estão divididas internamente na SearchOnMath entre 28 grupos, definidos segundo diferentes características que uma fórmula possa ter. Assim, cada grupo possui fórmulas com características semelhantes entre si. Quando o usuário solicita uma busca, com base na característica da fórmula do usuário, algum dos grupos em questão é carregado para que a busca seja processada. Se não houvesse essa classificação prévia, todo o universo de 1 905 358 fórmulas precisaria ser analisado em cada pesquisa. É uma estrutura

[4] en.wikipedia.org/wiki/Quantum_mechanical_scattering_of_photon_and_nucleus

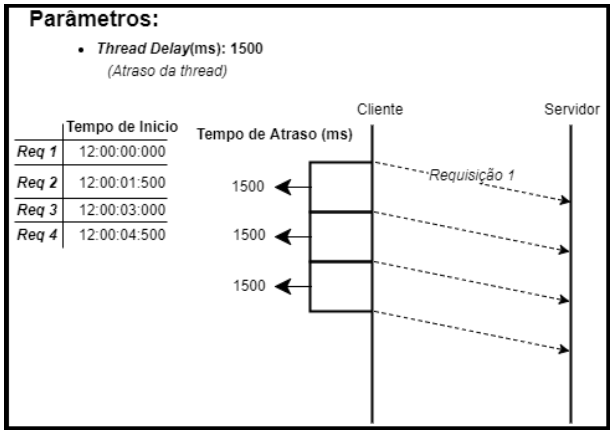

Figura 2: Funcionamento do *Constant Timer*
Fonte: Própria

bastante semelhante ao índice invertido, utilizado em ferramentas de busca textuais, sendo que quanto maior o tamanho do grupo, maior será o tempo de resposta da pesquisa feita. As fórmulas utilizadas nos testes (exibidas na Tabela 1) estão todas presentes no maior grupo da SearchOnMath, o que siginifica que já resultarão em maior tempo de processamento no servidor.

Buscando modelar as atividades de usuários de forma mais realista durante nossas experiências com os testes de carga, simulamos o tempo de reflexão do usuário (*"think time"*), ou seja, o tempo que ele leva para pensar e aplicar os comandos na aplicação[3]. Esse modelo de simulação é oferecido pelo Apache JMeter através de temporizadores (*"timers"*), que oferecem algumas alternativas de reprodução destes atrasos praticados pelos usuários. Esse tempo de reflexão do usuário será aplicado na simulação entre o acesso da página inicial e a busca definida pelo usuário. No presente estudo, foram manipulados quatro temporizadores diferentes; *"Constant Timer"*, *"Gaussian Random Timer"*, *"Poisson Random Timer"* e *"Uniform Random Timer"*. A Figura 2 ilustra o funcionamento do temporizador *"Constant Timer"*, um dos quatro utilizados nos testes.

Todos os quatro temporizadores foram configurados para gerar um atraso entre requisições. Uma vez que um usuário realize o primeiro acesso (à página *home*), ele aguardará um intervalo de tempo antes de solicitar a busca. No *"Constant Timer"* esse tempo é igual a 1 500 milissegundos (ms). O *"Gaussian Random Timer"*, define seu atraso baseando se na distribuição normal padrão, sendo estipulado neste caso, um retardo que varia de 500 ms a 1 500 ms. Já o atraso determinado pelo *Poisson Random Timer*, segue a distribuição de Poisson, na qual as requisições serão lançadas a cada 1 500 ms em média. E por fim, o *Uniform Random Timer*, define o atraso de forma aleatória, selecionando um valor entre 0 e 1 500 ms para indicar o tempo entre as requisições.

3 RESULTADOS

A principal dificuldade encontrada nos testes de desempenho é definir os parâmetros adequados e analisar os resultados dos testes. Embora a configuração dos parâmetros dependa do comportamento do cenário específico, a análise do resultado pode ter que ser feita

por uma referência padrão. No entanto, uma análise adequada pode proporcionar um conhecimento mais aprofundado sobre o desempenho da aplicação [5].

A análise será baseada em cima de dois parâmetros, o tempo de carregamento médio das páginas (*"Sample Time"*) e a porcentagem média de erros (*"Percentage of Errors"*), que representa a porcentagem de requisições não respondidas pelo servidor.

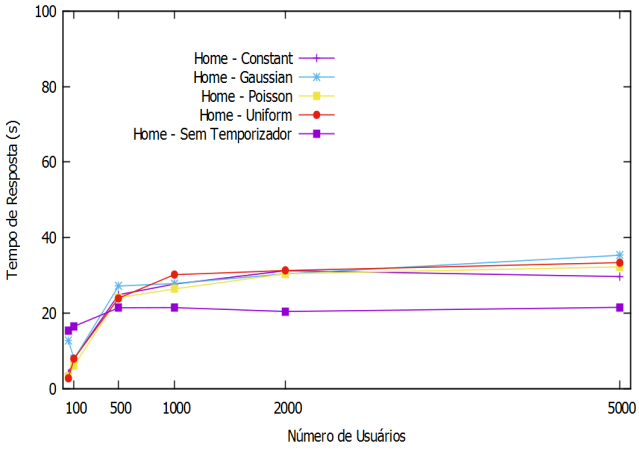

Figura 3: Tempo de Carregamento do Caso Menor - Home
Fonte: Própria

Com os parâmetros definidos, gráficos apontaram comparações considerando a simulação de diferentes cargas de usuários, seja com o uso dos quatro temporizadores citados, como também para o caso que representa o teste de estresse e que não utiliza temporizador. Cada um dos três cenários de fórmulas (menor, média e maior) serão tomados, levando em conta a página inicial para cada caso.

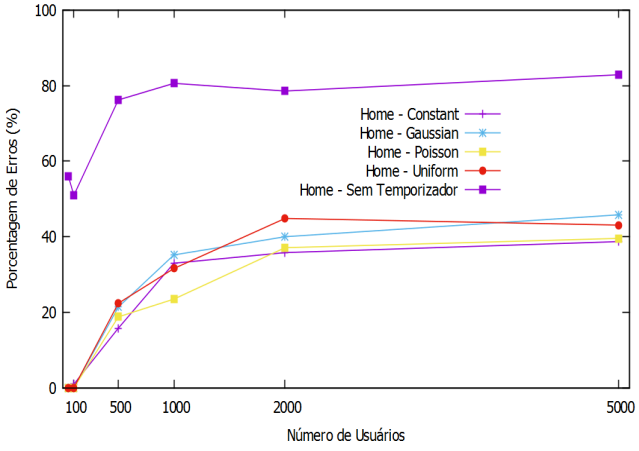

Figura 4: Porcentagem Erro do Caso Menor - Home
Fonte: Própria

Começando pelo cenário de menor fórmula, observando a Figura 3, que mostra o tempo médio de carregamento da página inicial,

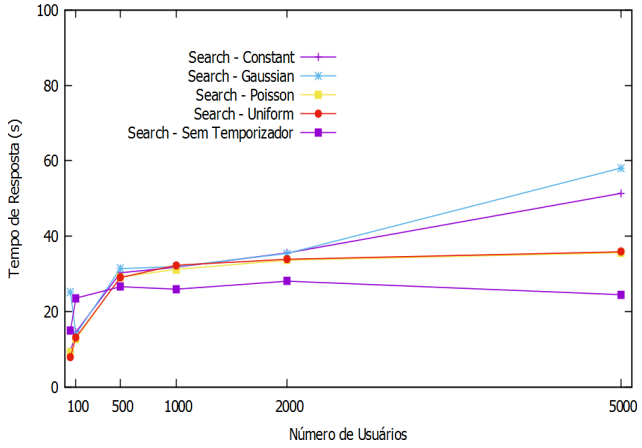

Figura 5: Tempo de Carregamento do Caso Menor - Search
Fonte: Própria

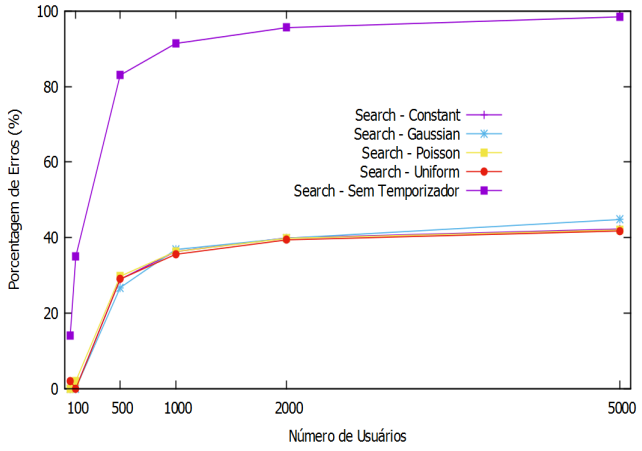

Figura 6: Porcentagem Erro do Caso Menor - Search
Fonte: Própria

podemos notar a semelhança no tempo médio entre todos os temporizadores independente do número de usuários. Percebe-se também um tempo baixo para o cenário que lança todas as requisições ao mesmo tempo, em relação ao cenário com uso de temporizadores.

Agora, analisando a Figura 4, a porcentagem de erros para a página inicial, para os casos de carga intermediária (500 e 1000 usuários) e pesada (2 000 e 5000 usuários), a taxa varia entre 20% e 40%. Para os cenários de carga leve (50 e 100 usuários), a porcentagem fica sempre próxima de 0%, considerando todos os temporizadores. Tomando o caso que não utiliza temporizador, a porcentagem de erros ultrapassa os 50%, mesmo para cargas leves. Nas demais cargas o erro aproxima de 80%, explicando assim a pequena quantidade de tempo gasta no carregamento da página, já que a taxa de sucesso foi pequena.

Voltando as atenções para a busca, a Figura 5 mostra que os tempos de resposta foram semelhantes aos da página inicial, com o *Gaussian* novamente apresentando maior tempo para o caso de

5000 usuários, mas dessa vez com um tempo mais baixo, cerca de 60 segundos. Considerando o cenário sem temporizador, o tempo foi baixo, ficando na faixa de 20 a 30 segundos.

Partindo para a Figura 6, a análise do gráfico com o uso dos temporizadores mostrou taxa de erros semelhante nos diferentes timers, variando de 30% a 40% para as cargas intermediárias e pesadas. Sem temporizador, a taxa de erros apresentou-se elevada para os cenários de carga intermediária e pesada, ultrapassando a marca dos 90%, o que novamente deixa claro o porquê de o tempo de carregamento ter sido baixo, os casos de sucesso foram poucos.

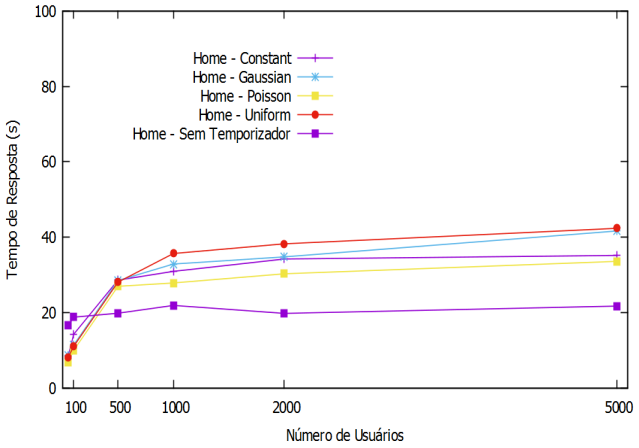

Figura 7: Tempo de Carregamento do Caso Médio - Home
Fonte: Própria

Considerando o cenário de fórmula de tamanho médio, representado na Figura 7, o tempo de carregamento da página inicial não possui valores muito diferentes se comparado ao tempo de carregamento da página inicial no caso de menor fórmula, representado na Figura 3.

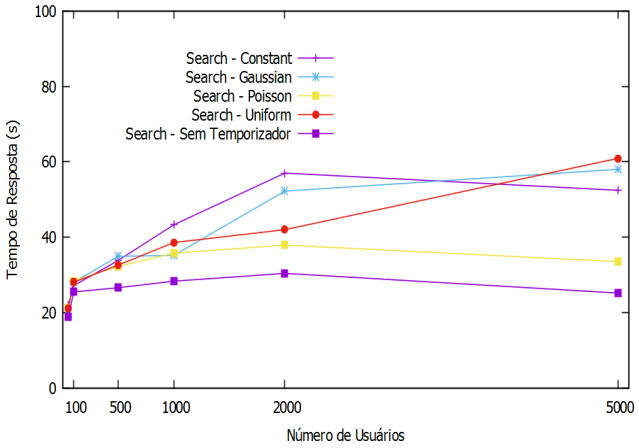

Figura 8: Tempo de Carregamento do Caso Médio - Search
Fonte: Própria

É possível observar na Figura 8, em relação ao tempo de carregamento da página utilizando a fórmula de tamanho médio, os temporizadores *Gaussian*, *Uniform* e *Constant*, demonstram valores maiores que os demais temporizadores. O valor do temporizador Poisson para o caso de 5 000 usuários, mostra valor semelhante ao mesmo temporizador, para o caso de menor fórmula, representado na Figura 5. Constata-se que neste cenário as fórmulas de tamanho menor e médio não apresentaram diferenças significativas para seu carregamento.

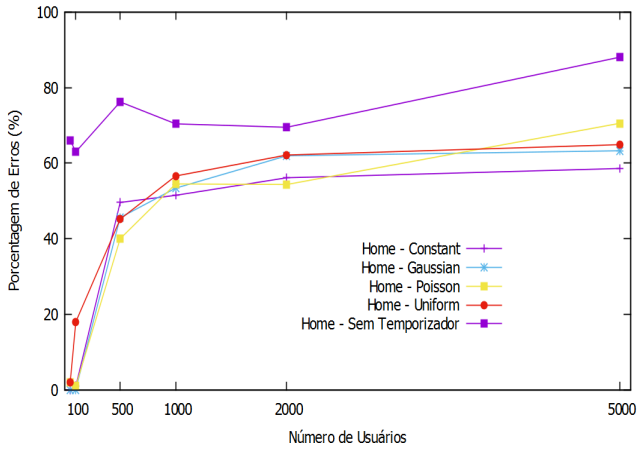

Figura 9: Porcentagem Erro do Caso Médio - Home
Fonte: Própria

A porcentagem de erros para o caso médio no tempo de carregamento da página inicial, visto na Figura 9, possui valores muito semelhantes para os temporizadores *Constant*, *Gaussian*, *Poisson* e *Uniform*. Também estes mesmos temporizadores possuem valores maiores no caso médio (aproximadamente 60%), comparados aos valores no caso menor com o mesmo cenário (aproximadamente 40%), que estão representados na Figura 4.

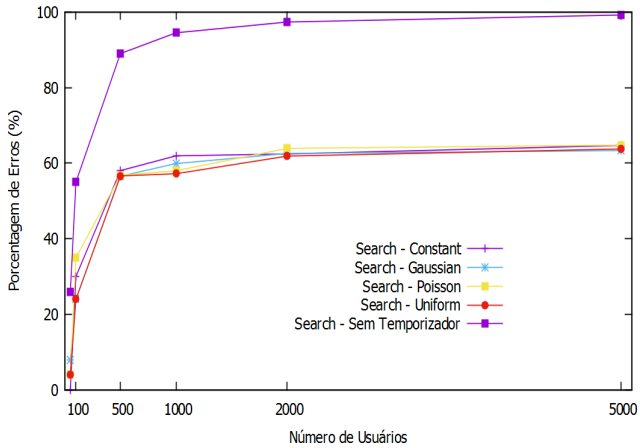

Figura 10: Porcentagem Erro do Caso Médio - Search
Fonte: Própria

Somente no carregamento sem uso de temporizador, que a porcentagem de erros para todas as quantidades de usuários testadas é muito superior em relação aos temporizadores, tanto no carregamento da página inicial (aproximando de 90%), quanto no carregamento da fórmula média (aproximando de 100%), como demonstrado na Figura 10.

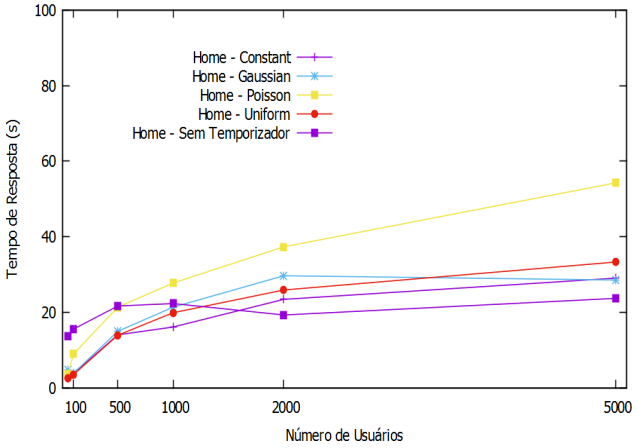

Figura 11: Tempo de Carregamento do Caso Maior - Home
Fonte: Própria

No cenário de maior fórmula, a partir da Figura 11, o tempo de resposta da página inicial aponta para valores que se concentram entre 20 e 30 segundos. O único com um valor um pouco maior, em torno de 50 segundos, foi o temporizador *Poisson*, que pode ser explicado por uma variância da velocidade da rede no decorrer dos testes. Como no caso da fórmula menor, para as requisições enviadas ao mesmo tempo, o tempo médio de resposta foi relativamente baixo, o que se explica pela alta porcentagem de erros apresentada.

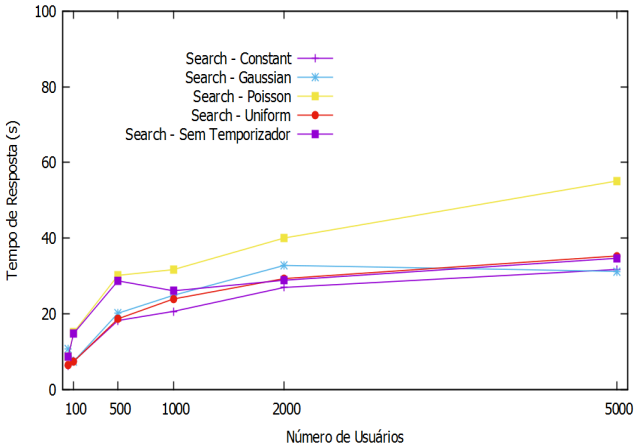

Figura 12: Tempo de Carregamento do Caso Maior - Search
Fonte: Própria

Analisando os tempos para a busca na Figura 12, novamente observamos uma quantidade de tempo maior, apenas para o *Poisson Timer*. Para os outros temporizadores, é possível enxergar uma variação que fica entre 30 e 40 segundos.

Em relação aos erros, presentes nas Figuras 13 e 14, quando não é usado um temporizador ocorre a mesma situação no carregamento do página inicial e na busca pela fórmula maior. A porcentagem de erros chega mais perto da marca de 90%, no momento em que a quantidade de usuários aumenta de 2 000 para 5 000.

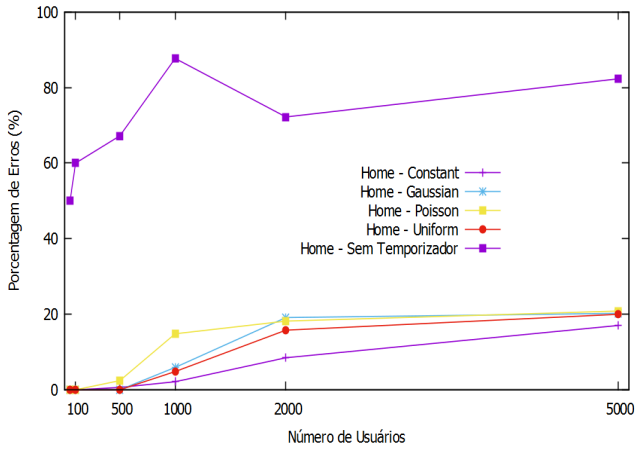

Figura 13: Porcentagem Erro do Caso Maior - Home
Fonte: Própria

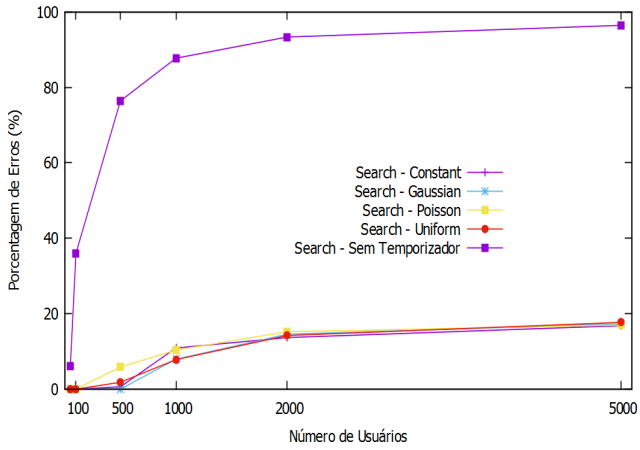

Figura 14: Porcentagem Erro do Caso Maior - Search
Fonte: Própria

4 CONCLUSÕES

Este foi o primeiro estudo dessa natureza conduzido no laboratório com a finalidade de coletarmos alguns parâmetros iniciais do sistema distribuído da ferramenta de busca SearchOnMath. É possível ver que, de forma geral, a ferramenta apresentaria bom tempo de resposta para algo em torno de 50 a 100 usuários simultâneos, isso

considerando casos extremos, onde todos os usuários submeteriam fórmulas para o maior grupo interno da ferramenta, e com intervalos entre acesso à página *home* e à submissão da busca, de cerca de 1 500 ms. Esse tempo é suficiente para se colar uma fórmula no campo de busca, mas é um tempo curto se pensarmos que o usuário pode também montar a fórmula usando o teclado matemático que a ferramenta oferece.

Os dados atuais da SearchOnMath mostram que em média a ferramenta fica com carga inferior a 10 usuários simultâneos, o que indica que há bastante espaço para expansão e investimento em propaganda (com a finalidade de aumentar o número de acessos), antes que seja necessária alguma expansão no sistema distribuído desenvolvido.

A ORGANIZAÇÃO

A.1 Introdução

A.2 Metodologia

A.3 Conclusões

A.4 Referências

REFERÊNCIAS

[1] Apache JMeter 2017. The Apache Software Foundation. (2017). http://jmeter.apache.org/ Acessed: May. 19, 2017.

[2] Flavio Barbieri Gonzaga. 2007. *Recuperação de Informação Orientada ao Domínio da Matemática.* Ph.D. Dissertation. Universidade Federal do Rio de Janeiro.

[3] Brendon Cahoon; Kathryn S. McKinley and Zhihong Lu. 2000. Evaluating the performance of distributed architectures for information retrieval using a variety of workloads. *ACM Transactions on Information Systems (TOIS)* 18, 10 (2000), 1–43. https://doi.org/citation.cfm?doid=333135.333136

[4] Ricardo M. Oliveira. [n. d.]. A Distributed System for SearchOnMath Based on the Microsoft BizSpark Program. ([n. d.]). Artigo não publicado até o momento.

[5] B. M. Subraya and S. V. Subrahmanya. 2002. Object Driven Performance Testing of Web Applications. *IEEE* 38, 4 (2002), 17–26. https://doi.org/document/883774/

[6] Shariq Hussain; Zhaoshun Wang; Ibrahima Kalil Toure and Abdoulaye Diop. 2013. Web Service Testing Tools: A Comparative Study. *IJCSI International Journal of Computer Science Issues* 10, 1–3 (2013), 641–647.

[7] A. Youssef. 2006. Roles of Math Search in Mathematics. In *The 5th International Conference on Mathematical Knowledge Management.* ACM, Wokingham, UK, 2–16.