



Teste de Carga





Grupo


- ★ Douglas Magalhães
- ★ Iara Frias
- ★ Paulo Rodrigues
- ★ Thais Martins
- ★ Victor Garcia



Apache JMeter

JMeter é uma ferramenta que realiza testes de carga e de estresse em recursos estáticos ou dinâmicos oferecidos por sistemas computacionais.

Para a realização de testes, a ferramenta **JMeter** disponibiliza diversos tipos de requisições e assertions (para validar o resultado dessas requisições), além de controladores lógicos como loops (ciclos) e controles condicionais para serem utilizados na construção de planos de teste, que correspondem aos testes funcionais.



O **JMeter** disponibiliza também um controle de threads, chamado Thread Group, no qual é possível configurar o número de threads, a quantidade de vezes que cada thread será executada e o intervalo entre cada execução, que ajuda a realizar os testes de estresse.

Existem diversos listeners que, baseando-se nos resultados das requisições ou dos assertions, podem ser usados para gerar gráficos e tabelas.

Portabilidade completa e 100% Java.

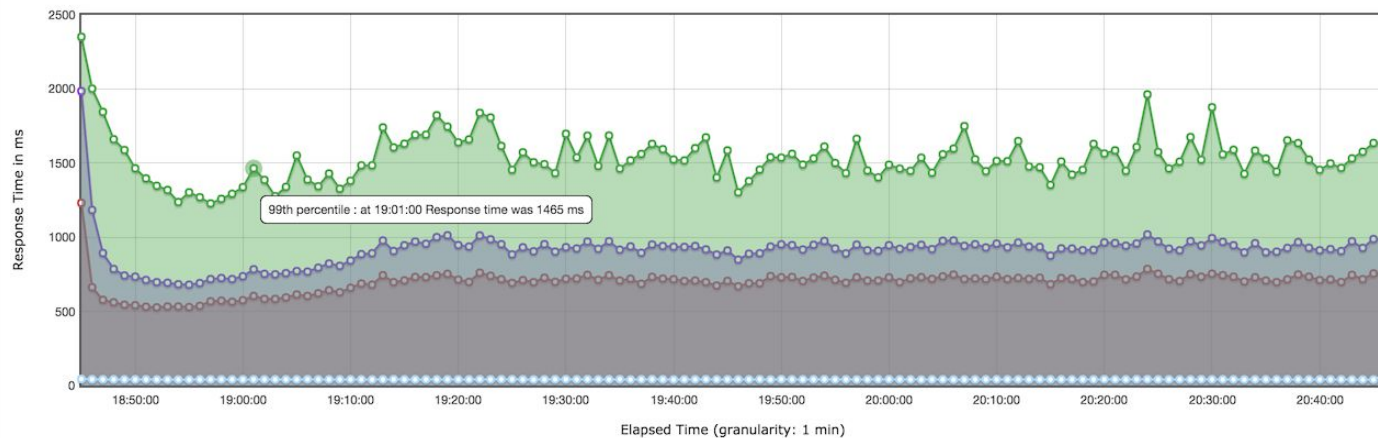


RELATÓRIOS

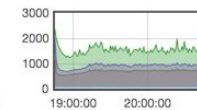
Relatório HTML completo e pronto para ser apresentado.



Response Time Percentiles Over Time (successful responses)



Zoom :



90th percentile 95th percentile 99th percentile Max Min



PROTOCOLOS

Capacidade de carregar e testar o desempenho de aplicativos, sistemas, sites e webservices.

- Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET,...)
- Webservices SOAP / REST
- FTP
- Banco de dados via JDBC
- LDAP




RECURSOS

Para qualquer teste que venha a ser feito utilizando o JMeter, é necessário criar um **Test Plan** incluindo os elementos do teste. Tais elementos podem ser:

Thread Group — É o ponto de início, sob o qual devem estar todos os outros elementos do Test Plan. Como o próprio nome ressalta, ele controla as threads que serão executadas pelo teste.


Controllers — São divididos em dois grupos: samplers e logic controllers.



Samplers — São controladores pré-definidos para requisições específicas. Podem ser personalizados com a inserção de configurações (configurations), asserções (assertions) etc.

Logic Controllers — São controladores mais genéricos. Podem ser personalizados com a inserção de outros controllers, configuration elements, assertions etc.


Listeners — São elementos que fornecem acesso às informações obtidas pelo JMeter durante os testes.



Timers — Por padrão, o JMeter faz requisições sem pausas entre elas. Os timers são utilizados para incluir pausas entre as requisições.

Assertions — Usado para verificar se a resposta obtida na requisição é a esperada. Expressões regulares (Perl-style regular expression) podem ser usadas na comparação.

Configuration Elements — Embora não faça requisições (exceto para HTTP Proxy Server), este elemento pode adicionar ou modificar as requisições.



Pre-Processor Elements — Executa alguma ação antes de fazer a requisição. É mais usado para pré-configurações das requisições.


Post-Processor Elements — Executa alguma ação depois de fazer a requisição. É mais usado para processar as respostas da requisição.


TIPOS DE SERVIÇOS



O JMeter suporta requisições dos seguintes tipos de serviços em suas rotinas de testes:

- FTP — Permite criar requisições usando o protocolo FTP (com autenticação ou não) e executa o comando de *retrieve* em um arquivo específico.
- HTTP — Permite criar requisições usando o protocolo HTTPS ou HTTP (com ou sem autenticação, respectivamente), podendo incluir parâmetros ou arquivos de requisição, escolher o método usado (*GET* ou *POST*) e manipular *Cookies*. Este *sampler* possui dois tipos de implementação: *Java HTTP* ou *Commons HTTPClient*.

- 
- JDBC — Com esta requisição é possível executar *queries* em um banco de dados específico.
 - Objeto Java — Auxilia no teste de carga de classes Java, exigindo que se implemente uma classe do tipo *JavaSamplerClient* para executar o método a ser testado. A estrutura deste objeto é similar à usada pelo *JUnit*.
 - SOAP/XML-RPC — Permite enviar requisições *SOAP* para um *WebService*, ou enviar XML-RPC através do protocolo HTTP.

- 
- LDAP — Permite enviar requisições para um servidor LDAP. Possui uma implementação simplificada e outra estendida.
 - Testes JUnit — Usado para fazer teste de carga em testes de unidade que utilizam o *framework* JUnit.


Existem outros tipos de requisições que estão em versão *alpha*. São eles: Web service (SOAP), Access Log, BeanShell, BSF, TCP, JMS Publisher, JMS Subscriber, JMS Point-to-Point.



Importância e a necessidade de implementação do nível de teste

O objetivo da estratégia do JMeter é prover cenários de testes mais reais a forma de uso dos sistemas testados.

Portanto, os testes de carga devem simular o mais próximo possível da realidade de utilização, cenários realistas ajudam a minimizar os efeitos da subestimação ou superestimação dos tempos de resposta das aplicações.



Tempo pra pensar: Tempo para que o usuário pare (pense) durante suas interações com o sistema. Faz os usuários virtuais se comportarem como se fossem usuários reais. Por meio deste tempo, são injetados atrasos e paradas variáveis nos testes para simular cargas de utilização mais realistas.

Cache de navegador: Permite que requisições estáticas como imagens, folhas de estilos, scripts entre outros sejam realizadas apenas uma vez por usuário, sabendo que o comportamento padrão dos navegadores de mercado fazem cache de arquivos que não são alterados com frequência.

Concorrência: Significa vários usuários usando a aplicação em diversas funcionalidades no mesmo período.



Requisições derivadas: Requisições feitas a partir de uma requisição HTTP principal, são elas as requisições para imagens, folhas de estilo e arquivos de script Javascript.

Número de Iterações: Recomenda-se que haja mais de uma iteração para cada thread, preferencialmente que o teste seja feito por um período de tempo que permita a todos os usuários estarem utilizando o sistema concorrentemente. A repetição dos testes por várias iterações permite ao sistema de adaptar ao número de usuários concorrentes tornando os tempos de resposta mais próximos de um cenário real.

Temporizador de Sincronização: O temporizador de sincronização deve ser utilizado em casos que se deseje garantir que uma requisição seja executada por todas as threads (usuários virtuais) ao mesmo tempo. Normalmente a requisição a ser sincronizada encontra-se entre outras requisições que podem tirar a sincronização dos usuários virtuais, como por exemplo o login no sistema, sendo que neste normalmente a funcionalidade que deseja-se testar esta após a entrada dos usuários no sistema.



Dificuldades de Implementação

- Infraestrutura do ambiente de testes diferente do ambiente real de produção
 - Custos para manter outro ambiente igual ao de produção
- Arquitetura do software que esteja apta a execução de teste de carga
- Profissionais capacitados a desenvolver e executar os testes
- Manutenção contínua dos cenários de teste
- Tempo dedicado na execução dos testes



Exemplo de Implementação

E-Commerce se preparando para Black Friday

- Quantidade de acessos simultâneos suportados
- Tempo de resposta das requisições
- Identificar outros pontos de gargalo no sistema
 - Ex: O e-commerce suporta 1500 requisições simultâneas por minuto, porém a intermediadora de pagamentos suporta somente 800



Exemplo de Implementação

Big Brother Brasil em um dia de votação

- Quantidade de requisições simultâneas máxima suportada
- Tempo médio de resposta das requisições
- Estabilidade e disponibilidade do ambiente com cargas altas por longos períodos
- Capacidade de intervir na aplicação, mesmo recebendo uma carga alta
 - Ex: Encerrar a votação



Demonstração Prática