Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Michelle da Nova Brum de Castro

Geração de carga sintética para avaliação de balanceamento de carga em SDN

Juiz de Fora
2015

Michelle da Nova Brum de Castro

Geração de carga sintética para avaliação de balanceamento de carga em SDN

Dissertação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Alex Borges Vieira

Juiz de Fora

2015

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Castro, Michelle da Nova Brum de .
Geração de carga sintética para avaliação de balanceamento de carga em SDN / Michelle da Nova Brum de Castro. – 2015.
39 f. : il.

Orientador: Alex Borges Vieira
Monografia – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Departamento de Ciência da Computação, 2015.

1. Servidor. 2. Carga de trabalho. 3. Teste. I. Vieira, Alex Borges, orient. II. Título.

Michelle da Nova Brum de Castro

Geração de carga sintética para avaliação de balanceamento de carga em SDN

Dissertação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Alex Borges Vieira - Orientador
Universidade Federal de Juiz de Fora

Professor Dr. Heder Soares Bernardino
Universidade Federal de Juiz de Fora

Professor Me. Francisco Henrique Cerdeira Ferreira
Universidade Federal de Juiz de Fora

À Mônica Brum, a eterna e incondicional
incentivadora dos meus sonhos,
a pessoa que sempre estará ao meu lado
em todos os momentos, minha Mãe.

AGRADECIMENTOS

A Deus, pela dádiva da vida, e por ter ajudado a manter a fé nos momentos mais difíceis.

Minha mãe, pela educação que me proporcionou, me concedendo muito além do que teve. Agradeço também a força por ela prestada durante toda sua vida. Por sempre acreditar em mim e me apoiar de incontáveis maneiras. A meu pai, pelo apoio.

Minhas irmãs Danielle e Larissa, pela conversas, companheirismos e incentivos.

Ao meu namorado, Paulo César, por estar sempre presente, me apoiando e acreditando em mim.

Ao meu orientador Alex Borges Vieira, pela ajuda e disponibilidade durante a realização desta obra.

A todos, que mesmo indiretamente, contribuíram para a conclusão deste trabalho.

"A persistência é o menor caminho do êxito."
Charles Chaplin

RESUMO

Atualmente, com o surgimento de novas tecnologias a complexidade da Internet tem aumentado. Consequentemente, os serviços devem ser capazes de lidar com o número crescente de requisições, o que exige um maior desempenho dos servidores *web*. A fim de maximizar o desempenho destes servidores a realização de experimentos de testes de carga são importantes para ajudar a visualizar e compreender como uma aplicacação reage. Neste contexto, o balanceamento de carga pode ser utilizado para distribuir a carga de maneira uniforme. Este trabalho apresenta o ciclo de trabalho de geradores de carga. Exemplificarem este ciclo nas ferramentas geradoras de carga *web* Jmeter e LoadRunner. Mostraremos suas caracteristicas, quais cenários abrangem, como é criado um teste simples e será realizado um comparativo entre eles. Serão criadas cargas de tamanho pequeno, médio e grande. Posteriormente, os resultados serão avaliados através de gráficos que ilustram o comportamento das aplicações *web* testadas em cada cenário.

Palavras-chave: Geradores. Carga de trabalho. Requisições. Servidor. Teste.

ABSTRACT

Today, with the emergence of new technologies the complexity of the Internet has increased. Consequently, the services must be able to handle the growing number of requests, which requires higher performance of servers *web*. In order to maximize the performance of these servers performing load testing experiments are important to help visualize and understand how one reacts aplicacação. In this context, load balancing can be used to distribute the load uniformly. This work shows the load generators duty cycle. Exemplify this cycle the load generating tools *web* Jmeter and LoadRunner. We show its characteristics, which cover scenarios, as it creates a simple test and there will be a comparison between them. Of small, medium and large loads will be created. Subsequently, the results will be evaluated through graphs illustrating the behavior of applications *web* tested in each scenario.

Key-words: Generators. Workload. Requests. Server. Test.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FTP	*File Transfer Protocol*
GB	*Gigabyte*
HP	Hewlett-Packard
HTML	*HyperText Markput Language*
HTTP	*Hypertext Transfer Protocol*
KB	*Kilobyte*
MB	*MegaByte*
SOAP	*Simple Object Access Protocol*
SURGE	*Scalable URL Reference Generator*
UFJF	Universidade Federal de Juiz de Fora
URL	*Uniform Resource Locator*
VUGen	*Virtual User Generator*
XML	*XML Metadata Interchange*
W4Gen	*World Wide Web Workload Generator*
WAN	*Wide Area Network*
WWW	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização e Motivação

Até meados da década de 80, a Internet era utilizada somente em comunidades acadêmicas e órgãos governamentais, basicamente apenas para serviços de correio eletrônico (e-mail) e transferência de arquivos (FTP). Com o surgimento da WWW (World Wide Web) nos anos 90 o conceito de computador pessoal se ampliou, páginas estáticas foram criadas, o acesso se multiplicou fazendo com que a Internet fosse considerada um grande veículo de comunicação do mundo. Nas últimas décadas, páginas estáticas vem sendo substituídas pelas dinâmicas, novas aplicações tais como *e-commerce*, internet *banking* e aplicações multimídias surgiram, gerando um aumento significativo no tráfego. Contudo, conforme Comer (apud TEIXEIRA,2004)[17], originalmente a Internet não foi projetada para o uso atual, sequer para dar suporte à carga que lhe é determinada. Ela surgiu como uma rede comutada por pacotes, de grande abrangência, que objetivava o transporte de dados convencionais entre computadores localizados em diferentes partes do mundo.

Dentro deste cenário, com o aumento de carga nos servidores *web*, se faz necessário que a insfraestrutura e qualidade dos serviço cresçam na mesma proporção. Entender como servidores se comportamentam frente a este tipo de carga é importante para que os sistemas existentes sejam aperfeiçoados. Para isso, simular a geração de cargas de trabalho de usuários reais acessando um servidor auxiliam na compreensão e planejamento para que a entrega deste serviço seja otimizada.

(BARFORD; CROVELLA, 1998) [3] propuseram um modelo chamado SURGE (*Scalable URL Reference Generator*) com o objetivo de descrever adequadamente o tráfego de redes de computadores. Eles propõem a ideia de o usuário como medida de intensidade da carga de trabalho em servidores, ou seja, utilizar o usuário para representar tráfego real em um servidor é uma medida válida para a avaliação de um sistema.

Afim de avaliar o desempenho e a capacidade de servidores e redes surgiram ferramentas que geram cargas (usuários) *web* para testes. Com o avanço das pesquisas para solucionar o problema de estrangulamento do sistema surgiu o paradigma Redes Definidas por Software (SDN), que propoe que os utilizadores dessas redes possam definir o fluxo de dados e seus caminhos usando *software* independente do *hardware*. Assim, surge a necessidade de gerar cargas realistas para avaliar este novo cenário .

Fundamentado nesta necessidade, este trabalho apresenta uma avaliação dos geradores de carga Jmeter[1] e LoadRunner[10]. Estes geradores criam *threads* que simulam usuários requisitando página em um servidor *web*. São eficazes para esboçar a capacidades dos servidores em um determinado quadro de usuários.

1.2 Objetivos

O objetivo principal deste trabalho é gerar cargas realistas de trabalho *web* para uma posterior avaliação. As ferramentas utilizadas para gerar as cargas são JMeter e LoadRunner. É mostrada uma visão geral de cada de um ciclo de trabalho básico de cada uma, ressaltando suas principais caracteristicas. Posteriormente, são realizados teste e uma avaliação das aplicações *web*. Finalmente, é realizado um comparativo entre elas destacando suas principais características.

1.3 Estrutura da Monografia

Este trabalho está dividido em cinco capítulos. O Capítulo 2 mostra um conjunto de características para ferramentas de teste de desempenho e faz um levantamento dos trabalhos relacionados que realizam testes de carga.

No Capítulo 3 é apresentada a carga típica atual, quais cenários mais utilizados .

No Capítulo 4 é apresentada uma descrição das funcionalidades das duas principais ferramentas para automatização de teste de software: JMeter [1] e LoadRunner [10]. As cargas de trabalho reais são geradas aqui.

O Capítulo 5 apresenta uma análise detalhada dos resultados gerados pelas cargas do Capítulo 4 e um compartivo das ferramentas.

Finalmente, no Capítulo 6 são mostradas as conclusões, mostrando uma visão geral do trabalho e possibilidades de trabalhos futuros.

2 GERAÇÃO DE CARGA WEB

2.1 Teste de Software

Com a presença cada vez maior dos *softwares* em nossas vidas os mesmos tem se tornado mais complexos em consequência do surgimento de novas tecnologias. Por isso, a exigência por *softwares* de qualidade aumentou significativamente. Problemas como erro, defeito ou falha trazem altos custos e má reputação com os usuários. Para evitar problemas como este é necessário investir em testes. Entre os diferentes tipos de testes ressaltamos os testes de desempenho.

2.2 Teste de Desempenho

Atualmente, os clientes estão exigindo que as aplicações atendam aos requisitos de desempenho, não somente aos requisitos funcionais. Os crescentes números de usuários têm gerado problemas em aplicações *web*.

Neste contexto, o teste de desempenho possui grande importância, pois visa verificar o comportamento de um sistema de acordo com restrições de recursos e de tempo. Para isso, testam requisitos de desempenho tais como escalabilidade, latência e vazão. O sistema é testado em condições reais de operação. Com ele também é possível identificar prováveis gargalos causadores da diminuição de desempenho no sistema

Segundo (PRESSMAN,2001) [15] testes de desempenho podem ser realizados ao longo do desenvolvimento de um sistema, mas só é possível ter uma avaliação real do seu desempenho quando todos os componentes são integrados. (MOLINARI,2003) [14] diz que o desempenho é o requisito que mais de deve ter dedicação em aplicações *web*. Há muita divergência entre os autores sobre os conjuntos que representam os testes de desempenho, porém os mais comuns são :

• Carga: tem como objetivo principal encontrar o limite de capacidade de uma aplicação. Para isso, o volume gerado é crescente no decorrer do tempo;

• Estresse: o objetivo é determinar a capacidade de estabilidade e recuperação do sistema. Para isso, uma grande carga é disparada simultaneamente contra a aplicação.

Para que a execução desse tipo de teste esteja seja correta é necessário um conjunto de objetivos bem definidos para os requisitos ou, do contrário, esses testes serão considerados medições cegas.

Realizar testes de forma manual pode ser custoso e levar muito tempo. A fim de reduzir a onerosidade é necessário fazer o uso de ferramentas de automação de testes, pois elas reduzem os riscos de defeitos, se tornam mais confiáveis, diminuem os custos do projeto e tempo contribuindo para a qualidade do software.

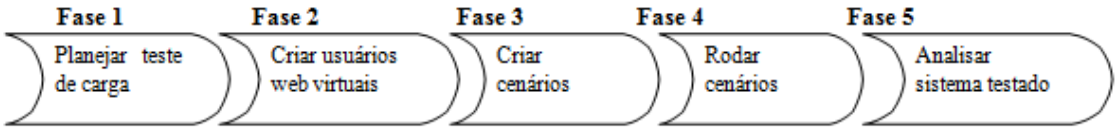
Atualmente, existem varias ferramentas que são capazes de avaliar o desempenho, disponibilidade e a funcionalidades das aplicações dentre elas destacamos Apache JMeter [1], IBM Rational Performance Tester [4] e HP LoadRunner [10] entre outras que tendem reportar problemas de desempenho das aplicações.

Porém, apesar destas ferramentas trazerem benefícios para a automatização de testes, ainda se faz necessária a execução de algumas atividades de forma manual como, por exemplo, a etapa de elaboração dos cenários de teste. Para que o conjunto de características fosse efetivado foi necessário a adoção de uma classificação basica para a criação e execução de testes de desempenho. Essa classificação se baseou no processo de teste utilizado por (MOLINARI,2003) [14], o qual descreve em cinco etapas o processo de geração de casos de teste de desempenho para aplicações.

Outras abordagens foram analisadas como (MICROSOFT, 2004) [13] e (MENASCé; ALMEIDA, 2003)[12], mas devido ao fato das informações das características corresponderem melhor com este processo descrito a abordagem de (MOLINARI, 2003) [14] foi a que melhor se adapta a este conteúdo.

(MOLINARI,2003) [14] propõe um processo de teste de desempenho em aplicações Web como mostra a figura 1abaixo:

Figura 1 – Processo de teste de desempenho proposto por Molinari



• Fase 1 (Planejar teste de carga): nesta etapa o objetivo é identificar e definir quais cenarios devem ser testadas na ferramenta. Os caminhos e atividades mais comuns a serem executados pelos usuários são definidos nesta etapa;

• Fase 2 (Criar usuários *web* virtuais): é nesta fase que ocorre a especificação e criação de usuários *web* virtuais;

• Fase 3 (Criar cenários): as características referentes à configuração de carga são definidas. Nessa etapa, como por exemplo, tempo de duração do teste e quantidade de usuários da rampa de subida;

• Fase 4 (Rodar cenários): fase em que o cenario criado é executado;

• Fase 5 (Analisar sistema testado): Analise dos resultados do sistema testado após o término do teste.

Além de servir como base para a classificação e estruturação das características,

este processo de teste descrito por (MOLINARI,2003)[14] também contribui para a complementação das informações destas características. Afim de automatizar testes baseados nos critérios acima, esta dissertação fará o uso das ferramentas Apache JMeter [1] e HP LoadRunner [10] onde criaremos *scripts* e cenários para avaliação.

2.3 Levantamento de informações

Inicialmente, foi realizado um levantamento bibliográfico sobre trabalhos que fazem referência a criação de *scripts* e cenários de teste utilizando ferramentas de automatização de teste de desempenho. Como base na necessidade de analisar o comportamento de um sistema diante de uma determinada carga foram escolhidas ferramentas que atendessem aos requisitos. A ferramenta LoadRunner[10] foi adotada por possuir uma grande fatia no mercado, já a JMeter[1] foi escolhida por ser uma opção *opensource* e ainda sim ser capaz de realizar os testes com qualidade.

Decididas as ferramentas, posteriormente foi realizada a instalação de ambas com o objetivo de verificar qual o conjunto de etapas é necessário para se criar um *script* e posteriormente um cenario de testes. Foi possível observar as características individuais necessárias para a automação de testes de carga nas ferramentas. De forma simultânea foi realizado o levantamento dos trabalhos científicos que criassem geradores de carga ou que descrevessem o ambiente de um gerador. A seguir, são representados alguns trabalhos que serviram como referência bibliográfica que citaram geradores e suas características para casos de testes.

2.3.1 Estado da arte

O modelo SURGE proposto por (BARFORD; CROVELLA, 1998)[3] é um dos mais citados pois serve como referência para a geração de tráfego *web*. Ele é baseado em um autômato *ON-OFF* que realiza a captura das caracteristicas comportamentais de um sistema.

O trabalho realizado por (SILVA,2006)[19] apresenta o gerador de cargas de trabalho sintéticas W4Gen (*World Wide Web Workload Generator*), desenvolvido na linguagem Java, tem como objetivo reproduz cenários *web* reais atuais para avaliar e validar o desempenho de servidor *web*. Para isto, funções de distribuição de probabilidade analisados geram os modelos de carga. Com a interface gráfica amigável ele fornece um ambiente para usuários criarem cenários comuns utilizados na Internet. Este trabalho também apresenta a infra-estrutura da *web* como um todo e, como funciona a qualidade de serviço na Internet e realiza testes com o modelo de carga de trabalho. Possui como principais caracteristicas a geração de cargas sintéticas para servidores, cargas equivalentes a ambientes pré-definidos (por exemplo, *sites* acadêmicos), permite a criação de modelos de carga ou a seleção de modelos pré-definidos e a modificação de algumas características da carga de trabalho.

(FARAH; MURTA,2006)[6] mostra o gerador de sobrecarga Toró. O um gerador de modelo aberto, ou seja, não limita o numero de *threads*. Com isso, é capaz de produzir milhares de requisições por segundo, gerando sobrecarga, o que permite um estudo aprofundado sobre desempenho de servidores *web*.

Em seu trabalho (JING *et al.*, 2010)[11] determina os fatores mais importantes para a degradação de desempenho em sistemas. Apesar de o foco estar nisto, ele faz o uso da ferramenta JMeter[1] e com isso mostra um conjunto de informações utilizadas para a execução de testes de desempenho . Também descreve um conjunto de informações como, por exemplo, HTTP *request* e *Number of threads*.

O artigo de (HUSSAIN *et al.*, 2013)[9] realiza um comparativo entre as ferramentas, JMeter, soapUI e Storm. Analisou os tempo de resposta durante alguns horários do dia. Observou-se que JMeter levou mais tempo para responder aos serviços *web* em comparação as demais.

No trabalho realizado por (GUARIENTI *et al.*,2014)[8] é abordada a Análise do Tempo de Resposta para Teste de Desempenho em Aplicações *web* e possui como foco principal medir o tempo de execução das sequências de teste. Utiliza as ferramentas MBT que geram casos e scripts de teste de desempenho para o gerador de cargas LoadRunner [10] que mediu o tempo de execução de cada transação.

Tendo como base as informações adquiridas dos trabalhos estudados, foi possível realizar um apanhado a respeito da criação de casos de teste (bem como *scripts* e cenários). Sendo assim, um conjunto de características bem definidas de cada ferramenta foi estabelecido e verificou-se semelhança nos requisitos necessários modificando, as vezes, apenas os nomes dados. Dentre eles:

• *Record&Playback*: esta técnica realiza a gravação das interações executadas pelo usuário com a aplicação. É utilizada logo após ser configurado o parâmetro referente ao endereço IP da aplicação desejada;

• Rampa de Subida (*ramp up*): parâmetro que define a quantidade de usuários que irão iniciar o teste em um determinado tempo que deve ser definido no parâmetro Tempo de Rampa de Subida;

• Tempo de Duração do Teste: é tempo total de execução do teste;

• Tempo de Pensamento: tempo gasto pelo usuário para realizar uma atividade entre duas requisições como, por exemplo, uma pausa para a leitura de uma página;

• *URL* da Aplicação: refere-se ao endereço IP da aplicação para onde será gerada a carga;

• Quantidade de Usuários: número total de usuários simulados que farão as requisições.

2.4 Caracterização da carga web

Com a constante expansão da Internet, diversas pesquisas que visam identificar as características mais comuns dos usuários *web* vêm sendo realizadas, a fim de solucionar possíveis problemas e o melhorar o desempenho. Para isso, uma análise cuidadosa deve ser realizada para a implantação de servidores *web* sendo que seu desempenho está diretamente relacionado ao tipo de tráfego ao qual é submetido.

(CALZAROSSA;SERAZZI,1993)[5] apresentam orientações no processo de análise da carga de um sistema. A principal é a identificação do tipo da carga *web* que se deve analisar, visto que a Internet é um ambiente complexo e existem inúmeros parâmetros como, por exemplo, o tamanho dos arquivos a cada requisição.

(ARLITT; WILLIAMSON,1996) [2] são uns dos pioneiros nesta área, apresentando o conceito de invariantes, que são características do comportamento comum dos usuários que persistem nos dados analisados. Eles observaram que apesar do aumento do tráfego em dez anos [1994-2004] ter sido de 30 vezes, as características da carga não tiveram mudanças drásticas. Apesar de ter havido grandes mudanças na tecnologia como, por exemplo, novos protolocos e linguagens, observou-se a maioria das invariantes persistiram ao longo dos anos. As dez invariantes da carga *web* propostas por (ARLITT; WILLIAMSON,1996) [2] estão apresentadas na tabela 1.

A primeira invariante analisada na tabela afirma que a quantidade de sucesso das respostas das requisições feitas ao servidor era de aproximadamente 70% .

No Estudo realizado em 1996, (ARLITT; WILLIAMSON,1996) [2] observaram o tipo de requisições feitas aos servidores da Universidade de Waterloo, Departamento de Ciência da Computação da Universidade de Calgary e Universidade de Saskatchewan. A tabela 2 mostra o percentual de cada tipo de documento.

Quanto ao tipo os arquivos foram classificados como: HTML, Imagens, Dinâmicos, Áudio, Vídeo e Outros. A segunda tabela mostra um número alto de arquivos no formato Imagem e HTML representando mais de 90% das requisições, pois páginas com conteúdo estático eram amplamente utilizadas. Neste contexto, a invariante tamanho da transferência se encaixa, pois mostra que a grande maioria dos arquivos transferidos eram pequenos, abaixo de 20%, já que ,antigamente, não havia grande disponibilidade de arquivos de vídeo.

(WILLIAMS *et al.*,2005) [21] repetiu este trabalho em 2005 e constatou que as invariantes estabelecidas em 1996 ainda eram validas apesar das mudanças nas tecnologias como protocolos e linguagens script.

Com a presença cada vez maior de sites de *e-commerce* (WEBER;HARIKARAN,2003) [20] fizeram a analise dos logs de uma determinada instituição bancária. Observaram que

Tabela 1 – Resumo das Invariantes. (Fonte: ARLITT & WILLIAMSON (1996))

Invariante	**Descrição**
1- Taxa de sucesso	Taxa de sucesso das requisições ao servidor Web foi de, aproximadamente, 70%
2- Tipos de arquivo	Documentos HTML e de imagem juntos representam 85% das requisições
3- Tamanho da transferência	O tamanho médio da transferência é pequeno, apenas 20KB
4- Requisições distintas	Apenas 1% das requisições ao servidor é de documentos distintos
5- Referencias únicas	Aproximadamente um quarto dos arquivos e *bytes* são acessados uma única vez
6- Distribuição dos tamanhos	A distribuição do tamanho dos arquivos segue o modelo de Pareto com média 0.4
7- Distribuição dos tamanhos	10% dos arquivos acessados correspondem a aproximadamente a 80-90% das requisições ao servidor e a 80-90% dos *bytes* transferidos
8- Tempo entre referências	Os tempos entre as solicitações sucessivas para o mesmo arquivo são exponencialmente distribuídos e independentes
9- Requisições remotas	Sites remotos representam mais de 70 % dos acessos ao servidor e mais de 80% dos *bytes* transferidos
10- Uso na WAN	10% dos domínios dão origem a mais de 75% dos acessos

Tabela 2 – Resumo tipos de arquivos. (Fonte: ARLITT & WILLIAMSON (1996))

	Waterloo	**Calgary**	**Saskatchewan**
Item	Req(%) Bytes (%)	Req(%) Bytes (%)	Req(%) Bytes(%)
Imagem	63.02 10.77	78.76 33.36	57.64 33.35
HTML	23.18 6.02	8.09 1.13	12.46 11.98
Dinâmicos	1.96 0.09	3.63 0.55	5.78 8.46
Áudio	00 00	0.01 0.16	0.01 0.29
Vídeo	00 00	0.40 54.02	0.06 5.25
Diretório	4.67 0.19	3.12 0.65	13.35 19.37
CSS	0.93 0.03	2.48 0.07	6.54 0.84
Formatado	5.13 82.32	1.02 8.30	1.30 17.25
Outros	1.11 0.58	2.49 1.76	2.86 2.81

30% das requisições são de objetos dinâmicos o que demonstra com a criação deste tipo de serviço os tipos de arquivo mais acessados vêm mudando. Ainda neste cenário, (WANG *et al.*, 2003) [18] também caracterizaram carga de trabalho de *e-commerce*. Foram coletados dados de dois servidores e concluiu-se que objetos imagem são os mais acessados com

88,11% como é observado na tabela 3.

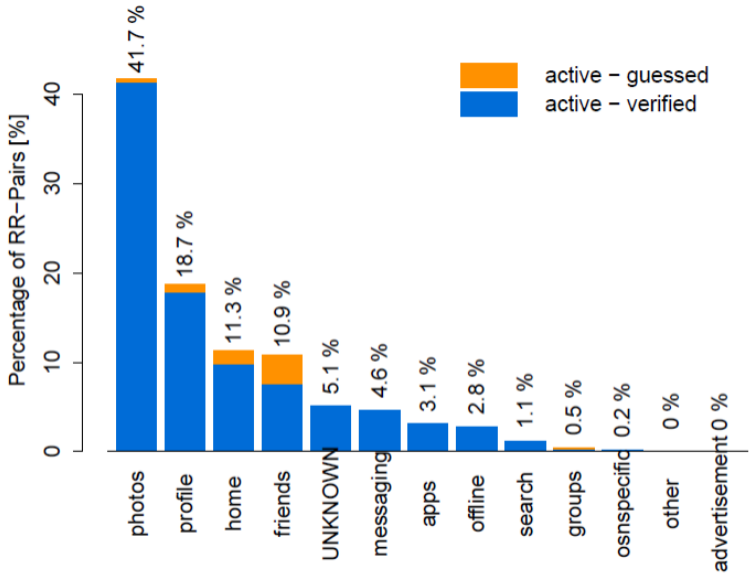
Tabela 3 – Percentual tipo de arquivos (Fonte: (WANG *et al.*, 2003))

Objetos	**Requisições (%)**	**Bytes Transferidos (%)**
Imagem	88,11	46,36
Asp	7,11	37,19
HTML	1,14	3,32
Js	1,00	11,16
Css	2,08	1,61
Outros	0,56	0,36

No estudo realizado por (MENASCé; ALMEIDA, 2003)[12] é possível observar que tem ocorrido uma diminuição ao acesso de conteúdo estático ao longo dos anos. Isso se deve a mudança dos conteúdos das páginas neste período. O surgimento da Web 2.0 foi responsável pela forma como os usuários participam da Internet. Agora eles são capazes de publicar seu próprio conteúdo ao invés de apenas consumir conteúdo publicado por um único administrador.

Atualmente, com o surgimento das redes sociais, vídeos sob demanda e outras tecnologias a caracterização da carga Web tem sofrido mudanças bruscas. Neste contexto (SCHNEIDER *et al.*,2009)[16] monitoraram o tráfego de rede com milhares de usuários utilizando as redes sociais Facebook, HI5, LinkedIn e StudiVZ. Sendo assim, caracterizaram a navegação a partir da identificação minusciosa do comportamento e ações dos usuários. Foi possivel identificar que o conteúdo fotos é o mais acessado nas redes sociais como mostra o gráfico 2 abaixo que ilustra este comportamento no Hi5 :

Figura 2 – Comportamento usuários do Hi5 (SCHNEIDER *et al.*,2009))



Um dos maiores tráfegos, dentre os sistemas sociais atuais, é a distribuição de vídeo. Com isso, no trabalho desenvolvido por (GILL *et al.*, 2007)[7], eles caracterizaram

as sessões dos usuários do Youtube durantes três meses observaram , aproximadamente 25 milhões de transaçãoes entre usuários e mais de 600.000 de downloads de vídeos como mostra a tabela 4.

Tabela 4 – Resumo dos dados Youtube (Fonte: GILL *et al.*, 2007)

Item	**Information**
Start Date	14/01/07
End Date	08/04/07
Total Valid Transactions	23,250,438
Total Bytes	6.54 TB
Total Video Requests	625,593
Total Video Bytes	6.45 TB
Unique Video Requests	323,677
Unique Video Bytes	3.26 TB

Assim, foi possível analisar os padrões de uso, popularidade e comportamento da taxa de transferência e compararam com a carga de trabalho tradicional. Também compararam os resultados com as sessões tradicionais dos usuários da *web*. Desta forma, foi possível identificar que os usuários do Youtube transferem mais dados e tem mais tempo de espera do servidor do que os usuários *web* tradicionais.

Com base nestes estudos propusemos gerar cargas voltadas para o cenário atual. Para isso, separamos em três classes:

• Pequena: cargas de conteúdo estático como, por exemplo, sites simples;

• Média: consideramos como média uma carga de até 10MB, ou seja, fotos e arquivos;

• Grande: é representado por conteúdo com vídeo, como o Youtube, DailyMotion;

Diante desta classificação serão criados os cenários de testes que estarão presentes no próximo capítulo.

3 CICLO DE TRABALHO DOS GERADORES ATUAIS

Este capítulo apresenta, de forma detalhada, uma descrição das ferramentas escolhidas para a realização dos testes de carga. Para a efetivação do trabalho adotou-se a classificação de (MOLINARI,2003) [14] para as informações contidas no conjunto de características implementado como foi dito no capítulo 2. O ciclo de trabalho básico é genérico, ou seja, todas as ferramentas utilizam o mesmo processo. Porém, podem se diferencias pela organização e nome dos componentes . A afim de exemplificar selecionamos duas ferramentas de automatização de testes. A escolha foi feita a partir de um estudo, com base em diversos trabalhos citados no capitulo 2, no qual demonstra que as ferramentas LoadRunner[10] , a mais utilizada do mercado para testes, e JMeter[1], *opensource* com qualidades semelhantes ao LoadRunner[10] muito utilizada por *testers*, são as melhores disponíveis atualmente.

3.1 JMeter

Diante da dificuldade de se obter cargas reais Stefanno Mazzochi, membro da Apache Software Foundation, desenvolveu o JMeter[1] afim de gerar cargas de trabalho sinteticas .

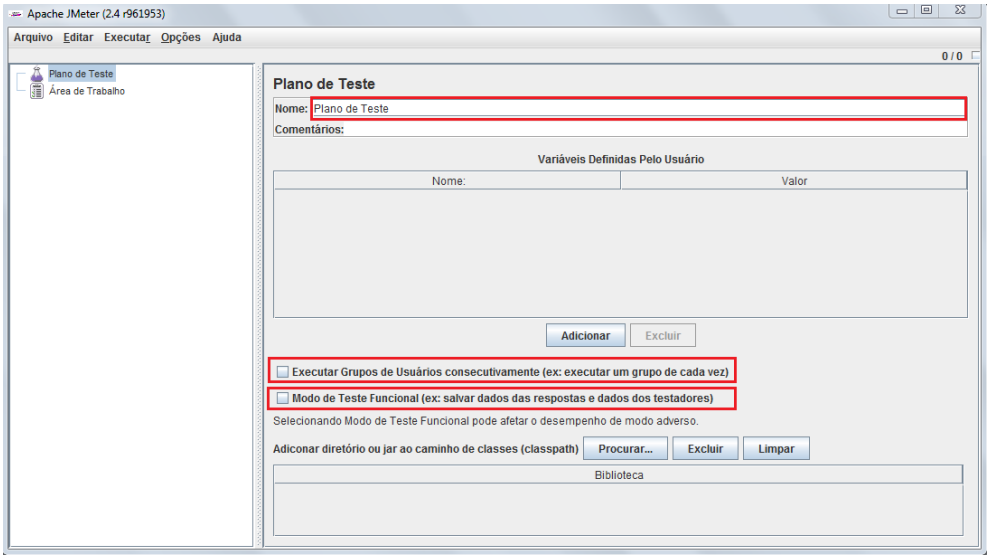
Atualmente, o Apache Jmeter[1] é uma ferramenta *opensource*, escrita totalmente em Java para realizar testes de desempenho e estresse em aplicações cliente/servidor e *web* a fim de testar sua robustez . Possui como principais características a simulação de usuarios simultâneos através de *threads* e o suporte aos protocolos HTTP FTP, SOAP entre outros .

3.1.1 **Ciclo básico de trabalho**

A ferramenta oferece uma interface amigável para o usuário, distribuída de forma hierarquica e possui como componente básico para a criação de *script* o Plano de Teste (*TestPlan*) e é nele que serão adicionados os demais componentes significativos para a execução do teste. Nele é possível definir um nome, adicionar comentários e criar variáveis que serão visíveis por todas as requisições do Plano de Teste definido. Também é possível selecionar as caixas de seleção Executar Grupos de Usuários consecutivamente ,com o qual pode-se definir se o grupo de usuários serão executados de forma consecutiva, e Modo de Teste Funcional em que é feita a gravação dos dados de cada requisição como mostra a figura 3 abaixo:

É importante dizer que para a geração e execução de casos de teste no JMeter[1] é preciso antes configurar uma série de informações Um Plano de Teste simples inclui os principais elementos utilizados para a criação de casos de testes :

Figura 3 – Configuração do Plano de Teste



• Grupo de Usuários (*Thread Group*): Utilizado para controlar o número de usuários (*threads*) virtuais simultâneos em um período de tempo. É aqui que a carga do teste é definida;

• Testador (*Samplers*) que representa cada requisição para um servidor quando o plano de teste é executado. Pode ser do tipo HTTP, FTP, SOAP entre outros;

• Ouvintes (*Listeners*): Capturam os resultados dos dados dos processos gerados pelo plano de teste;

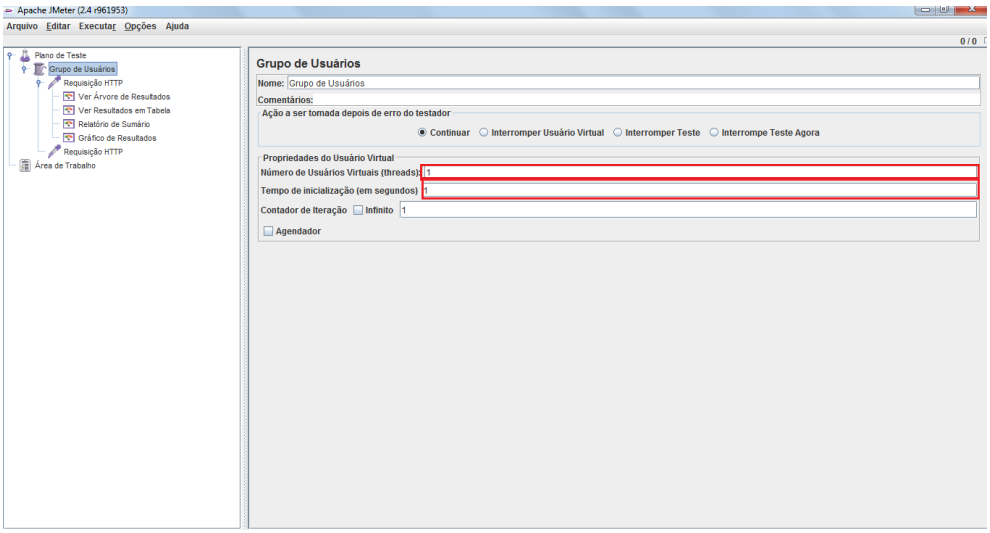
• Asserções (*Assertions*): É possível adicionar pontos para verificar se a resposta obtida está de acordo com uma determinada afirmação feita no elemento *Sampler*;

O nó Área de trabalho (*WorkBench*) fornece um local para armazenar temporariamente os elementos de um teste. Já é visto na inicialização do programa.

Após a definição do Plano de Teste é necessário configurar o Grupo de Usuários, que pode ser mais de um. Dentro do Grupo de Usuários é possível controlar o número de usuários virtuais (*threads*), o tempo de inicialização de cada *thread* e também o contador de iteração que define quantas vezes é teste será executado como mostra a figura 4 a seguir:

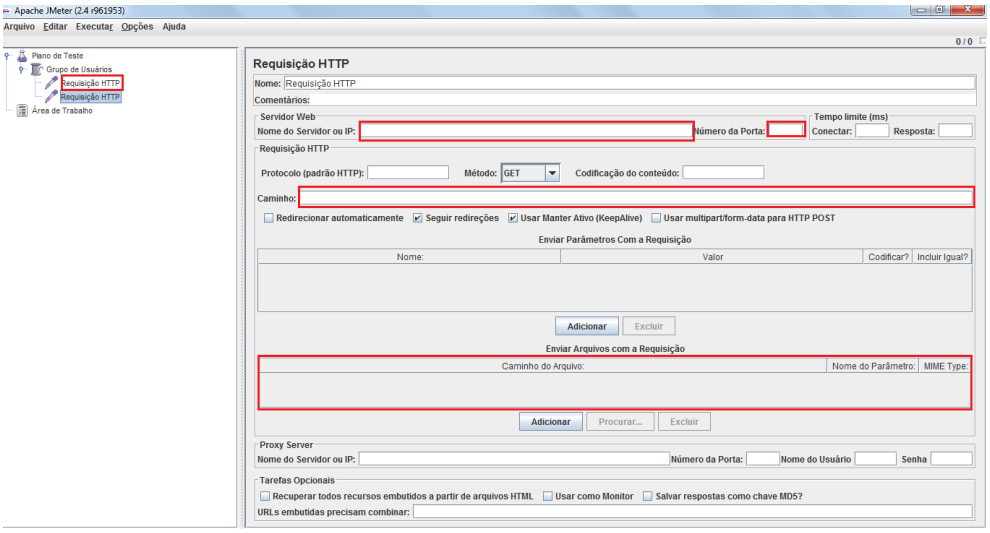
Os demais parâmetros serão adicionados utilizando o menu Requisição HTTP (HTTP *Request*) a partir do controlador de requisições de serviços Testador (*Sampler*) pois, o objetivo é realizar teste de desempenho em aplicações *web*. Este componente é responsável por gerenciar as requisições HTTP/HTTPS em uma página *web*. Possui um campo chamado Nome do Servidor ou IP onde informamos a URL ou IP da página alvo de testes. No campo Numero da Porta deve-se indicar qual porta de comunicação para a página solicitada. Destacamos também os campos Caminho que define o caminho da página e Enviar Arquivos com a requisição no qual adicionamos os parâmetros desejados

Figura 4 – Criação do Plano de Teste



para serem enviados junto com a requisição. A figura 5 mostra essa situação.

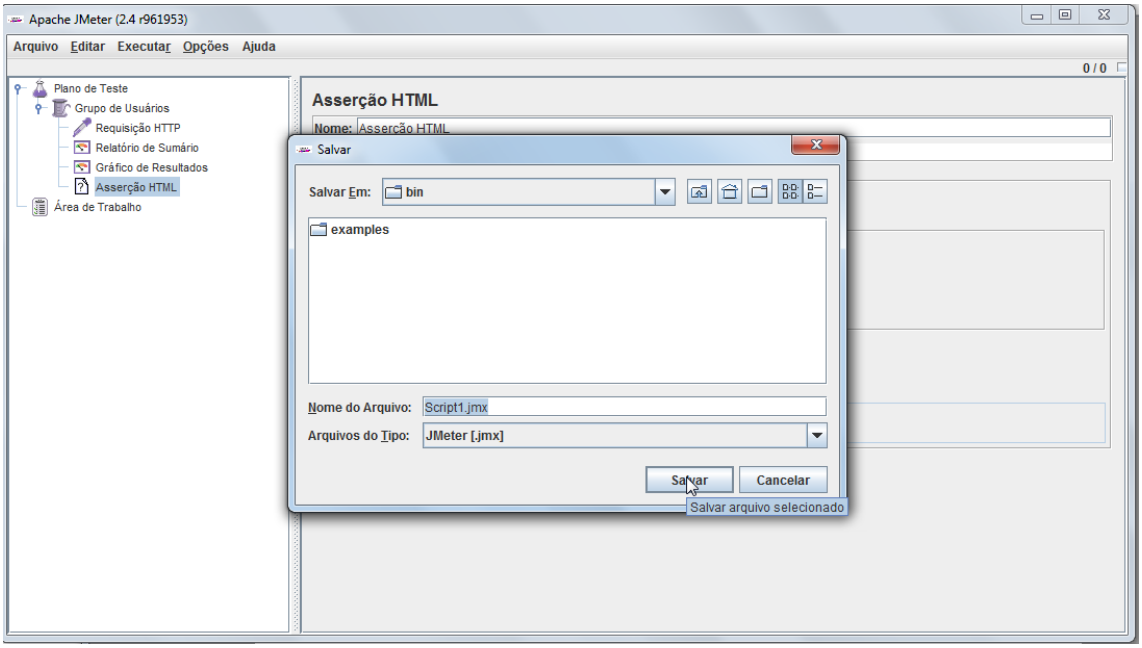
Figura 5 – Configurando Requisição HTTP



Depois de preencher estes campos é necessário inserir componentes para a coleta e avaliação dos dados obtidos. Para isso, o elemento Ouvinte (*Listener*) é utilizado pois é responsável por monitorar, coletar e apresentar os dados capturado em diversos formatos para analisar em tempo de execução ou posteriormente. Caso seja necessário, pode-se incluir o elemento Ver árvore de Resultados (*View Results Tree*) que mostra uma árvore com as respostas de todos os testadores Gráfico de Resultados (*Graph Results*) que gera um gráfico com os tempos das requisições realizadas por segundo.

Para incluir alguns testes de validação sobre a resposta da solicitação feita no testador utilizamos o Asserções (*Assertions*) . Permite também visualizar quais falharam e o porquê da falha. Posteriormente salvamos este *script* para utiliza-lo na execução dos testes.

Figura 6 – Salvando um *script* no JMeter



Logo após a criação do *script* executaremos os testes. Aplicaremos cargas de tamanho pequeno, médio e grande .

3.2 HP LoadRunner

Criando por Mercury International e posteriormente vendido a HP o LoadRunner [10] é uma ferramenta de teste de *software* proprietário, que realiza testes de desempenho e estresse em aplicações *web*. Responsável por simular múltiplos usuários simultaneamente, faz gravação e posteriormente a análise do desempenho da aplicação.

3.2.1 **Ciclo básico de trabalho**

Com uma interface completa e amigável LoadRunner possui três componentes individuais:

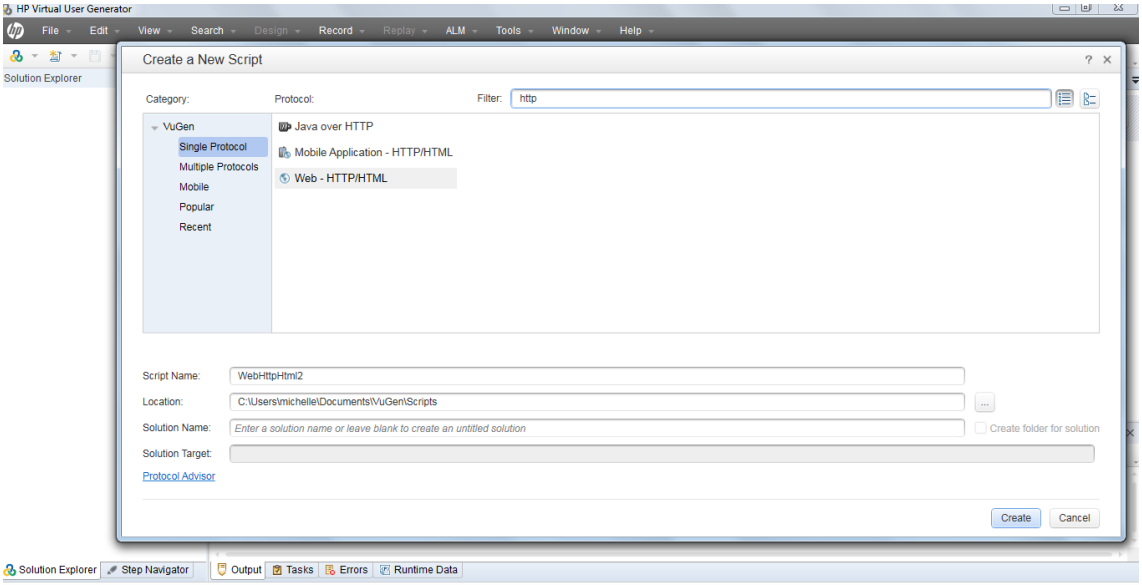
• *Virtual User Generator* ou VuGen é o gerador de usuários virtuais chamados de Vusers, que são responsáveis capturar as informações referentes às interações do usuário com a aplicação a ser testada. Posteriormente, estas informações são salvas em um *script* de teste;

• O *Controller* que é responsável por gerenciar, controlar e executar todos os testes criados nos cenários;

• *Analisys* é onde se analisa os resultados e faz comparativos de testes de carga.

Um teste de carga simples no LoadRunner [10] consiste em cinco fases correspondentes as criadas por (MOLINARI,2003):

Figura 7 – Composição de um *script* no LoadRunner



• Planejamento : É a criação do plano de teste de carga. Nesta fase se defina os seus requisitos de teste de desempenho como, por exemplo, número de usuários simultâneos e tempos de resposta necessários;

• Criação do Roteiro: Nesta fase se criar os *scripts* do Vuser. É necessário utilizar o VuGen para capturar as atividades do usuário final em *scripts* automatizados;

• Definição do Cenário. Utilizar o *Controller* para configurar o ambiente de teste de carga;

• Execução do cenário: Faz o uso do *Controller* para dirigir, gerenciar e monitorar o teste de carga;

• Análise dos resultados. Utilizar o componente *Analysis* para a criação de gráficos e relatório para avaliar o desempenho do sistema.

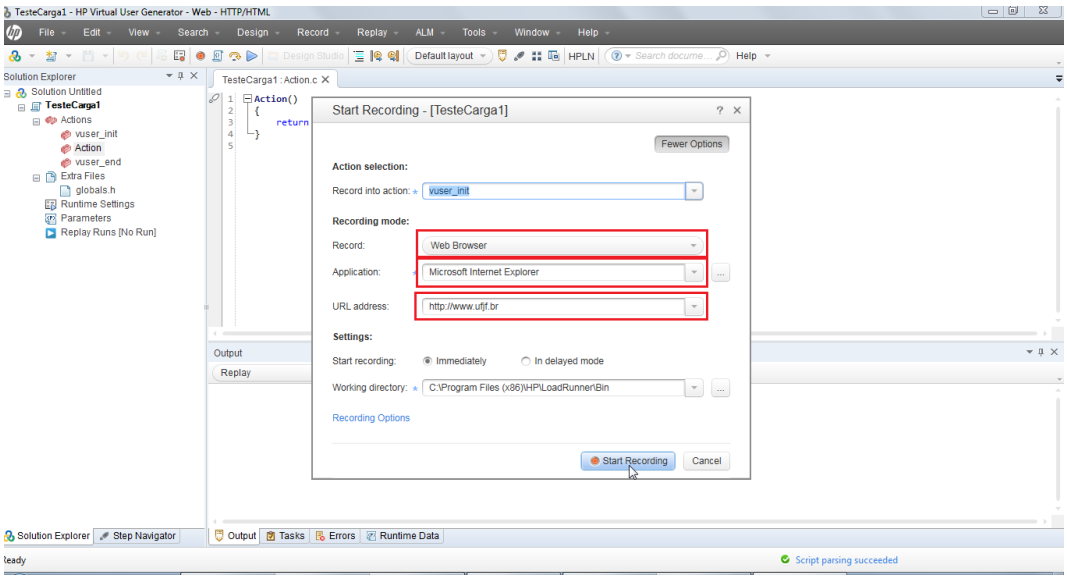
Em um ambiente de testes de desempenho, LoadRunner substitui usuários humanos por usuários virtuais, também conhecidos como Vusers. Para cria-los é necessário utilizar o componente VuGen .

Para desenvolver um *script* Vuser, primeiro é necessario abrir o VuGen e criar um *script* em branco. Depois disso, é possível registrar eventos e adicionar melhorias manuais para o *script*. Neste caso utilizamos um Vuser baseado em Web HTTP/protocolo HTML pois o objetivo é criar cargas *web* como mostra a figura 7 .

O próximo passo no desenvolvimento do *script* Vuser é registrar as ações executadas por um usuário real. Para isso é preciso na barra de ferramentas no item *Record- Record*. Irá abrir um *pop-up* com as seguintes opções como mostra a figura 8:

• Em *Record into action box*, selecionar *Action*. No item *Record*, selecionar *Web*

Figura 8 – Pop-up Record



Browser e em *Application* é possível escolher o nevegador que rodará a pagina a ser testada. No item URL é preciso digitar o endereço da página a ser testada. Ao clicar em *Start Recording* o navegador escolhido abre automaticamente .Enquanto a navegação ocorre normalmente a gravação dos dados é realizada simultaneamente. Ao clicar em *Stop* a gravação é interrompida e pode-se observar no VuGen o passo-a-passo realizado na página *web* alvo.

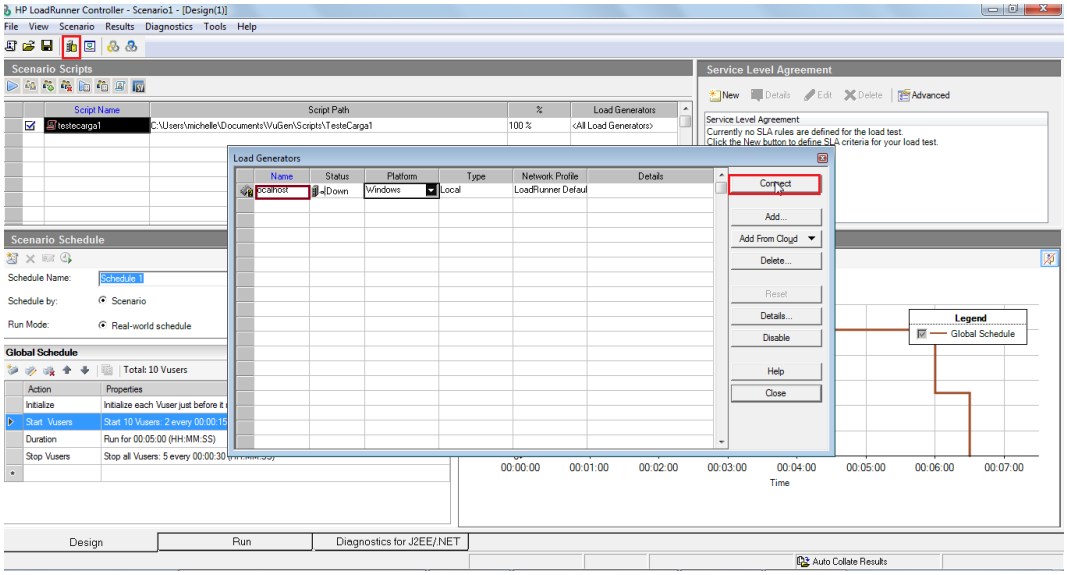
Para realizar o teste é preciso abrir o componente *Controller* e carregar o cenário gerado na etapa anterior . É possível definir no componente *Controller* alguns parâmetros, tais como: quantidade de *threads* que simulam o usuário acessando a aplicação e tempo total de execução de teste. Para finalizar a configuração do cenário de teste, é necessária a adição de alguns contadores como, por exemplo, *Trans Response Time* (tempo de resposta), *Hits per second* (número de requisições por segundo) e *Throughput* (vazão). Os gráficos são gerados em tempo de execução.

Os dados relativos a cada tipo de contador configurado para o teste podem ser visualizados em tempo de execução por gráficos e tabelas .Com o *Load Generator* aberto e preenchido o *localhost* a conexão com o servidor local já pode ser realizada como é mostrado na figura 9 .

Na aba *Run* pode-se escolher a quantidade de usuarios simulados. Assim, conclui-se os testes básicos, mas é possível complementar ainda mais utilizando o componente *Analisys* que é responsável por nos prover mais informações sobreo a execução dos cenários com gráficos mais apurados. Ele também correlaciona diferentes gráficos, além de comparar execuções já realizadas. Gera um arquivo HTML com um resumo dos gráficos, quantidade de erros, *throughput*, horário do teste, duração entre outros dados.

Com os dados pré-definidos os cenários foram criados e executados em ambas as

Figura 9 – Load Generator



ferramentas. Os resultados dos testes automatizados foram armanezados e serão mostrados no próximo capítulo alguns gráficos importantes utilizados para a avaliação de testes de desempenho. Um comparativo entre as ferramentas sobre o processo de criação e execução de scripts também será realizado.

4 PRÁTICA

4.1 Definição dos Casos de Testes

Com o intuito de verificar os aspectos funcionais do conjunto de características, foram definidos alguns casos de teste para serem executados pelas duas ferramentas por ela geradas. Neste contexto, foram definidos casos de teste de desempenho idênticos para serem executados pelas ferramentas JMeter e LoadRunner afim de ser obter um comparativo do comportamento de ambas. Os casos de teste foram divididos de acordo com o tamanho das cargas como mostra a seguir:

• Pequena: representam sites simples. No caso, foi utilizado como exemplo a navegação no site da Universidade Federal de Juiz de Fora (UFJF).

• Média: cargas de até 10MB. O site 1000imagens que representa uma comunidade fotográfica foi utilizado com cenário.

• Grande: carga acima de 10 MB. Como cenário foi escolhido o DailyMotion, site que disponibiliza vídeos, para gerar o caso de teste.

A escolha dos tamanhos das cargas se deu pelo fato, da necessidade em se avaliar o comportamento dos servidores ao receberem diferentes tamanhos de carga. Cada cenário criado possui um conjunto definido de informações necessárias para o teste, tais como:

• Host da aplicação : *localhost*

• Número da Porta : 9090

• *Browser* de navegação: Mozilla Firefox versão 37.0.2

• Tempo de rampa de subida : 00:01:00

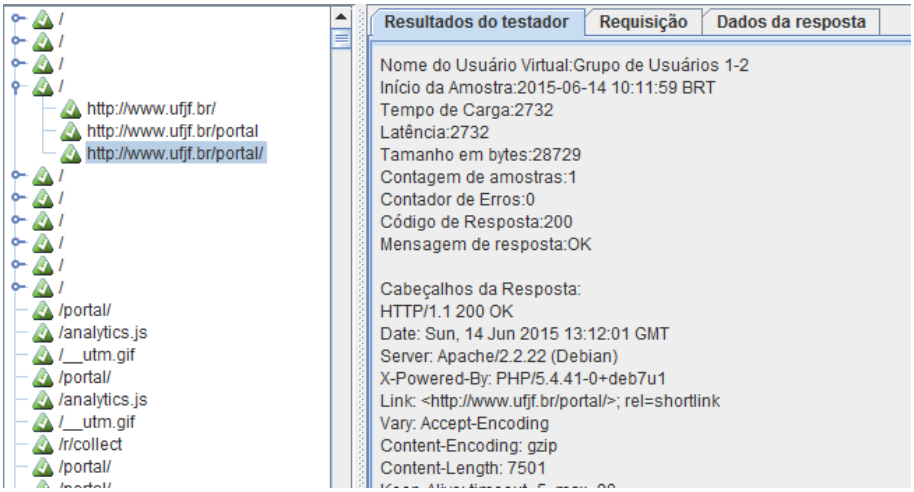
• Quantidade de usuários virtuais : 10

Após a definição do tamanho das cargas e do conjunto de informações comuns os testes foram realizados nas respectivas ferramentas.

4.2 Automação de testes

Definidos os cenários na ferramenta JMeter, para visualizar os resultados dos testes é necessário adicionar os Ouvintes (*Listeners*) que geram os resultados nos formatos: tabela, gráficos, árvore e arquivos de log. Eles podem sem salvos nos formatos XML ou CSV. Para exibir gráficos mais precisos é necessário adicionar *Plugins*. No Ouvinte Ver Árvore de Resultados, o resultado do teste é exibido para cada amostra como mostra a figura 10 que apresenta um trecho do cabeçalho dos dados da página inicial da UFJF como latência, tamanho e código de resposta.

Figura 10 – Árvore de Resultados UFJF



A seguir, nas figuras 11, 12 e 13 são exibidos o *throughput*, que representa o número real de requisições feitas ao servidor, e a média do tempo de resposta.

Figura 11 – Gráfico Agregado UFJF

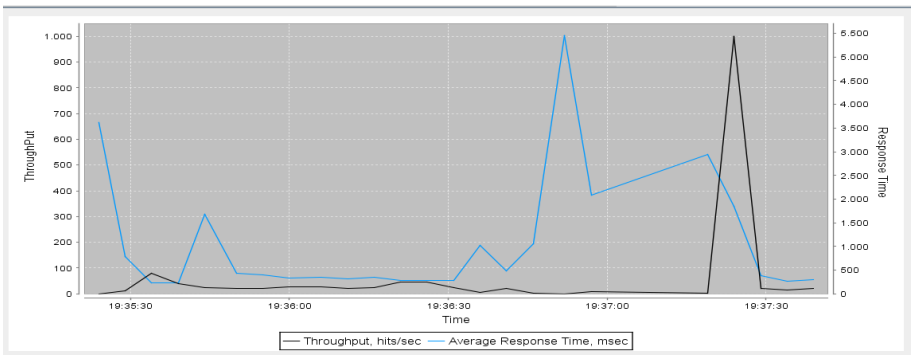
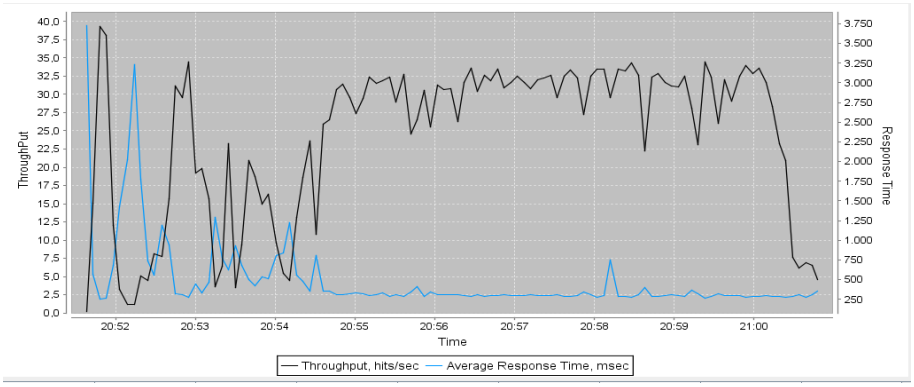
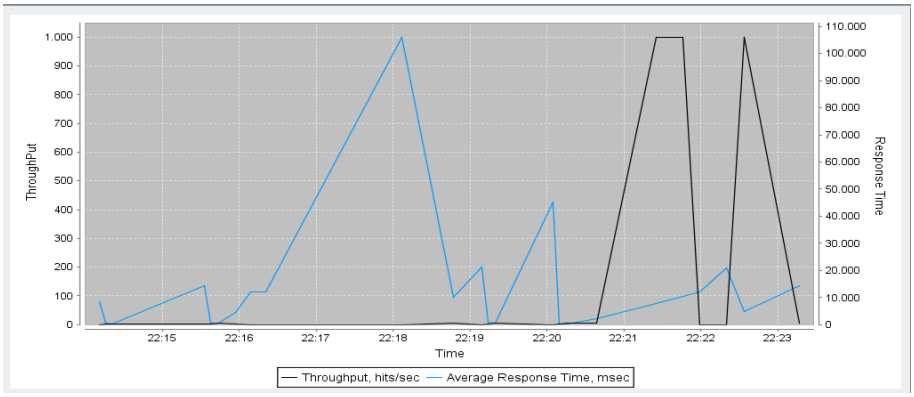


Figura 12 – Gráfico Agregado 1000Imagens



É possível observar a oscilação do *throughput* no decorrer do tempo, e o tempo de resposta se mantém, a maior parte do tempo, baixo com cargas pequenas como mostra a figura 11. O pico sofrido pode ser explicado como algum link externo dentro do site da UFJF. No segundo Gráfico Agregado, com cargas de tamanho médio, o *throughput* e tempo de resposta se mantém baixo mas as oscilações geradas são mais frequentes, apesar

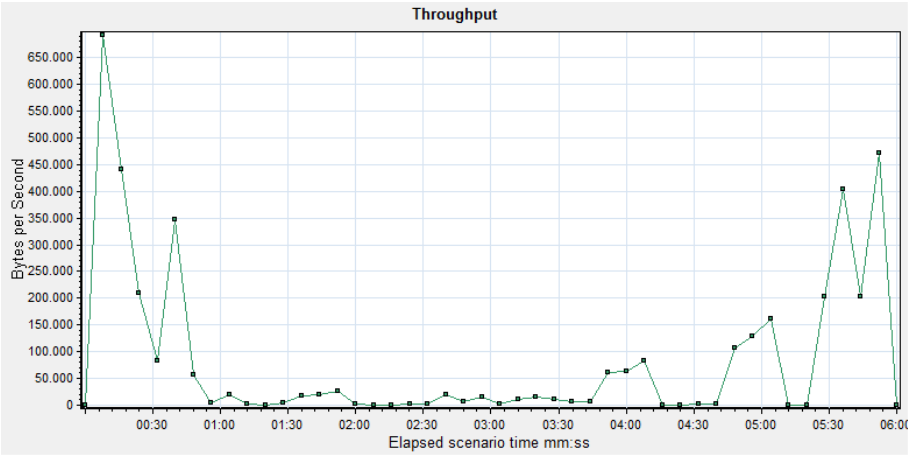
Figura 13 – Gráfico Agregado Daily Motion



do que se espera com cargas maiores que é o tempo de resposta aumentar. No terceiro caso, o *throughput* e o tempo de resposta se mantém altos e quando um vídeo em alta resolução(1080p) é carregado o o tempo de resposta sofre picos como é observado a partir do minuto 22:21.

A ferramenta LoadRunner disponibiliza diversos gráficos para a visualização dos resultados das amostras. Os gráficos dos requisitos de desempenho para a avaliação são plotados de forma separada no componente *Controller*. As figuras 14, 15 e 16 apresentam, como exemplo, o *throughput* dos três casos de testes, respectivamente:

Figura 14 – *Throughput* UFJF



Observou-se que em todos os casos o *throughput* sofre oscilações, isto ocorre devido ao tempo de pensamento. No primeiro cenário, de aplicações leves, mostra que a quantidade de carga gerada pelo Vusers é inferior aos cenário médio e grande, que apresentam um aumento desta quantidade, pois geram cargas maiores. Também geramos os gráficos que representam o número de conexões abertas com o servidor em cada cenário, como mostra as imagens 17, 18 e 19 a seguir:

Diante disto, podemos observar que no primeiro cenário o número de conexões abertas foi o menor chegando ao máximo de 150. No segundo cenário, sofreu um aumento

Figura 15 – *Throughput* 1000Imagens

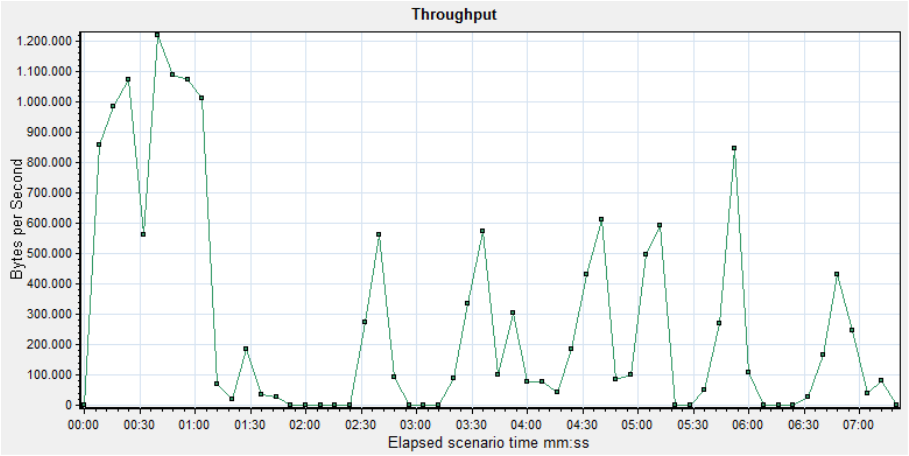
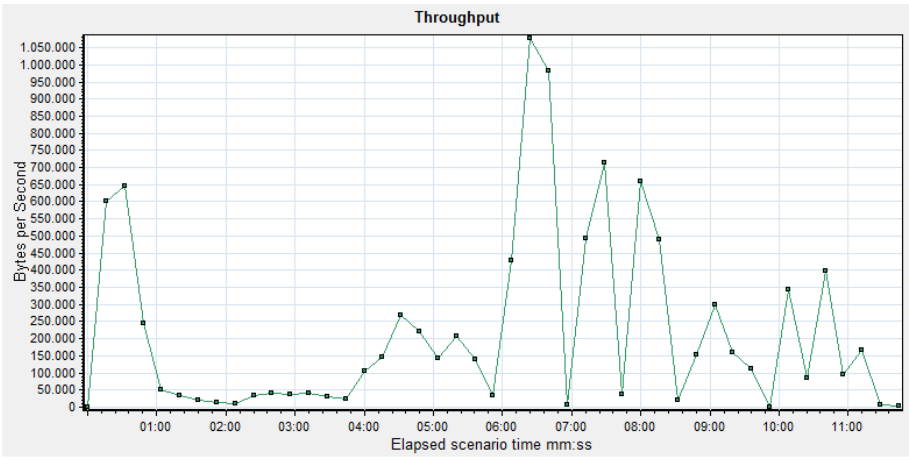


Figura 16 – *Throughput* DailyMotion



chegando a 200 conexões no máximo. O terceiro cenário representou o maior numéro de conexões abertas onde a máxima foi de 290.

Através do Analysis é possível gerar um relatório de sumário com a quantidade de Vusers, *throughput*, média do *throughput*, total de *hits* (número de requisições), média de *hits* e, caso haja, o número de erros. As imagens 20, 21 e 22 a seguir ilustram os relatórios gerados dos testes de cada caso que mostram um resumo dos respectivos dados: máximo de VUseres rodando, total de *throughput*, média de *throughput*, total de *hits*(número de requisições) e a média de requisições por segundo. Através deste relatório também foi possivel observar que o total de *throughput* foi maior na aplicação com carga pesada.

Em relação aos testes automatizados foi possivel observar que os gráficos de tem comportamento semelhante em todos os casos. *Throughput* e tempo de resposta no primeiro e segundo cenários se manteve baixo aumenta. Tendo apresentado as etapas do processo de criação do conjunto básico para testes de carga com as ferramentas JMeter e LoadRunner e a análise dos dados gerados, a seguir faremos um comparativo dos geradores.

Figura 17 – Conexões cenário UFJF

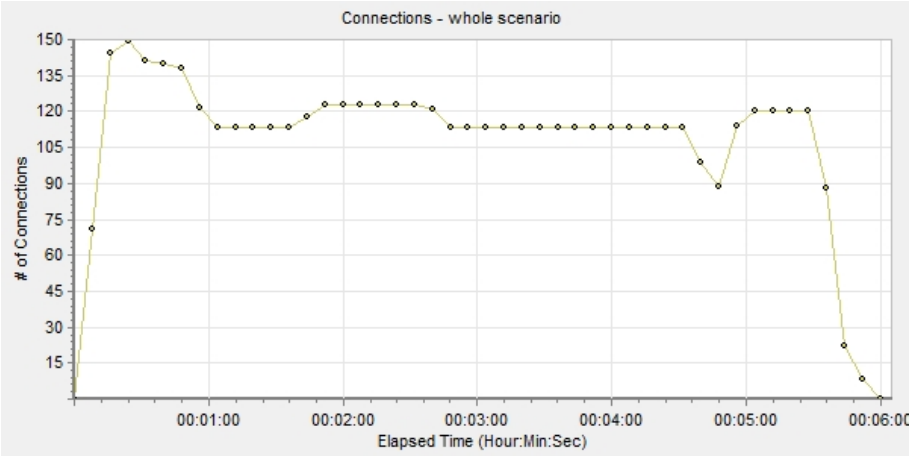
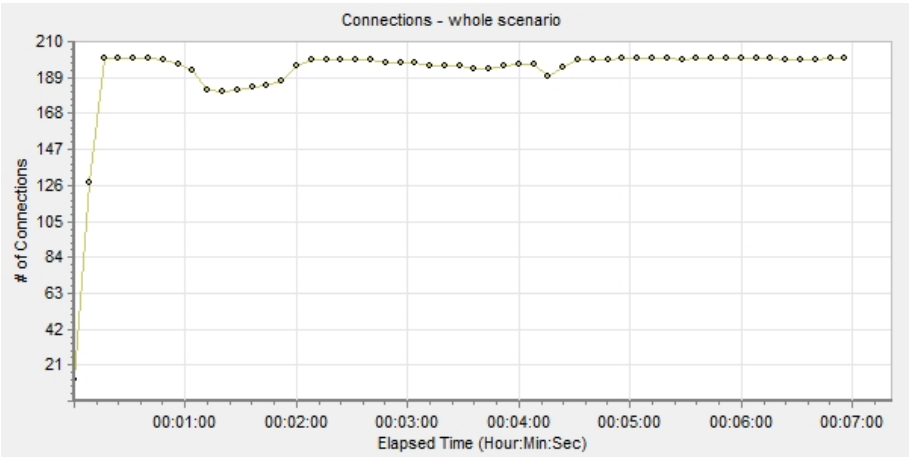


Figura 18 – Conexões cenário 1000Imagens



4.3 Comparativo entre os Geradores de Carga analisados

Primeiramente, com base na experiência obtida com a utilização das ferramentas JMeter e LoadRunner foi possivel estimar aproximadamente o tempo gasto no processo de criação do script e execução dos testes em cada uma. Pelo fato da ferramenta LoadRunner trabalhar com componentes individuais e sendo mais complexa, ocorre uma pequena diferença no tempo gasto para a criação e execução dos testes.

A etapa de geração dos scripts de teste é muito semelhante. Ambas possuem métodos para capturar as interações do usuário com a aplicação. Entretanto, elas suportam diferentes linguagens para gravação dos scripts de teste, LoadRunner permite a gravação nas linguagens C, Java ou Visual Basic. Já o JMeter grava todas as informações relativas às interações em um arquivo no formato JMX .

A ferramenta JMeter disponibiliza realizar todos os comandos em apenas uma tela, ou seja, não possui um módulo potente de análise de resultados enquanto o LoadRunner trabalha com componentes individuais(*VuGen*, *Controller* e *Analysis*) mas que se complementam. Em relação ao número de protocolos que cada uma suporta para a gravação dos

Figura 19 – Conexões cenário DailyMotion

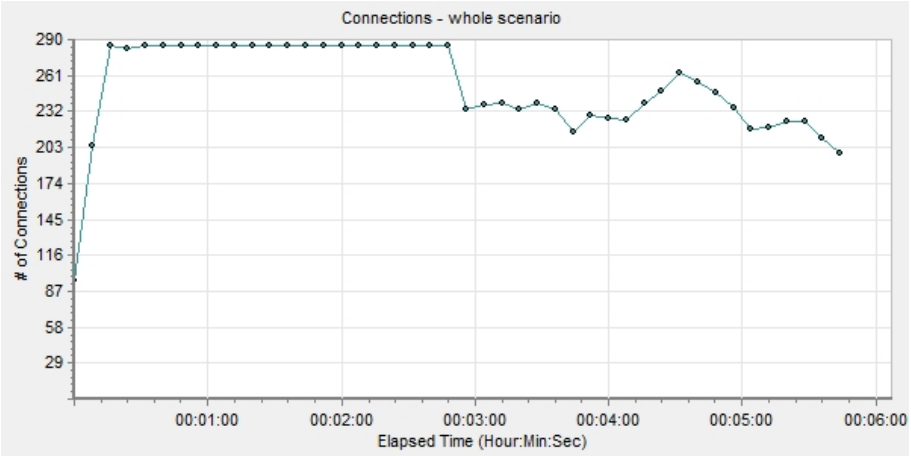


Figura 20 – Relatório de Sumário UFJF

Statistics Summary

Maximum Running Vusers:	20	
Total Throughput (bytes):	31.286.624	
Average Throughput (bytes/second):	85.483	
Total Hits:	4.291	
Average Hits per Second:	11,724	View HTTP Responses Summary
Total Errors:	1	

Figura 21 – Relatório de Sumário 1000Imagens

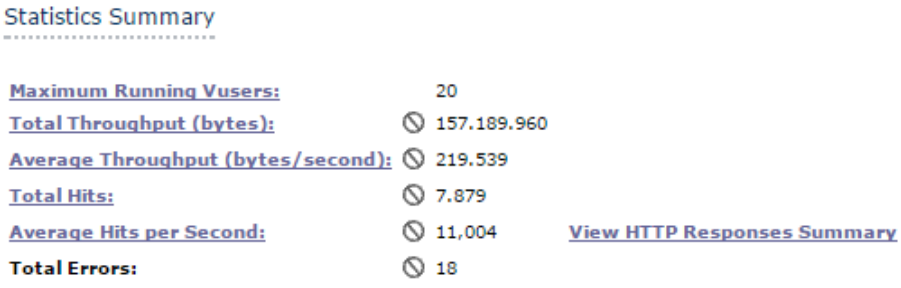
Statistics Summary

Maximum Running Vusers:	10	
Total Throughput (bytes):	123.977.266	
Average Throughput (bytes/second):	281.767	
Total Hits:	4.679	
Average Hits per Second:	10,634	View HTTP Responses Summary

scripts de teste o LoadRunner se difere por possuir suporte a uma maior quantidade de protocolos (HTTP/HTML, Flex, Oracle, Java sobre HTTP, entre outros).

Somente o LoadRunner possui a funcionalidade Perfil de Carga de Trabalho, que representa o perfil de testes que será executado. Essa informação é configurada em *Start VUsers* e com ela é possível determinar um perfil que todos os usuários irão iniciar o teste de forma gradativa ou simultaneamente. JMeter não permite definir um Perfil de Carga de Trabalho, mas possibilita especificar rampa de subida (*ramp up*) que define a quantidade de usuários virtuais que iniciarão o teste em um intervalo de tempo pré-definido.

Figura 22 – Relatório de Sumário DailyMotion



Em relação as plataformas ambas possuem suporte ao Windows e Linux. Em ambas também é possível definir contadores para medir informações de desempenho do ambiente de testes(*hardware* a fim de encerrar a configuração do cenário, tais como: percentual de utilização de CPU, do disco, quantidade de memória disponível e etc.

É possivel observar que JMeter é um *software* muito leve (cerca de 20 MB) em comparação com LoadRunner (cerca de 3,7 GB). Jmeter é uma ferramenta muito simples de trabalhar, enquanto LoadRunner é bastante complexa. Por isso, LoadRunner disponibiliza possui uma maior gama de recursos a seres utilizados para realizar testes de carga de forma mais completa. Para melhor visualização, a tabela 5 a seguir ilustra o comparativo.

Tabela 5 – Comparativo das ferramentas para teste de desempenho

Características	**JMeter**	**LoadRunner**
Opensource	X	
Relatórios em Gráfico	X	X
Usuários Virtuais Ilimitados	X	X
Gravação Automática de Testes	X	X
Record & Playback	X	X
Perfil da Carga de Trabalho		X
Tempo de Rampa de Subida	X	X
Componentes para geração e execução de *scripts*		X

5 CONCLUSÕES

5.1 Considerações Finais

Grande parte do crescimento recente da Internet está relacionado ao aumento do número de serviços oferecidos aos usuários da WWW. Com isso, a capacidade das aplicações de responderem as requisições rapidamente é considerada uma medida de sucesso que garante a qualidade do serviço ofertado. Quando um grande número de usuários gera sobrecarga nos servidores o serviço não deve ser afetado, ou seja, o tempo de resposta deve se manter suficientemente rápido mesmo com o numero crescente de solicitações.

Com base nesta ideia, este trabalho apresentou a descrição de um conjunto de características para a geração e execução de casos de teste concretizados nas ferramentas de teste de desempenho JMeter e LoadRunner. Para isso, destacam-se duas etapas como sendo as principais para efetivação do trabalho: criação e execução de *scripts*. A principio, foi realizado um levantamento sobre os trabalhos que descrevem o processo de criação e execução de cenários utilizando ferramentas para automatização de teste. Para a execução do conjunto, foi adotado um processo de cinco etapas para a geração de casos de teste de desempenho para aplicações *web* utilizado por (MOLINARI,2003) que serviu para complementar e definir o conjunto de características apresentado nesta dissertação.

A partir deste conjunto foram gerados os cenários com as cargas de tamanho pequeno, médio e grande utilizando as ferramentas de automatização de teste de desempenho: Apache JMeter e HP LoadRunner. Tinha como objetivo verificar as diferenças de configuração para a criação e execução de casos de teste. Ao final do capítulo, foi apresentado um comparativo referente à a geração e execução de casos de teste em ambas. Também foi realizada uma análise do comportamento de cada caso de teste gerado.

Foi possivel observar que as ferramentas são capazes de representar um cenário real. Também disponibilizam várias opções de relatórios para analisarem o cenário proposto. Porém, LoadRunner possui um número maior de opções enquanto JMeter possui a vantagem de adição de *Plugins*.

Para completar o trabalho, com as cargas geradas foi possível analisar os resultados dos gráfico e concluimos que à medida que as cargas aumentam o *throughput* também aumenta. No JMeter foi possivel observar também que o tempo de resposta aumenta na proporção do tamanho da carga. Representamos também o número de conexões abertas com o servidor e observamos que aplicações menores abriram menos conexões com o servidor. Mostramos um conjunto de requisitos necessários para a criação de *scripts* de forma genérica, para pudesse ser utilizado em qualquer ferramenta.

Diante deste cenário, sugerimos que os testes criados nessa dissertação possam ser utilizados futuramente para a avaliação de balanceamento de carga em Redes Definidas

por *Software* (SDN).

REFERÊNCIAS

[1] APACHE. Apache JMeter. Disponível em <http://jmeter.apache.org/> Acesso em 5 de maio de 2015

[2] ARLITT, M. F.; WILLIAMSON, C. L. Web Server Workload Characterization: the Search for Invariants. In: *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. Nova York, NY, EUA: ACM Press, 1996. p. 126–137.

[3] BARFORD, P. and CROVELLA, M. E. (1998). Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proceedings of 1998 ACM SIGMETRICS Int. Conf. Measurement and Modeling of Computer Systems*,Madison,Wisconsin.

[4] IBM (2015). IBM Rational Performance Tester. Disponível em <http://www-03.ibm.com/software/products/pt/performance > Acesso em 20 de maio de 2015.

[5] CALZAROSSA, M.; SERAZZI, *G. Workload Characterization: A Survey. IEEE Computer Society*, v. 81, n. 8, p. 1136–1150, Ago 1993.

[6] FARAH, Paulo R.; MURTA, Cristina D. *TORO: Um Gerador de Sobrecarga para Servidores Web*.

[7] GILL,P.;ARLITT,M.; LI,Z.;MAHANTI, A.(2007). YouTube traffic characterization: A view from the edge, *Proceedings of the Seventh ACM SIGCOMM Conference on Internet Measurement* (San Diego), pp. 15–28.

[8] GUARIENTI, Priscila; BERNARDINO, Maicon; ZORZO, Avelino; OLIVEIRA, Flávio. Uma Abordagem de Análise do Tempo de Resposta para Teste de Desempenho em Aplicações Web.*Anais do XV Workshop de Testes e Tolerância a Falhas*, Porto Alegre,2014.

[9] HUSSAIN,S.; WANG Z.;TOURE, I. K.; DIOP A. Web Service Testing Tools: A Comparative Study, *International Journal of Computer Science Issues (IJCSI)*, 10(1), Pages 641-647, (2013)

[10] HP. HP LoadRunner. Disponível em <http://www8.hp.com/br/pt/software-solutions/loadrunner-load-testing/> Acesso em 10 de maio de 2015

[11] JING, Y. et al .JMeter-based Aging Simulation of Computing System, in *Proceedings of the International Conference on Computer, Mechatronics, Control and Electronic Engineering*, 2010, pp. 282–285.

[12] MENASCé, D. A.; ALMEIDA, V. A. F. *Planejamento de Capacidade para Serviços na Web*. 1. ed. Rio de Janeiro, RJ: Campus, 2003. 472 p.

[13] MEIER,J.; VASIREDDY, A. ;BABBAR, A.; MACKMAN,A. *Improving. NET Application Performance and Scalability*. Microsoft Press, 2004

[14] MOLINARI, Leonardo. *Testes de Software: Produzindo Sistemas Melhores e Confiáveis*. São Paulo: Editora Érica Ltda, 2003

[15] PRESSMAN, Roger . *Software Engineering-A practitioner's Approach*, Fifth edition, McGraw-Hill, 2001.

[16] SCHNEIDER, F.; FELDMANN, A.; KRISHNAMURTHY, B.; WILLINGER, W. (2009). Understanding online social network usage from a network perspective, in: *Proceedings of the ACM SIGCOMM Conference on Internet*.

[17] TEIXEIRA, M. A. M. *Suporte a serviços diferenciados em servidores Web: modelos e algoritmos*. Tese(Doutorado) - ICMC-USP, São Carlos - SP, Março.2004

[18] WANG, Q.; MAKAROFF, D. J.; EDWARDS, H. K.; THOMPSON, R. Workload Characterization for an E-commerce Web Site. In: *Proceedings of the 2003 Conference of the Centre for Advanced Studies on Collaborative research*. Toronto, Ontario, Canada: IBM Press, 2003. p. 313–327.

[19] SILVA, L.H.C. *Caracterização de carga de trabalho para testes de modelos de servidores web*. Dissertação (Mestrado em Ciência da Computação e Matemática)- Universidade de São Paulo, São Paulo, 2006.

[20] WEBER, S.; HARIHARAN, R. A New Synthetic Web Server Trace Generation Methodology. In: *IEEE International Symposium on Performance Analysis of Systems and Software*. Austin, Texas: IEEE, 2003. p. 80–90.

[21] WiILLIAMS, A., ARLITT, M., WILLIAMSON, C., BARKER, K. (2005). *Web Workload Characterization: Ten years later*, In Web Content Delivery, v. 2, p. 3-21.