MARCELO JOSÉ SANTOS DA SILVA

UMA ABORDAGEM PARA AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS *WEB*

RECIFE-PE – MAIO/2015.

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA

UMA ABORDAGEM PARA AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS *WEB*

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada como exigência parcial à obtenção do título de Mestre.

Orientador: Prof. Dr. Fernando Antônio Aires Lins

Co-orientadora:Profª. Drª.Erica Teixeira Gomes Sousa

RECIFE-PE – MAIO/2015.

Ficha Catalográfica

S586a Silva, Marcelo José Santos da
Uma abordagem para avaliação de desempenho de serviços
Web / Marcelo José Santos da Silva. – Recife, 2015.
85 f.: il.

Orientador(a): Fernando Antônio Aires Lins.
Dissertação (Programa de Pós-Graduação em Informática
Aplicada) – Universidade Federal Rural de Pernambuco, Departamento de Informática, Recife, 2015.
Inclui apêndice(s) e referências.

1. Rede de Computadores 2. Sistemas distribuídos
3. Serviços Web 4. Metodologia de avaliação de desempenho
5. REST 6. SOAP I. Lins, Fernando Antônio Aires, orientador
II. Título

CDD 004.678

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA

UMA ABORDAGEM PARA AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS *WEB*

MARCELO JOSÉ SANTOS DA SILVA

Dissertação julgada adequada para obtenção do título de Mestre em Informática Aplicada, defendida e aprovada por unanimidade em 28/05/2015 pela Banca Examinadora.

Orientador:

_____

Prof. Dr. Fernando Antônio Aires Lins
Universidade Federal Rural de Pernambuco

Banca Examinadora:

_____

Prof. Dr. Gilberto Amado de Azevedo Cysneiros Filho
Universidade Federal Rural de Pernambuco

_____

Profª. Drª. Juliana Basto Diniz
Universidade Federal Rural de Pernambuco

_____

Prof. Dr. Sérgio Murilo Maciel Fernandes
Universidade de Pernambuco

Dedicatória

Dedico este trabalho a minha esposa Claudia, e ao meu filho Eduardo, pelo tempo que deixamos de estar juntos. Aos meus pais, Maria e Severino, a eles todos os créditos.

AGRADECIMENTOS

A Deus;

A minha família, em especial minha mãe Maria de Lourdes, meu pai Severino José, minha esposa Cláudia e meu filho Eduardo, por estarem sempre presentes em minha vida, seja em horas boas ou tristes, me dando alegria e amor;

Ao Prof. Dr. Fernando Antônio Aires Lins, pela dedicação nas correções e orientações neste período de aprendizado;

A Profª. Drª. Érica Teixeira Gomes Sousa, que ajudou bastante também fazendo correções e principalmente na parte de Simulação;

Agradeço também aos professores Sérgio Murilo, Juliana Diniz, Gilberto Cysneiros que aceitaram o convite para compor essa banca de mestrado;

A UFRPE, por ter apoiado esta minha iniciativa de capacitação.

“Eu Acredito que às vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar.”

Alan Turing.

RESUMO

UMA ABORDAGEM PARA AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS *WEB*

Um dos temas mais relevantes na área de computação, atualmente, é a Computação Orientada a Serviços. Este paradigma envolve, em geral, a utilização de serviços *Web* para a implementação das funcionalidades do negócio. Sistemas orientados a serviço, promovem o reuso e a interoperabilidade, namedida em que são desenvolvidos, usando padrões amplamente difundidos e aceitos (baseadosprincipalmente no protocolo HTTP). Neste contexto, dois protocolos vêm sendo usados para a invocação de serviços *Web*: SOAP e REST. Este trabalho propõe uma abordagem, que consiste em uma metodologia e um modelo, para avaliar o desempenho de serviços *Web*. A metodologia visa detalhar as atividades necessárias para a realização de um estudo de avaliação de desempenho especificamente focado em serviços, enquanto o modelo visa representar, com um nível aceitável de semelhança, o sistema real.Adicionalmente, também foi proposto um estudo de caso comparando a medição e a simulação dos resultados obtidos. Em especial, considerando que os dois protocolos de comunicação mais relevantes, SOAP e REST, foram propostos nos últimos anos para apoiar a execução de projeto paraserviços *Web*, o estudo de caso centra-se em um teste comparativo destes protocolos, a fim de avaliar os aspectos de desempenho em cenários do mundo real. Na fase de medição, foi criado um ambiente com os dois tipos de serviços *Web* instalados em um servidor. Já o modelo de simulação foi criado com base no cenário avaliado na fase de medição. Em todos os testes realizados neste trabalho, apontam que o SOAP obteve um resultado inferior em relação a desempenho em comparação com o REST no estudo desenvolvido.

Palavras-chave: Redes de Computadores, Sistemas Distribuídos, serviços *Web*, Metodologia de Avaliação de desempenho, REST, SOAP

ABSTRACT

AN APPROACH FOR PERFORMANCE EVALUATION OF *WEB* SERVICES

One of the most relevant topics in computing today is the Service Oriented Computing. This paradigm involves, in general, the use of Web services for the implementation of business capabilities. Service oriented systems, promote the reuse and interoperability insofar as they are developed using widely diffused and accepted standards (mainly based on the HTTP protocol). In this context, two protocols have been used for the invocation of Web services: SOAP and REST. This paper proposes an approach, consisting of a methodology and a model to evaluate the performance of Web services. The methodology aims to detail the activities necessary for the achievement of a performance evaluation study specifically focused on services, while the model is intended to represent with an acceptable level of similarity, the actual system. Additionally, it was also proposed a case study comparing the measurement and simulation of results. In particular, considering that the two most important communication protocols, SOAP and REST have been proposed in recent years to support design for implementing Web services, the case study focuses on a comparative test of these protocols, in order to evaluate performance issues in real-world scenarios. In the measurement phase, an environment was created with two kinds of Web services installed on a server. But the simulation model was created based on the assessed scenario in the measurement phase. In all tests performed in this work show that the SOAP achieved a lower result compared to performance compared to REST in the study developed.

Keywords: Computer network, Distributed Systems, Internet, Methodology for Performance evaluation, REST, SOAP, serviços*Web*

LISTA DE ILUSTRAÇÕES

ÍNDICE DE TABELAS

LISTA DE ACRÔNIMOS

API - *Application Programming Interface*
CORBA - *Common Object Request Broker Architecture*
CPU - *Central Processing Unity*
CRUD - *Create, Read, Update e Delete*
CSV - *Comma-separated values*
DOE - *Design of Experiments*
HTTP - *Hypertext Transfer Protocol*
IDE - *Integrated Development Environment*
JAX RS - *Java API for RESTfulWeb Services*
JAX WS - *Java API for XML Web Services*
JDK - *Java Development Kit*
JSON - *JavaScript Object Notation*
JVM - *Design of Experiments*
LAN - *Local Area Network*
MAN - *Metropolitan Area Network*
PHP - *Hypertext Preprocessor*
REST - *Representational State Transfer*
RPC - *Remote Procedure Call*
RPM – Rotações Por Minuto
SOAP - *Simple Object Access Protocol*
SOC - *Service Oriented Computing*
TCP - *Transmission Control Protocol*
UDDI - *Universal Description, Discovery and Integration*
ULA – Unidade Lógica e Aritimética
URI - *UniformResourceIdentifier*
URL - *Uniform Resource Locator*
WAN - *Wide Area Network*
WS – *Web Service*
WSDL - *Web Services Description Language*
XML - *eXtensibleMarkupLanguage*

SUMÁRIO

1 INTRODUÇÃO

1 INTRODUÇÃO

Este capítulo apresenta uma breve introdução relacionada ao contexto geral da área em que o trabalho se refere.Mais especificamente, são apresentados a motivação do trabalho, um resumo do estado da arte e suas deficiências, a solução proposta e a estrutura do trabalho.

1.1 MOTIVAÇÃO

A convergência das tecnologias de comunicação e computação têm favorecido o surgimento de novos tipos de serviços e aplicações oferecidos pela Internet, contribuindo consideravelmente para o seu crescimento. Hoje, várias transações comerciais, antes realizadas apenas presencialmente com a participação do cliente ou por telefone, agora podem ser realizadas com a facilidade e flexibilidade propiciada pelo uso da Internet. Alguns exemplos desses serviços são: sistemas de banco, comércio eletrônico (*e-commerce*), educação a distância, multimídia e entretenimento, entre outros. Grande parte dos serviços atualmente utilizam a Internet como via de informação principal e novas demandas são apresentadas observando questões de desempenho, confiabilidade e segurança.

Nos últimos anos, um tema que vem ganhando interesse é o *Service Oriented Computing* (SOC) [1]. Este paradigma envolveu uso de serviços *Web* para a implementação das funcionalidades do negócio. Sistemas orientados a serviços promovem a reutilização e interoperabilidade, basicamente porque eles são desenvolvidos utilizando padrões amplamentes disseminados e aceitos (na sua maioria baseados no protocolo HTTP), o que facilita sua aprovação pela comunidade científica. Dois protocolos de comunicação têm sido usados no contexto deserviços *Web*: SOAP (*Simple Object Access Protocol*)[2] e REST (*Representational State Transfer*) [3]. O protocolo SOAP é considerado mais tradicional e está sendo utilizado por empresas relevantes (por exemplo, IBM, Microsoft e outros), enquanto que o protocolo REST obteve um crescimento considerado nos últimos anos [4].

Uma questão crucial no desenvolvimento de aplicações baseadas em serviços é a escolha do protocolo a ser utilizado. Tanto o REST[3] quanto o SOAP[2] podem, e têm sido, utilizados para esta finalidade. No entanto, eles diferem

consideravelmente em relação a I) ferramentas/mecanismos necessários para a sua execução; II) tempo de execução e III) apoio a requisitos não funcionais (tais como segurança, desempenho e custo). Mais especificamente, alguns trabalhos relacionados atuais enfatizam o impacto de desempenho que podem aparecer, dependendo do protocolo de comunicação adotado. No cenário atual, dependendo do tipo de sistema, há sempre uma pergunta sobre qual configuração de serviços *Web* deve ser utilizada [5].

Em relação as ferramentas e mecanismos necessários para a execução de cada protocolo, normalmente é uma tarefa bastante trabalhosa instalar e configurar serviços *Web*, às vezes por falta de materias e também por conta da sua complexidade. Antes mesmo de inicar a fase dos mecanismos e ferramentas, é importante identificar se o protocolo oferece os requisitos não-funcionais que são necessários para o objetivo. Logo após estas duas fases, pode-se realizar os testes de desempenho e identificar qual se adequa melhor a necessidade.

Avaliar o desempenho de determinado objeto é uma tarefa bastante importante para qualquer área, principalmente em serviços *Web*. Neste caso específico, uma avaliação prévia de qual protocolo utilizar ainda na fase de planejamento de requisitos, pode economizar tanto tempo quanto recursos financeiros.

Atualmente existe a falta de uma abordagem que descreva como realizar um estudo de avaliação de desempenho de serviços *Web*, focado em cada passo necessário para planejar, executar e analisar um estudo. Alguns estudos até apresentam algum tipo de proposta similar [6], [7], [8], apresentam um teste relacionando alguns tempos de respostas entre os dois protocolos, outros mostram a quantidade de *bytes* enviados e recebidos pelo servidor, porém não apresentam detalhes de como foi feito o trabalho. Este estudo foi proposto devido a deficiência atual de uma abordagem que apresente, com detalhes, como realizar um estudo de avaliação de desempenho de serviços *Web*.

1.2 LACUNA NO ESTADO DA ARTE ATUAL

Prover uma avaliação de desempenho detalhada em relação ao funcionamento de um serviço *Web* é uma tarefa complexa. Existem uma série de

passos a seguir:Primeiramente a instalação do ambiente em si; logo após a configuração do serviço *Web*; o entendimento do funcionamento do mesmo; depois disso identificar qual ferramenta de captura de dados melhor se adequa a sua necessidade.

Clinks e Malley [6] avaliaram o desempenho dos serviços *Web* SOAP e REST, porém voltado para a execução de Arquiteturas Orientadas a Recursos. A principal contribuição de Clinkse Malley é a visão geral de ambos os estilos de serviços *Web* e suas arquiteturas associadas, bem como a apresentação dos resultados experimentais em que demonstram o desempenho de cada estilo, ao longo de um número de cenários de testes.

Outro estudo que também se concentrou sobre este assunto é o de Potti[9]. No trabalho de Potti, os autores também fazem várias requisições HTTP como se fossem clientes reais, porém fazem alterações em pedidos de clientes com e sem fio, também fazem alguns testes para realizar o *upload* de arquivos e comparam todos os resultados tanto em REST quanto em SOAP. Mas o autor não explica em detalhes as características sobre o ambiente adotado ou a forma como os experimentos foram realizados. Em outras palavras, esta iniciativa não apresenta uma abordagem geral (composta de uma série de passos a serem seguidos) para melhor representar ao leitor e possíveis interessados como poderia ser replicada as experiências em outros contextos e ambientes.

Na literatura relacionada a simulação, tem-se o trabalho de Santos[8] que fez um modelo de simulação de um processo de requisições em um servidor *Web* de alta disponibilidade. O modelo foi gerado a partir de informações extraídas dos *logs* de sistema, que são arquivos gerados pelo sistema operacional após cada evento que nele acontece.

1.3 SOLUÇÃO PROPOSTA

Este trabalho apresenta uma abordagem para avaliação de desempenho de serviços *Web*. A necessidade de uma metodologia para este fim é baseada no fato de que uma quantidade considerável de aplicações de vários tipos estão usando os serviços *Web* atualmente, e a avaliação do desempenho pode ajudar os analistas de negócios a escolher quais configurações de protocolo e *hardware* devem ser

utilizados considerando uma carga de trabalho específica. A metodologia é composta por atividades que utilizam as técnicas de medição e simulação para avaliação de desempenho de serviços *Web*. Esta metodologia foi inspirada em uma obra similar desta área [10], mas que foi adaptada para o contexto de serviços *Web*, além de conter um modelo de simulação.

Considerando este contexto, o presente trabalho propõe uma abordagem para a medição e modelagem de serviços *Web*. Essa abordagem é composta de uma metodologia, que descreve um conjunto de passos necessário para a execução da avaliação de desempenho.Uma aplicação interessante da metodologia é a avaliação do desempenho de protocolos de comunicação de serviços *Web* (mais especificamente, REST e SOAP). Essa avaliação é importante porque proporciona a seleção dos cenários mais favoráveis para utilizar o protocolo SOAP e quais são os outros cenários em que o uso do protocolo REST é mais adequado.Um estudo de caso foi feito tanto com a medição quanto com a simulação do ambiente. Na medição, foram utilizadas cargas de requisições HTTP simulando usuários acessando simultaneamente os serviços *Web*. As métricas obtidas destes estudos de cargas foram: tempo de resposta, *throughput* e tempo de utilização do CPU do servidor que hospedou os serviços *Web*. Já o modelo de simulação representou o cenário da medição e pode-se utilizar um número muito maior de requisições do que na medição. Esse número de requisições é limitado pelo *hardware*do qual se faz as solicitações.Além da medição, foi utilizada a Simulação para avaliar o teste de desempenho. De acordo com Freitas[11], a simulação tem sido cada vez mais aceita e utilizada, visto que houve aumento significante no poder de processamento das estações de trabalho aliadas à simplicidade de uso e à sofisticação dos ambientes de desenvolvimento de modelos computacionais[11].

Nesse contexto, a simulação discreta torna-se útil, pois é uma excelente ferramenta e proporciona aos administradores de servidores *Web* a certeza que está sendo bem administrado e opera buscando o máximo de eficiência possível.

1.4 ESTRUTURAÇÃO DO DOCUMENTO

Além deste capítulo de introdução, o presente trabalho inclui ainda mais cinco capítulos, como descrito a seguir:

Capítulo 2: Neste ponto, uma discussão sobre os trabalhos relacionados a este trabalho é apresentada. Recentemente dissertações vêm tratando deste tema em particular, porém sem detalhamento na metodologia que foram aplicadas. Esses trabalhos são apresentados, e uma avaliação sobre os mesmos é feita. Por fim, considerações comparativas sobre esta dissertação e esses trabalhos relacionados são realizadas.

Capítulo 3: neste capítulo serão introduzidos todos os conceitos básicos associados a este trabalho. Mais especificamente, são detalhados os aspectos relacionados aos serviços *Web* (SOAP e REST) e a avaliação de desempenho.

Capítulo 4: este capítulo apresenta uma metodologia para avaliação de desempenho de serviços *Web*. Inicialmente, os princípios básicos desta metodologia são apresentados, vindo logo em seguida como é feita a avaliação. Ainda neste capítulo é apresentado um modelo de simulação para dar suporte ao estudo de desempenho.

Capítulo 5: O estudo de caso deste trabalho é apresentado neste capítulo. No estudo de caso foi feita uma avaliação de desempenho no protocolo REST e no SOAP e logo após foram feitas comparações entre eles.Este capítulo também mostrará como a simulação foi feita, de acordo com o cenário de medição, e quais configurações foram necessárias, além de mostrar os resultados obtidos.

Capítulo 6: Finalmente, este capítulo apresenta as conclusões e os trabalhos futuros associados a esta dissertação. São explicadas também as contribuições da mesma e as limitações da abordagem apresentada, seguidas de propostas de aperfeiçoamento.

2 TRABALHOS RELACIONADOS

2 TRABALHOS RELACIONADOS

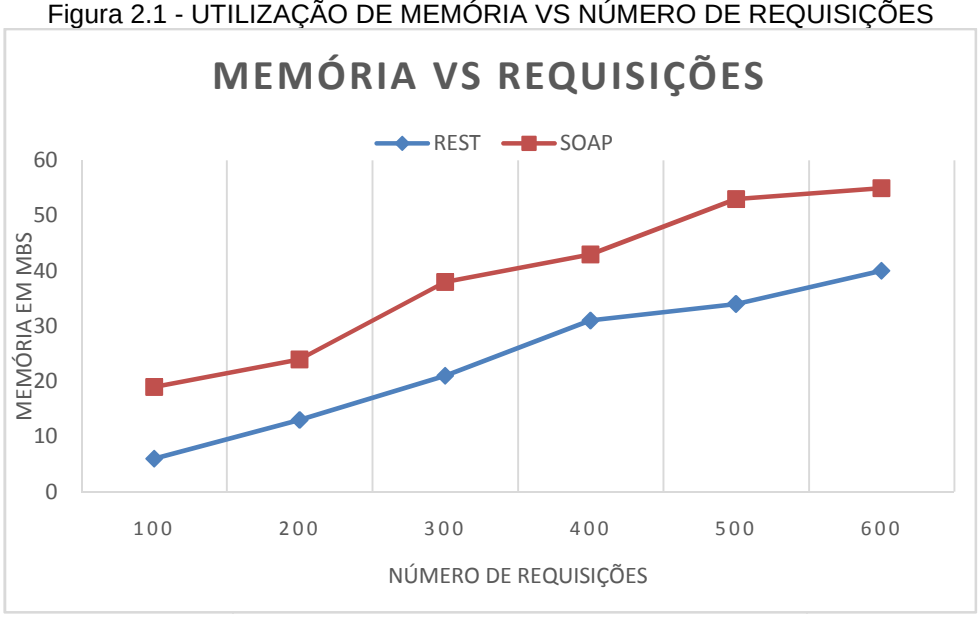
Neste capítulo são apresentados os trabalhos relacionados ao tema desta dissertação. Mais especificamente, considerando a contribuição principal deste trabalho (que envolve medição e simulação), são apresentadas iniciativas relevantes que foquem em pelo menos uma dessas duas temáticas (medição ou simulação) na área de serviços *Web*. Adicionalmente, uma visão crítica e comparativa dos mesmos em relação a este trabalho será apresentada.

2.1 AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS *WEB*ATRAVÉS DE MEDIÇÃO

A primeira iniciativa a ser apresentada nesta seção é o trabalho de Wambua [12]. Neste trabalho, o autor apresenta um estudo comparativo dos dois tipos de serviços *Web,* o REST e o SOAP, e faz um estudo de caso prático na Secretaria de Partidos Políticos do Kenya. Nesta iniciativa, foi feita a comparação das duas tecnologias e, baseado nos dados da quantidade de memória utilizada por cada requisição, foram feitas comparações da utilização de memória pelo servidor utilizando os dois tipos de serviços *Web* o REST e o SOAP. Também foi feita uma comparação entre o tempo de resposta obtido. A medição deste trabalho, foi feita através do mesmo software de cargas utilizado nesta dissertação, o JMeter. Mesmo com bastante informações relevantes, o trabalho não deixou claro quais foram as métricas que utilizou-se nem também a quantidade de carga de trabalho que foi utilizada.

Um resultado bem interessante do trabalho de Wambua [12] foi a comparação da utilização de memória dos dois serviços *Web*. Como se pode observar na Figura 2.1, a utilização de memória aumenta de acordo com o número de requisições. O resultado mostra que utilizando o serviço *Web* SOAP, em todos os testes realizados, a quantidade de memória requisitada foi superior ao REST.

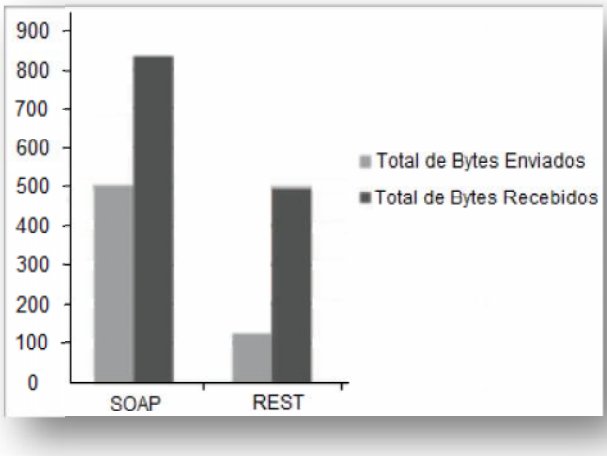
Figura 2.1 - UTILIZAÇÃO DE MEMÓRIA VS NÚMERO DE REQUISIÇÕES



Outro trabalho importante é o de Markey e Clynch [6]. Eles abordaram a quantidade de tráfego de rede de ambos os protocolos de comunicação para serviços *Web* (SOAP e REST).

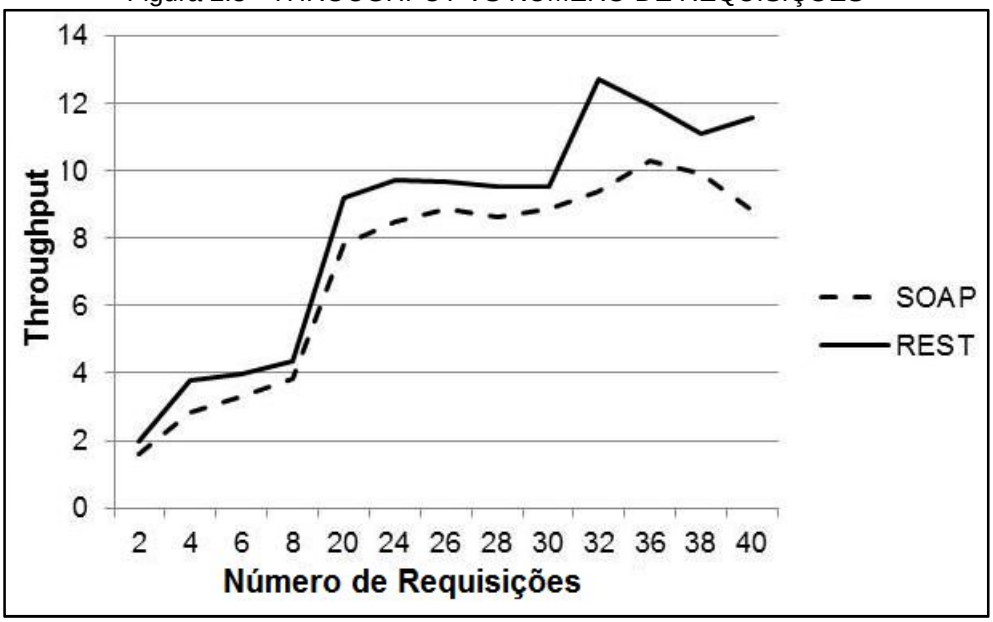
Dentro deste cenárioapresentado, os resultados podem ser sumarizados conforme exposto na Figura 2.2. Esta figura mostra que o envio foi de 505 *bytes* no protocolo SOAP e apenas 128 *bytes*no protocolo REST, isto representa apenas 25% do tamanhodo pedido SOAP. Sobre os dados recebidos, a chamada REST em comparaçãopara a resposta SOAP foi basicamente a metade deste com apenas 496 *bytes*. No trabalho não há detalhes de como foram escolhidas as métricas, e nem de quantas requisições foram utilizadas.

Figura 2.2 - QUANTIDADE DE BYTES ENVIADOS E RECEBIDOS



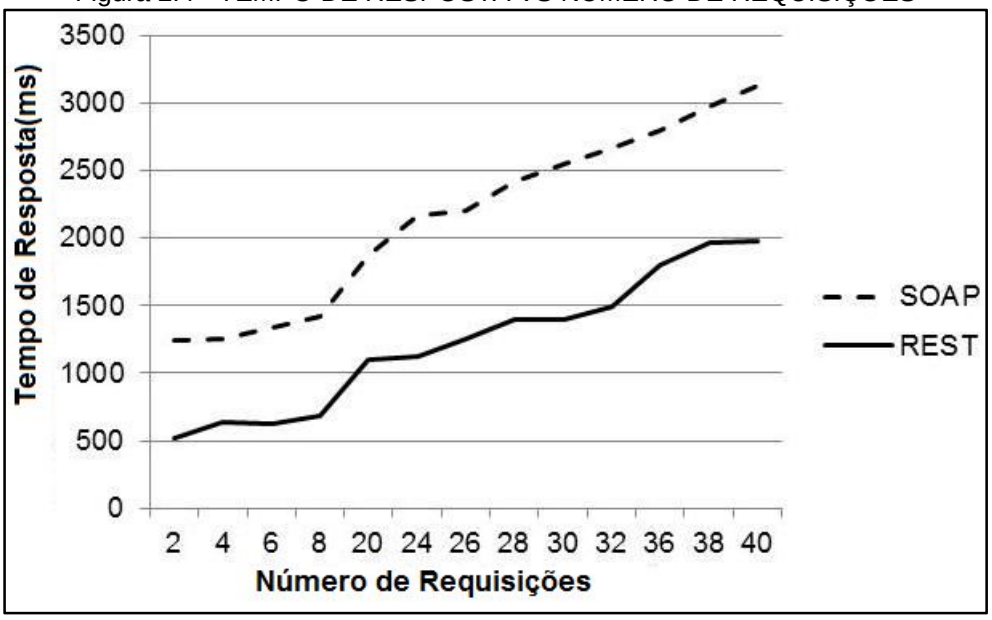
Por outro lado, Kumar [9] fez um estudo comparando o desempenho dos estilos de interações dos serviços *Web* SOAP e REST. No ambiente em que foram realizados os testes, o autor aborda duas métricas comparando os dois serviços *Web*, o *throughput* e o tempo de resposta. Todo projeto foi feito em cima das operações do CRUD, e testes foram feitos e os resultados foram obtidos do *log* do servidor *Web*. Uma delas foi o *throughput* em relação a quantidade de requisições que foram enviadas, e o resultado associado está apresentado na Figura 2.3. Também nesta figura pode-se notar que o REST teve uma vazão maior de dados, por ser mais rápido em tempo de execução e com respostas com menos *bytes* em relação ao SOAP.

Figura 2.3 - THROUGHPUT VS NÚMERO DE REQUISIÇÕES



Ainda no trabalho de Kumar, pode-se observar na Figura 2.4, a comparação do tempo de resposta de acordo com o número de requisições HTTP. O resultado foi que o REST, em relação ao SOAP, obteve um tempo de resposta menor. Sendo assim, o resultado encontrado por Kumar, seguiu a mesma tendência dos resultados descritos no estudo de caso desta dissertação. A diferença foi que na metodologia proposta por Kumar, não é reaplicável a outros ambientes de medição.

Figura 2.4 - TEMPO DE RESPOSTA VS NÚMERO DE REQUISIÇÕES

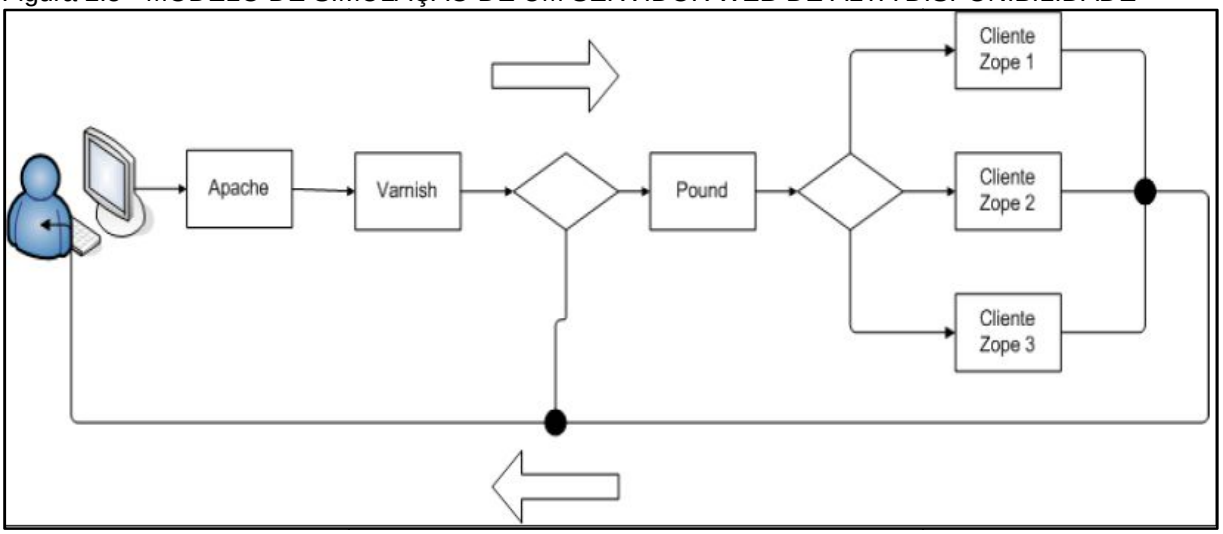


2.2 AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS *WEB* ATRAVÉS DE SIMULAÇÃO

Nesta seção, o foco será a apresentação de trabalhos que abordem a avaliação de desempenho de serviços *Web* usando a técnica de simulação. Inicialmente, uma iniciativa interessante da área é o trabalho de Santos [8], que fez um modelo de simulação de um processo de requisições em um servidor *Web* de alta disponibilidade. O modelo foi gerado a partir de informações extraídas dos *logs* do servidor *Web*, os quais são arquivos gerados pelo sistema operacional após cada evento que acontece nele [8].

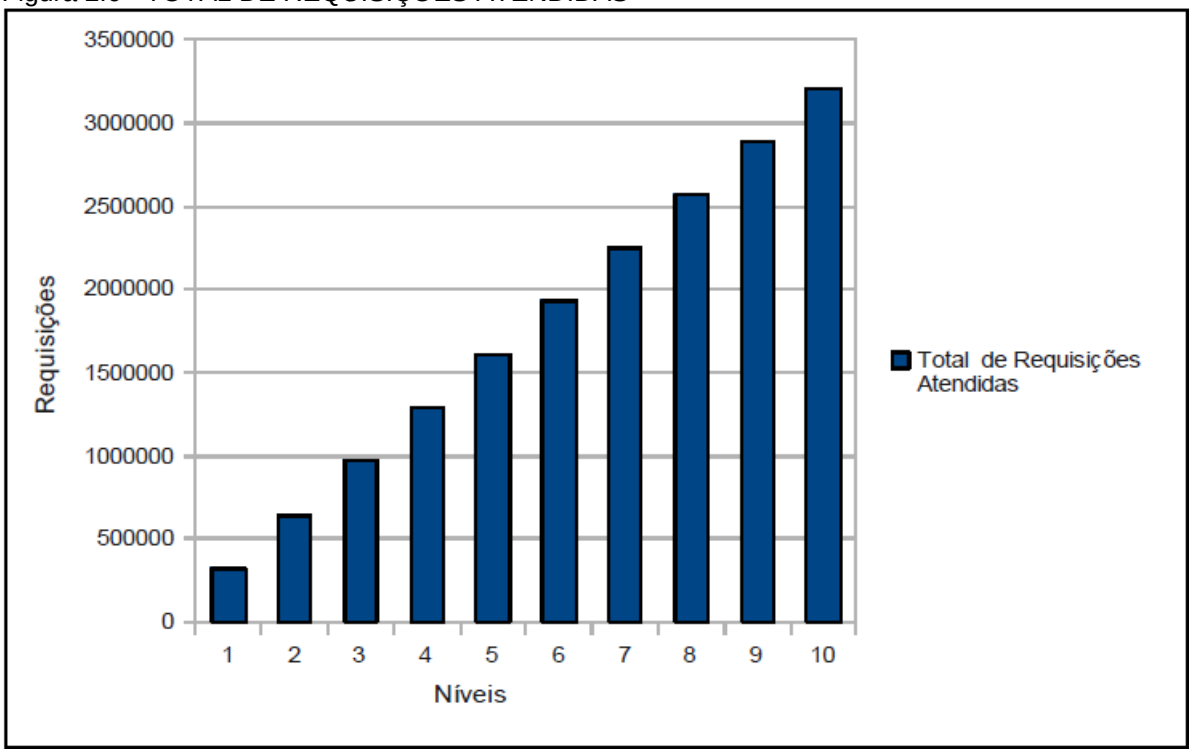
Na Figura 2.5 é apresentado um modelo conceitual da simulação disposta no trabalho. A requisição que chega ao servidor *Web* é processada e encaminhada para um *software* chamado Varnish[13], que é responsável por fazer o *cache* das informações; ou seja, ele consegue guardar, dentro de um limite de espaço, as informações que são mais comumente requisitadas e, assim, retornar diretamente para o cliente, reduzindo o tempo de atendimento à requisição. Essa etapa otimiza aproximadamente 32% das requisições recebidas pelo servidor [8]. O Pound[14] é responsável por encaminhar a requisição de forma balanceada para um dos três clientes Zope[15].

Figura 2.5 - MODELO DE SIMULAÇÃO DE UM SERVIDOR *WEB* DE ALTA DISPONIBILIDADE



Um resultado interessante da iniciativa se refere à quantidade de requisições atendidas nos diferentes cenários simulados, conforme a Figura 2.6. O modelo foi preparado para rodar em um período de um dia útil, contando às 24h, e cada nível foi replicado 10 vezes, a fim de obter uma convergência dos resultados.

Figura 2.6 - TOTAL DE REQUISIÇÕES ATENDIDAS



Outro trabalho bem relevante que utiliza simulação em serviços *Web* é o de Ardagna [16]. O autor observou que muitas vezes os projetistas de processos só obtém uma avaliação confiável do desempenho de um serviço muito tarde no ciclo de vida de um projeto. Uma vez que os serviços já foram implantados, não é possível avaliar o comportamento do serviço, visto que eles já estarão sendo utilizados. Este trabalho propõe uma metodologia que gera um *script* de simulação e este pode ser utilizado para uma avaliação precoce do desempenho do serviço. Este trabalho foi voltado para uma avaliação de mais alto nível e de métodos específicos de um projeto. Foi feito um estudo de caso simulando um caixa ATM, e feito comparações com as funções de Adicionar Dinheiro e a Retirada de dinheiro. Neste caso foi utilizado o protocolo SOAP para fazer os testes das requisições. O que pode-se notar é que este trabalho foi específico para engenharia de projetos, pois a metodologia criada foi específica para este fim.

A Tabela 2.1 mostra as características dos trabalhos mais relacionados ao trabalho proposto (Baseline). Essa tabela mostra que grande parte dos trabalhos adotam apenas uma estratégia. O trabalho proposto adota o uso destas duas características trabalhando em conjunto, além de trazer uma abordagem prática para avaliação de desempenho de serviços *Web*.

Tabela 2.1 - CARACTERÍSTICAS DOS TRABALHOS RELACIONADOS

Artigos	Medição	Simulação
Baseline	X	X
Wambua[12]	X	-
Markey e Clynch[6]	X	-
Kumar[7]	X	
Santos[8]	-	X
Ardagna[16]	-	X

2.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais trabalhos relacionados ao tema desta dissertação. De forma resumida, existiram dois tipos de iniciativas; uma é relacionada a medição e a outra a simulação. O trabalho proposto nesta dissertação mostrou com detalhes a junção dos dois trabalhos, a medição sendo complemento da simulação.

3 FUNDAMENTAÇÃO TEÓRICA

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são introduzidos os conceitos básicos e as tecnologias adotadas neste trabalho. Inicialmente, uma explicação sobre o que são serviços *Web* é apresentada. Em um segundo momento, são introduzidos os dois protocolos de comunicaçãodeserviços *Web* mais utilizados atualmente no mundo atualmente. Finalmente, as técnicas de avaliação de desempenho são introduzidas.

3.1 SERVIÇOS *WEB*

De acordo com a W3C[2], os serviços *Web* fornecem um padrão de interoperabilidade entre diferentes aplicações de *software*, rodando em uma variedade de plataformas. Serviços *Web* são caracterizados por facilitarem atributos importantes de sistemas distribuídos, como interoperabilidade e extensibilidade, bem como o seu processamento realizado entre máquinas (aplicações) devido a descrições utilizando padrões de representação, tais como XML. Serviços *Web* podem ser combinados de forma flexível, a fim de construir aplicações mais complexas.Permitem que os sistemas que fornecem serviços simples possam interagir uns com os outros, a fim de fornecer serviços sofisticados. Outra definição interessante é apresentada em Ben-harref[17], que define os serviços *Web* como um “aplicativo que expõe sua funcionalidade através de uma descrição de uma interface padronizada que o torna disponível para ser usado por outros programas (clientes)”.

Como mencionado anteriormente, serviços *Web* nada mais são do que uma nova tecnologia de computação distribuída. Sendo assim, existem alguns pontos que a diferencia de outras.Os pontos são:

- Independência de plataforma – Independe da linguagem de programação, pois utiliza o XML, que é uma linguagem padrão para troca de dados;
- Utilização de HTTP para transmissão de mensagens – Protocolo mais difundido e utilizado atualmente.

Um número considerável de protocolos têm sido propostos nos últimos anos para apoiar os aspectos funcionais e não funcionais de serviços *Web*. A primeira

geração destes protocolos foram compostas por SOAP, WSDL(*Web ServiceDescriptionLanguage*) e UDDI [50]. Em suma, o SOAP foi utilizado como protocolo de comunicações (isto é, fornece o formato de mensagens para a comunicação entre o cliente e o serviço), o WSDL foi usada para descrever a interface de serviços (isto é, que serviço pode realmente ser feito e quais os parâmetros necessários para invocá-lo) e o UDDI foi destinado a ser um facilitador padrão para registro de serviço. O WSDL foi usado em conjunto com o SOAP para tornar as mensagens disponíveis viaServiços *Web*.

Os protocolos de primeira geração foram largamente utilizados na última década. No entanto, eles eram vistos como insuficientes para aplicações mais "complexas" (isto é, sistemas corporativos que exigiam suporte a vários requisitos funcionais e não-funcionais mais elaborados). Com base nisso, um conjunto de protocolos foi proposto e citado como a segunda geração de protocolos de serviços *Web[18],*tais como *WS-Policy*, *WS-Security*, *WS-Transaction* e assim por diante. Em adição, durante este período, outro protocolo de comunicação ganhou interesse pela indústria: *Representational State Transfer* (REST[3]). Competindo nos dias atuais com o SOAP, o REST está sendo visto como um protocolo leve em que os serviços podem ser invocados, exigindo uma infraestrutura menos robusta. As funções do REST estão disponíveis para os quatro comandos mais comuns do HTTP: GET, POST, PUT e DELETE. Baseado no fato de que todos os protocolos de segunda geração (WS- *) são baseados em SOAP, esta nova abordagem impôs várias mudanças no cenário de computação orientada a serviços, porque a maioria dos protocolos e aplicativos desenvolvidos não foram inicialmente desenhados para serem executados com REST.

Um ponto central do desenvolvimento de sistemas de serviço orientado, é a escolha do protocolo de comunicação. Esta escolha impacta não só no desempenho da invocação de serviço, mas também no uso de sistemas. Com base nisso, os protocolos de comunicação de serviços da *Web* (SOAP e REST) desempenham um papel muito importante nos sistemas baseados em serviços. Considerando este fato, eles são detalhados nas próximas seções.

3.1.1 PROTOCOLO SOAP

O protocolo SOAP[2], que foi criado inicialmente para possibilitar a invocação remota de métodos através da internet, é um protocolo para troca de informações estruturadas em redes de computadores. Em um contexto geral, SOAP em conjunto com WSDL é chamado serviços *Web* SOAP. Até pouco tempo atrás, sem o SOAP, serviços *Web*eram incapazes de cruzar sistemas operacionais e plataformas, e muitas vezes havia falta de interoperabilidade. SOAP foi concebido para preencher esta lacuna, tornando mais fácil para o desenvolvedor projetar um programa *Web* independente destas limitações.

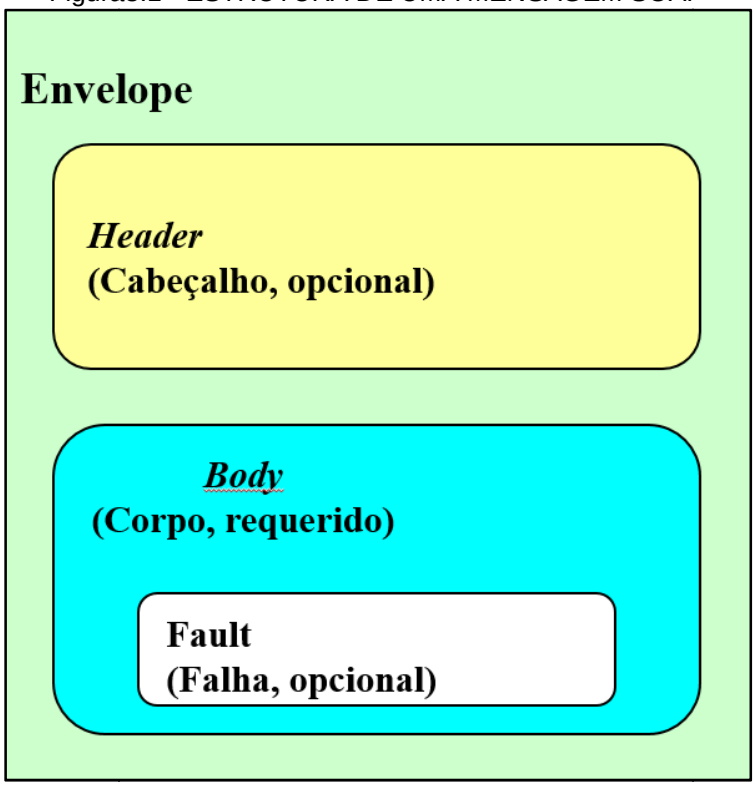
SOAP atualmente atua sobre HTTP como um protocolo subjacente, mas pode usar outros protocolos para a transferência de dados. A Microsoft foi a primeira a começar a desenvolver SOAP, e logo depois a *Ariba*, *CommerceOne*, *Compaq*, *DevelopMentor*, HP, IBM, *Lotus*, SAP e *UserLand*contribuiram para o seu desenvolvimento. A maior parte dos protocolos de serviços *Web* tentam "unir" os computadores em plataformas semelhantes, pois não é sempre que se tem a mesma tecnologia e o mesmo sistemas operacional. A necessidade de um protocolo que fosse independente destas limitações deu origem ao protocolo SOAP, que usa XML para transmitir mensagens sobre a camada de transporte.

Um cliente SOAP pode ser concebido apenas por saber a URL do servidor e o endereço da porta do serviço. Isto mais tarde tornou-se mais fácil com o advento do WSDL e do apoio de IDEs (*IntegratedDevelopmentEnvironment*). O WSDL é um padrão, baseado em XML, utilizado para a especificação das interfaces dos serviços *Web*, servindo como base para informar aos possíveis clientes o que o serviço pode fazer, como invocá-lo, qual o formato das respostas e outras informações requeridas para a correta invocação.Por exemplo,WSDL é extensível para permitir a descrição da URL e suas mensagens, independentemente do formato ou protocolos de rede[19].

As mensagens SOAP são vistas como envelopes que a aplicação utiliza para enviar os dados. Na Figura 3.1 é ilustrada uma mensagem SOAP, o envelope é um elemento obrigatório, e serve como um container para os demais. O *header*, é um elemento opcional e contém informações específicas para alguma aplicação (autenticação, criptografia). Já o *body*, é outro elemento obrigatório, nele estão

contidas as informações trocadas entre o transmissor e o receptor. O elemento*fault*do SOAP é opcional e é usado para indicar mensagens de erro.

Figura3.1 - ESTRUTURA DE UMA MENSAGEM SOAP



Logo abaixo na Figura 3.2, tem-se uma solicitação de um CEP, de acordo com as informações enviadas do nome do endereço, que neste exemplo, foi “Rua Alfredo de Medeiros”.

Figura 3.2 - SOLICITAÇÃO SOAP (CLIENTE)

```
<?xml version='1.0' encoding='ISO8859-1' ?>
<SOAP-ENV: Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <SOAP-ENV:Body>
            <ns1:getCEP>
               xmlns:ns1="urn = correios-cep"
               SOAP-ENV:encodingStyle=
                              "http://schemas.xmlsoap.org/soap/encoding" >
                  <end-postal> Rua Alfredo de Medeiros </end-postal>
            </ns1:getCEP>
      </SOAP-ENV:Body>
</SOAP-ENV: Envelope>
```

A Figura 3.3 apresenta a resposta do servidor, informando o CEP desejado no XML, que neste caso foi o 52021030.

Figura 3.3 - RESPOSTA DO SOAP (SERVIDOR)

```
<?xml version='1.0' encoding='ISO8859-1' ?>

<SOAP-ENV: Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsi="http://www.w3.org/2001/X MLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <SOAP-ENV:Body>
        <ns1:getCEPResponse
        xmlns:ns1="urn=Correios-CEP"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding" >
            <returnCEP xsi:type="xsd:string"> 52021030 </returnCEP>
          </ns1:getCEPResponse
      </SOAP-ENV:Body>
<SOAP-ENV: Envelope>
```

3.1.2 PROTOCOLO REST

REST é uma arquitetura para o desenvolvimento de serviços *Web*. REST tenta imitar arquiteturas que usam HTTP ou protocolos semelhantes, através da limitação da interface para um conjunto bem conhecido de operações padrão (ou seja, GET, PUT, POST e DELETE para HTTP)."A arquitetura REST está projetada para mostrar como o HTTP existente é suficiente para construir um serviço*Web* e para mostrar sua escalabilidade "[3].

A principal ideia por trás do REST era usar o HTTP,que já é bem desenvolvido e largamente utilizado, para transferir dados entre máquinas, ao invés de utilizar um protocolo que trabalhe no topo da camada HTTP para transferências de mensagens. Uma aplicação seguindo os princípios do REST usaria HTTP para fazer chamadas entre as máquinas, em vez de confiar em mecanismos complexos como CORBA (*Common Object Request Broker Architecture*) [20], RPC (*Remote Procedure Call*), ou SOAP. Assim, as aplicações em REST usariam funções de solicitação HTTP para postar dados, ler dados, e excluir dados, utilizando assim a funcionalidade completa das operações CRUD(Criar, Ler, Atualizar e Deletar) do

HTTP. REST também pode ser executado em HTTPS, que prevê um caminho seguro para transmissão de dados.

A aplicação dos princípios do REST para serviços *Web* exigem protocolos como o HTTP. Qualquer programador ou usuário confortável com HTTP, em teoria, vai achar que é fácil compreender e aplicar os princípios do REST.

URI é o endereço *Web* do recurso, e faz com que este seja disponível online. Um recurso pode ser acessado / referenciado pelo seu URI.Um outro conceito de REST é *stateless,*que significa que todas as solicitações HTTP são independentes uma da outra. Cada solicitação HTTP originada do cliente deve conter todas as informações para o servidor, para que o servidor não tenha que manter informações de estado de solicitações deste cliente. Já que o endereçamento é*stateless*, todos os estados possíveis do servidor também são recursos e devem ser representados por um URI [3].

Considere o exemplo de uma *Blockbuster* (uma locadora de DVD); pode-se procurar a disponibilidade de um DVD online. A pesquisa irá representar um DVD como disponível, chegará em breve, ou indisponível, por isso o mesmo recurso, o DVD, é representado em diversos estados. Isto ilustra a importância da palavra-chave "Representação" em REST. De acordo com Fielding[3], existem duas perspectivas regulares sobre o processo de concepção arquitetônica; um é para começar a construir tudo sem qualquer plano ou construir a partir do zero, e mais tarde adicionar componentes para corresponder às exigências. A outra é começar a construir um plano para implementar todos os requisitos. As características adicionais podem ser adicionadas mais tarde tornando-os de acordo com os recursos existentes e completá-las, dependendo dos requisitos[3].

Os métodos do REST no lado do servidor são os seguintes:

- GET: todos os métodos de obtenção de dados do servidor; os formatos e interfaces que o servidor suporta para acessar os detalhes do cliente.
- POST: todos os métodos de adicionar detalhes para o servidor; todas as diferentes interfaces e formatos que o servidor suporta para adicionar dados ao seu banco de dados.
- PUT: todos os métodos para atualizar os dados no servidor;

- DELETE: todos os métodos para apagar os dados no servidor;

Dada a facilidade de criação e flexibilidade na codificação fornecida por REST, vem gradualmente tornando-se popular. Yahoo e eBay foram os primeiros a usar o REST para projetar seusserviços da *Web*. Eles se juntaram posteriormente a empresas populares, como *Amazon* e *Google*.

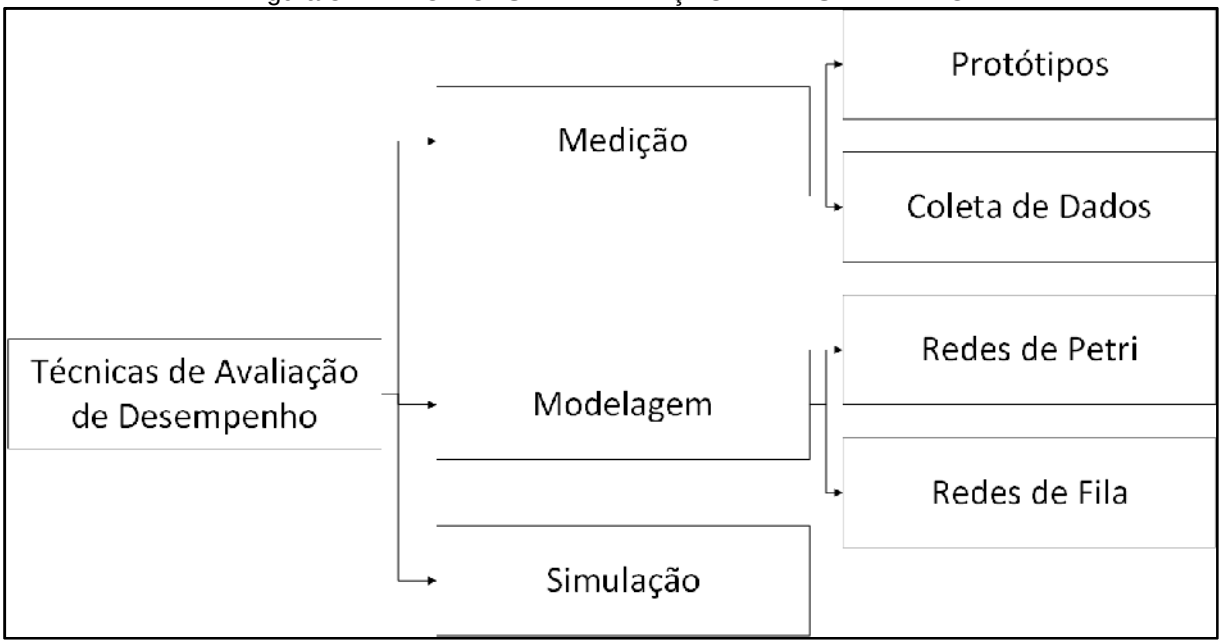
3.2 AVALIAÇÃO DE DESEMPENHO

A avaliação de desempenho surgiu entre os anos de 1970 e 1980 como um componente importante em Ciência da Computação. Para se obter conclusões sobre o desempenho e o comportamento em determinadas situações, é essencial avaliar o desempenho de sistemas computacionais. A avaliação de um sistema necessita de um conhecimento de todo o sistema, e também uma cuidadosa seleção de metodologia, ferramentas e carga de trabalho. Existem alguns conjuntos de técnicas, de modo geral, que podem ser agrupadas em[21]:

- Técnicas de Modelagem;
- Técnicas de Simulação;
- Técnicas de Medição.

Normalmente estes três conjuntos de técnicas são utilizados como suporte ao estudo de avaliação de desempenho em várias áreas, para fornecer dados para construção de modelos para representar ambientes reais,como também de avaliação de um sistema.A Figura 3.4 apresenta as técnicas de medição e modelagem mais utilizadas[21].

Figura 3.4 - TÉCNICAS DE AVALIAÇÃO DE DESEMPENHO



Para se medir o desempenho é necessário uma monitoração do sistema no mesmo tempo que ele está recebendo uma quantidade de *workloads* (cargas de trabalho). Alguns mecanismos são utilizados para geração decarga de trabalho como os*benchmarks que são* programas específicos de *workloads*. Esta escolha de qual mecanismo utilizar e quais as métricas que serão mensuradasé tão importante quanto que estratégia de medição deve ser escolhida. Por exemplo, pode-se buscar o tempo de processamento, acesso ao disco, dentre outros [22].

A avaliação de desempenho é uma técnica que permite quantificar a qualidade do serviço fornecido por um sistema computacional. A partir da quantificação dessa qualidade do serviço, é possível comparar várias opções de *hardware* a fim de definir aquela que será utilizada. Também é possível analisar se o sistema está se comportando conforme o esperado ou ainda, como o sistema irá se comportar a médio e longo prazo. A avaliação de desempenho pode ser utilizada para medir o desempenho do sistema computacional ou de alguns recursos específicos.

A avaliação de desempenho em sistemas computacionais torna-se algo necessário em todo o ciclo de vida dos mesmos. Os fabricantes de sistemas utilizam a avaliação de desempenho nas fases de projeto, fabricação, vendas, uso, atualização, entre tantas outras, tendo em vista conseguir sistemas computacionais

com o máximo de desempenho e com o mínimo de custo, objetivo esse que tem sido continuamente alcançado e que levou à proliferação das estações de trabalho e dos computadores pessoais, muitos dos quais com desempenho muito superior ao dos primeiros supercomputadores [22].

3.2.1 MEDIÇÃO

A medição é realizada através da observação constante de um sistema a fim de medir, no mundo real, como as métricas do sistema são afetadas pelas variações de fatores específicos. Medição de desempenho envolve a monitoração do sistema enquanto está sob a ação de uma carga de trabalho. A escolha da carga de trabalho é muito importante e alguns mecanismos são usados para geração de carga de trabalho como programas sintéticos, *benchmarks*[23].

Uma vantagem da medição é a pouca ou nenhuma perturbação ocasionada por eventos que ocorrem com pouca ou nenhuma frequência. A medição apresenta duas desvantagens para a avaliação de desempenho. A primeira e principal desvantagem é a necessidade da existência física do sistema que será avaliado. De um modo geral, esse fato traz como consequências: custo e tempo.A segunda desvantagem a ser levada em conta é a sensibilidade da técnica de medição à quantidade e representatividade das amostras de funcionamento que estão sendo consideradas. Para que essas sejam significativas, faz-se mister o uso de técnicas estatísticas.

As técnicas de medição podem ser divididas em:

- Protótipos – A construção de protótipos é, normalmente, utilizada em sistemas ainda não existentes, mas que se encontram em fase final de desenvolvimento [22]. Essa técnica consiste na simplificação do sistema computacional, em que são representadas somente suas características essenciais. Uma vantagem desta técnica é que pode-se observar algo concreto, mesmo em etapas iniciais do projeto. A grande desvantagem da construção de protótipos está relacionada aos altos custos para sua implantação;

- Coleta de Dados – As técnicas de medição, em geral, são as que fornecem os resultados mais precisos; em particular, a coleta de dados é a mais precisa. Seu objetivo principal é a avaliação de desempenho através da obtenção direta de dados em um sistema já existente[22]. A coleta de dados pode ser efetuada através de monitores. A função principal de um monitor é coletar dados referentes à operação do sistema. A atividade de monitoração deve ser realizada de modo que não afete a operação do sistema em questão. Os monitores podem ser divididos em monitores de *software*, *hardware* e híbridos[22]. Monitores de hardware são dispositivos de hardware específicos para obtenção e avaliação dos dados do objeto em estudo. Já os de softwares são usados para monitorar sistemas operacionais e também redes ou base de dados.

Técnicas de medição são aplicadas na avaliação de sistemas já existentes, ou em fase final de desenvolvimento [22]. Na medição de um sistema normalmente utilizam-se monitores de software, hardware ou híbridos, que permitem a coleta de dados reais a partir do funcionamento do sistema, fornecendo informações sobre o comportamento do ambiente em termos de desempenho.

No caso das técnicas de medição, deve-se ter muito cuidado para se evitar perturbações no sistema sob avaliação.Este é um dos grandes desafios existentes nessa técnica, e visa minimizar as interferências no resultado observado. Técnicas de medição são também necessárias quando se deseja monitorar um sistema, durante sua operação diária, como exemplo, à melhoria dos aspectos de escalonamento de processos.

3.2.2 MODELAGEM

Modelar um sistema consiste na construção de um modelo (uma representação literal, numérica ou simbólica), segundo algumas técnicas de construção de modelos, que abstrai características a serem analisadas do sistema real. No processo de modelagem, uma das tarefas mais complexas é o levantamento de quais elementos do sistemas serão incluídos no modelo. Assim, ao modelar um sistema, deve-se procurar abstrair quais elementos são importantes na

resolução de uma questão particular e quais relações entre esses elementos são necessárias à resolução do modelo. Dessa forma, pode-se ter vários tipos de modelos de um mesmo sistema, cada um com características e especificações mais adequadas à resolução de um problema específico.

Um importante fator para o desenvolvimento de um modelo é a definição da abordagem a ser utilizada para a representação do modelo. No caso de sistemas computacionais a modelagem pode ser realizada através de técnicas de modelagem que utilizam Cadeias de Markov[24], Redes de Filas[25], Redes de Petri[26].

Após a escolha da técnica utilizada para a representação do modelo, deve-se decidir qual a solução a ser dada. Uma técnica de solução disponível éa analítica. Na solução analítica, uma vantagem é que pode-se substituir os componentes do ambiente por um modelo matemático. Uma desvantagem é que nem sempre pode ser utilizada em virtude da complexidade de alguns modelos.

3.2.3 SIMULAÇÃO

A simulação é realizada através da reprodução do sistema utilizando alguma linguagem de programação ou algum ambiente de simulação. Esta abordagem é necessária para construir um modelo que simule o funcionamento do sistema a ser avaliado, modelo este que deve descrever as características funcionais do sistema em uma escala adequada de tempo. O modelo de simulação deve conter osdetalhes importantespara representação do sistema a ser avaliado, sem no entanto, ser necessário que haja a representação da totalidade de suas características, ou seja, há um certo nível de abstração nesse modelo. Essa abstração deve ser cuidadosamente planejada. Em comparação com a monitoração é menos dispendiosa, consome em geral menos tempo para que as medidas sejam obtidas, permite tantas repetições. de experimentos quantas forem necessárias e é bastante segura. Assim como na monitoração a quantidade e a representatividade das amostras consideradas é de fundamental importância para a obtenção de resultados corretos tornando indispensável uma análise estatística dessa amostragem.

Atualmente a simulação é uma das ferramentas disponíveis para avaliação de desempenho de sistemas computacionais e se tornou atrativa nas mais diversas áreas de aplicação devido a:

- Flexibilidade: é adaptável a novas e diferentes situações;
- Baixo custo: com um mesmo programa podem-se simular diferentes situações;
- Versatilidade: pode ser utilizada em diferentes ocasiões.

O processo de desenvolvimento de uma simulação envolve uma série de etapas. Deve-se primeiramente especificar o modelo, considerando as características mais importantes do sistema. Com esse sistema modelado e especificado, é necessário transformar o modelo em um programa de simulação. A modelagem e simulação de sistemas computacionais, em geral, podem ser divididas nas seguintes etapas [27], [28]:

- Definição do Problema – Definição de quais os problemas serão analisados e também os equipamentos e serviços disponíveis no sistema;
- Construção do Modelo - Tendo-se o sistema a ser simulado bem definido, é necessário representá-lo como um modelo o qualfornecerá uma visão particular do sistema. Um modelo deve ser de fácil compreensão, sem perda das características importantes e essenciais para avaliá-lo. É recomendado que o modelo, a princípio, seja simples, e que os detalhes sejam inseridos sistematicamente;
- Coleta de dados – Após gerar a descrição do modelo, há a necessidade de se definir quais os tipos dos dados serão coletados. Eles serão dos dados de entrada e servirão de parâmetros para o modelo, podendo ser tanto valores baseados em alguma análise preliminar;
- Refinamento dos dados – Assim que os dados forem coletados, é necessário refinar os dados da coleta de dados, para tornaros resultados mais fiéis;
- Validação – Na fase de validação deve-se demonstrar que o modelo proposto é uma representação do sistema a ser simulado, assim, o modelo conceitual deve substituir o sistema real;
- Análise dos Dados – Os resultados obtidos devem ser compilados, tabulados adequadamente e inseridos em gráficos, para serem analisados.

3.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma introdução sobre serviços *Web* e a sua importância, desde a sua criação até os dias atuais. Em seguida, apresentou conceitos essenciais para o entendimento dos dois serviços *Web* mais utilizados atualmente em escala global: REST e SOAP. Também foram apresentados conceitos básicos sobre avaliação de desempenho. Esses conceitos serão fundamentais para o entendimento da metodologia proposta para avaliação de desempenho de serviços *Web* que será descrita no capítulo seguinte.

4 ABORDAGEM PARA AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS *WEB*

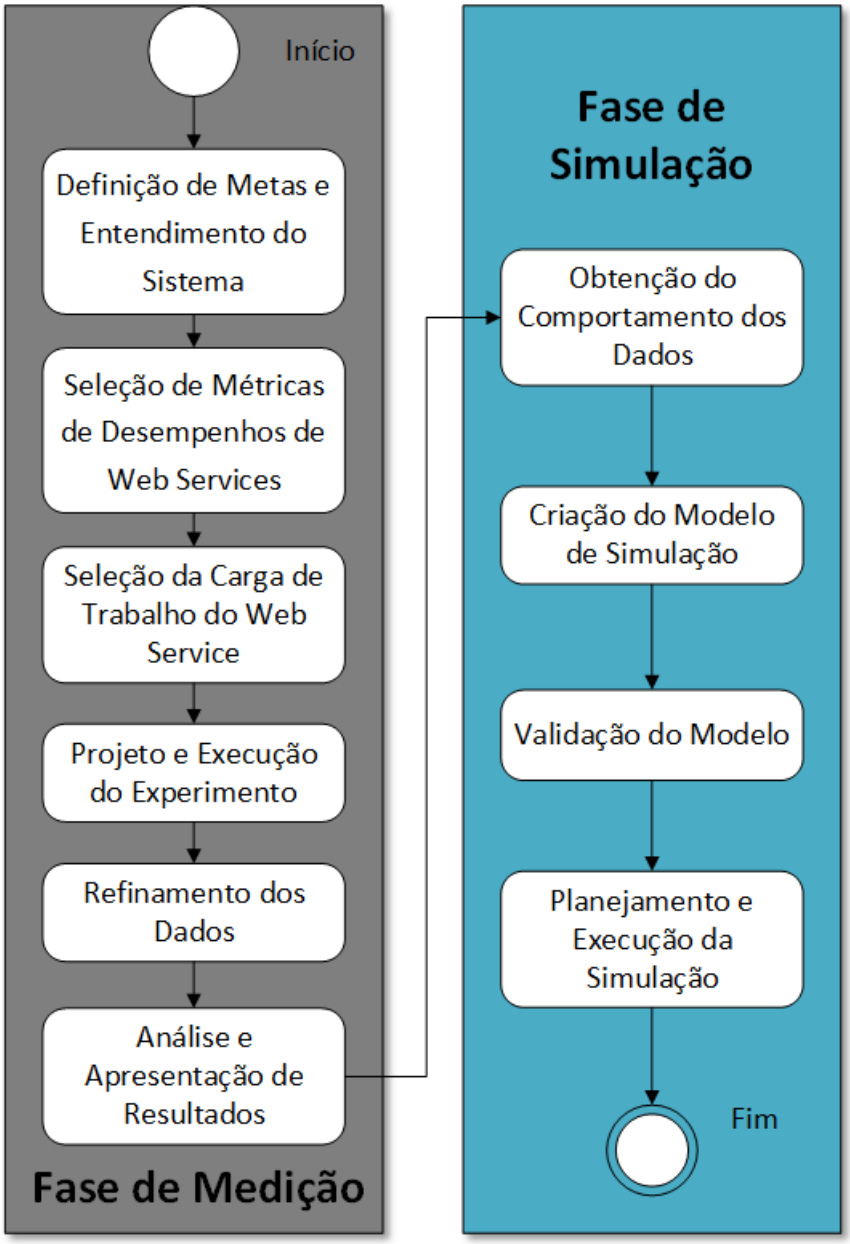
4 ABORDAGEM PARA AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS*WEB*

Este capítulo descreve aabordagemproposta para a avaliação de desempenho de aplicações baseadas em serviços *Web*. A necessidade de uma metodologia para este contexto é baseada no fato de que um número considerável de aplicativos está utilizando os serviços *Web* atualmente e a análise de aspectos de desempenhopode ajudar os analistas de negócios a escolherem quais configurações *(*tanto de *hardware* como *software)* devem ser utilizadas considerando umacarga de trabalho específica.

4.1 VISÃO GERAL

A abordagem proposta neste trabalho consiste em uma abordagem para a avaliação de desempenho de serviços *Web* e um modelo de simulação.A abordagem proposta foi dividida em duas fases: a primeira detalha as etapas de medição; já a segunda apresenta as atividades de modelagem juntamente com a simulação e a análise dos resultados. Como se pode observar na Figura 4.1, a fase de medição é necessária para a obtenção de dados a serem capturados de um sistema real. Além desta função, também é importante a calibração do modelo de simulação. Ainda na Figura 4.1, há a fase de simulação.Logo depois que o modelo é confeccionado e avaliado, pode-se testar configurações que não seriam possíveis com o ambiente real.

Figura 4.1 - METODOLOGIA PARA AVALIAÇÃO DE DESEMPENHO DE SERVIÇOS *WEB*



4.2 FASE DE MEDIÇÃO

A fase de medição da abordagem proposta é composta por 6atividades, que serão apresentadas nesta seção. Cada atividade é dependente uma da outra, como pode ser observado na Figura 4.1.

4.2.1 DEFINIÇÃO DE METAS E ENTENDIMENTO DO SISTEMA

Nesta atividade, o principal objetivo é a indicaçãodos objetivos do estudo e a definiçãosobre o que delinear no entendimento do sistema. Tudo isto tem que ser claramente definido no início da medição.

Algumas metas são comuns em processos de avaliação de desempenho e estão presentes em grande parte dos estudos de avaliação. Trabalhos que analisam o planejamento de capacidade de um servidor, e as análises de sobrecargas, têm que primeiramente definir as metas e entender o ambiente em que está sendo utilizado.

Dado o mesmo conjunto de *hardware* e *software*, a definição do sistema pode variar, dependendodos objetivos do estudo. A escolha do sistema afeta as métricas de desempenho, bem como as cargas de trabalho utilizadas para comparar os sistemas dependendo da análise. Este entendimento do sistema consiste em identificar alguns pontos da infraestrutura de *hardware* e de *software*. A seguir, são destacados exemplos comuns em estudos de desempenho envolvendo serviços *Web*:

- **Configuração da Máquina Cliente/Servidor –**Os componentes físicos do servidor e do cliente (ex: Processador, Memória);
- **Velocidade do HD –**A quantidade de rotações por minutos de um HD (ex: 5400RPM, 7200RPM);
- **Rede do Cliente/Servidor –**A velocidade da rede (ex: 1mb, 5mb, 10mb);
- **Sistema Operacional do Servidor –**O sistema operacional do servidor (ex: Windows, Linux, MacOS);
- **Servidor *Web*/Aplicação –**O tipo do servidor *Web*/aplicação (ex: Apache, Glassfish [19], ASP.NET);
- **Serviços *Web* –** O protocolo de serviços *Web* (ex: SOAP, REST, e XML-RPC).

Neste caso todos os *hardwares* e *softwares*citados acima devem ser bem definidos e selecionados nesta atividade da metodologia. Um estudo mais detalhado de cada ponto exposto a seguir para o melhor entendimento.

Configuração da Máquina Cliente/Servidor –É muito relevante ter esta definição exatamente nesta etapa da metodologia. A quantidade de memória que o

servidor tem é muito importante pois, se ele receber uma carga alta de requisições HTTP, por exemplo, os processos ficarão na memória em uma fila, aguardando chegar a vez, e se a memória não for grande o suficiente, eles terão que fazer um *swap* com o HD, fazendo com que o processo se prolongue ainda mais, pois a velocidade para acessar o HD é muito mais lenta em relação a memória RAM.

Velocidade do HD – Atualmente, grande parte dos *softwares* utilizam algum tipo de banco de dados para guardar as informações do sistema. Esta informação é relevante pois, a maioria dos serviços *Web* utilizam um banco de dados e implementam o CRUD. O termo RPM (rotações por minuto) é utilizado para ajudar a determinar o tempo de acesso em discos rígidos em um computador.É uma medida de quantas rotações o disco rígido de um computador faz em um único minuto. Quanto maior a RPM, mais rapidamente os dados serão encontrados; por exemplo, se fossem comparar dois discos rígidos, um com 5400 RPM e outro com 7200 RPM, o disco rígido com RPM 7200 seria capaz de acessar dados mais rapidamente do que o outro.

Rede do Cliente/Servidor - Falar em velocidade da rede significa enfatizar a capacidade de um serviço de conexão em enviar e receber centenas ou milhares de bits numa dada fração de tempo. Usualmente a medição se dá em “segundos”: *kilobits* por segundo, *megabits* por segundo ou*gigabits* por segundo são algumas das medidas de velocidade mais utilizadas atualmente. Este ponto é importante pois, dependendo da velocidade do servidor ou do cliente e do tamanho do arquivo que for ser enviado/recebido, vai existir uma diferença no tempo final da transmissão.

Sistema Operacional do Servidor – Neste caso, o que vai diferenciar a escolha são os *softwares* necessários para o teste. Existem *softwares* que só funcionam no Linux, outros só no Windows, tudo isso tem que ser analisado nesta fase do processo. Outro fator que é importante, é que no Linux por exemplo, pode-se escolher um servidor apenas com linha de comando, e também configurá-lo só com os softwares que necessitar. Já no Windows, há uma interface gráfica, que já consome uma parte do processamento, além de ter vários programas desnecessários instalados por padrão.

Servidor *Web*/Aplicação – Existem diversos servidores *Web* e de aplicações atualmente como o Apache[29], *Internet Information Services*(IIS)[30], Lightttpd[31], Nginx[32], G-WAN[33], *Glassfish*. A escolha vai depender da

necessidade do negócio. Pode-se escolher vários destes tipos de softwares e avaliar o desempenho de cada um, por exemplo. Também pode-se escolher apenas um deles, e fazer qualquer outro tipo de teste de carga.

Protocolo de comunicação do serviço *Web* – Dependendo de qual tipo de teste a ser realizado, esta escolha fica a critério das necessidades de cada projeto. Sendo assim, pode ser utilizado nos testes qualquer tipo de serviços *Web*, REST, SOAP ou XML-RPC. Tanto em teste relacionando um com o outro, quanto em um teste de um serviço *Web* isolado. No protocolo REST, pode-se escolher o tipo de invocação, que pode ser GET quando a informação é passada diretamente pela URL, ou POST quando a informação é encapsulada junto a requisição HTTP.

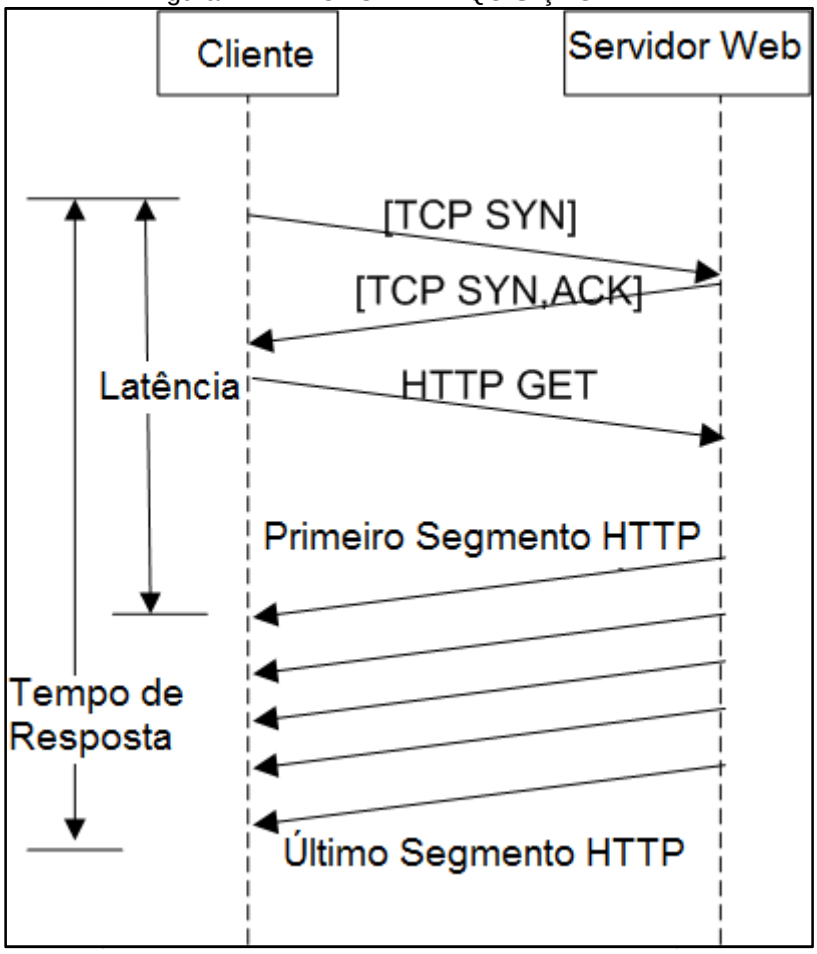
4.2.2 SELEÇÃO DE MÉTRICAS DE DESEMPENHO DE SERVIÇOS *WEB*

O próximo passo da metodologia é a seleção de critérios que servirão como base para o estudo de desempenho. Estes critérios são nomeados de métricas. Em geral, eles estão relacionados com a velocidade, precisão e disponibilidade dos serviços *Web*.

Em geral, quatro métricas são adotadas na avaliação de desempenho dos sistemas baseada em serviços. Estas métricas são detalhados a seguir:

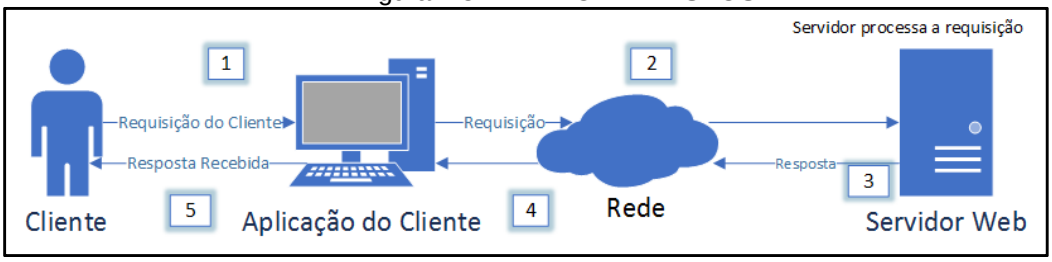
- **Latência -** no contexto de serviços *Web*,corresponde ao tempo necessário para a invocação do pedido, para o estabelecimento da conexão e para que a primeira parte do pacote da resposta chegue ao cliente [34]. Como pode ser visto na Figura 4.2, o Cliente faz o pedido de conexão do TCP, o servidor *Web* retorna o pedido, assim que a conexão é estabelecida, o Cliente pode enviar e receber os pacotes HTTP.

Figura 4.2 - FLUXO DA REQUISIÇÃO HTTP



- **Tempo de Resposta -** Essa métrica é definida como o intervalo entre a solicitação de um usuário e término da resposta do sistema[34]. Para ilustrar essa métrica, a Figura 4.3 é apresentada. No passo 1, o usuário executa a requisição inicial. Depois disso, no passo 2, o aplicativo do usuário envia o pedido para o serviço *Web* especificado. Em seguida, na etapa 3, o serviço*Web* recebe e processa a solicitação do usuário e envia a resposta para o aplicativo do usuário (passo 4). Finalmente, no passo 5, o usuário recebe a resposta requerida.

Figura 4.3 - TEMPO DE RESPOSTA



- **Utilização do processador -** é a porcentagem de tempo em que o servidor está ocupado com processos. Sendo assim, um servidor com apenas um núcleo, vai demorar mais para atender a quantidade de requisições simultâneas que um com múltiplos núcleos, já que este último pode processar simultaneamente por diferentes núcleos.Esta métrica é muito importante pois, pode-se fazer um cluster por exemplo, de computadores com múltiplos núcleos, fazendo com que as requisições sejam atendidas rapidamente.
- ***Throughtput* -** É definido como a taxa (pedidos por unidade de tempo) em que os pedidos podem ser atendidos pelo sistema. A fórmula seguinte pode ser utilizada para calcular a vazão expressa como clientes por segundo [9]:

$$\textit{Throughtput} = \frac{\text{Número de Clientes (pedidos)}}{\text{Tempo de resposta (unidade de tempo)}}$$

Para redes, o *throughtput* é medido em pacotes por segundo ou bits por segundo. No caso de Serviços *Web*, foi usado número de pedidos por milissegundos.

4.2.3 SELEÇÃO DA CARGA DE TRABALHO DO SERVIÇO *WEB*

A carga de trabalho é composta por uma lista de pedidos de serviço para o sistema. Por exemplo, a comparação de serviços *Web* pode ser constituída por um conjunto de pedidos. Dependendo da metodologia/tecnologia de avaliação escolhida, a carga de trabalho pode ser expressa de diferentes formas. Primeiramente é necessário determinar que tipo de característica é importante ser representada, a qual depende do objetivo da avaliação. Por exemplo, em um

sistema computacional é a ULA (Unidade Lógica e Aritimética) que executa a instrução logo a carga de trabalho neste caso seria a frequência e o tipo das instruções aritméticas.

Em se tratando de medição, de forma prática, a carga de trabalho geralmente consiste de *scripts* executados nos sistemas que representam uma quantidade específica de solicitações por segundo. Em todos os casos, é essencial que a carga de trabalho represente a utilização do sistema em cenários do mundo real.

4.2.4 PROJETO E EXECUÇÃO DO EXPERIMENTO

Um fator importante é como o experimento deve ser realizado, considerando tanto o cliente que envia o pedido quanto o servidor que processa. No lado do servidor é necessário fazer verificações em seus processos internos a fim de obter os resultados necessários, como nas métricas de tempo de resposta e utilização do processador.Além disso, tem-se que isolar o processo do servidor *Web* para que o resultado seja o mais fiel possível.

Outro ponto relevante nessa atividade é o projeto da própria execução da medição. Antes de executar os passos necessários, é importante definir, por exemplo, quantas interações devem ser executadas para a obtenção de um resultado estatístico válido e o tamanho da carga útil que deve ser enviado na rede.

Esta é sem dúvida uma das principais atividades da metodologia, que é exatamente a fase exatamente da coleta de dados. Sem um correto planejamento da medição, podem ser gerados dados/informações que podem não ser úteis ou suficientemente descritivas do ambiente que está sendo avaliado. Dependendo de qual métrica e carga de trabalho que serão utilizados, esta etapa foi criada para mostrar a execução do que foi descrito anteriormente. Se a métrica for tempo de resposta, por exemplo, pode-se executar o tempo de atendimento das requisições HTTP que simulam usuários acessando o sistema. Para fazer uma medição de solicitações HTTP, pode-se utilizar um software que gere cargas automáticas de requisições HTTP, como o JMeter [34], SoapUI, Fiddler, etc. Mais informações sobre esses softwares são apresentadas a seguir:

- **JMeter –** A aplicação Apache JMeter [34] é um software de código aberto projetada para carregar o comportamento funcional do teste e medir seu desempenho. Ele foi originalmente projetado para testes em *sitesWeb*, mas, desde então, expandiu-se para outras funções, como por exemplo, o teste do ambiente virtual de aprendizagem Moodle[35]. Apache JMeter pode ser usado para testar o desempenho tanto em recursos estáticos quanto dinâmicos. Pode-se utilizar em serviços *Web* (SOAP / REST), linguagens dinâmicas da *Web* - PHP, Java, ASP.NET, arquivos, entre outros. Ele pode ser utilizado para simular um grande número de carga em um servidor, grupo de servidores ou rede. Essa ferramenta pode ser usada para fazer uma análise gráfica do desempenho ou para testar o seu comportamento.
- **SoapUI–**SoapUI é um *framework* para simplificar o teste de aplicações *Web* e serviços *Web*. Os casos de teste podem ser inseridos usando uma interface gráfica do cliente. Emum ambiente de testeindividual,SoapUIfornececobertura de testecompleta esuporta protocolos etecnologias padrão. Esta ferramenta suporta os tipos de serviços *Web*, REST e o SOAP;
- **Fiddler[36] –** O Fiddlergravatodo otráfego HTTP e HTTPSque passa entreo computador e aInternet.Fiddler tambémcaptura o tráfegodetodos os processos que estão sendo executadoslocalmente, servidor-a-servidor (por exemplo,*Web* Services) e o tráfego de dispositivo-a-servidor (por exemplo,clientesde telefonedo iPad eWindows).

Para medir o desempenho do processador, e dos componentes internos do servidor como o uso de memória RAM, *CACHE*, acessos ao HD, etc, pode-se utilizar as seguintes ferramentas:

- **Windows Performance Monitor –** Esta ferramenta pode serusada para examinar como os programas afetam o desempenho do computador e coletar dados de *log* para uma análise posterior. O *Windows Performance Monitor*[37] usa contadores de desempenho, dados de rastreamento de eventos e informações de configurações, que podem ser combinadas em conjuntocom coletores de dados. Contadores de desempenho são medições quanto ao estado ou atividade do sistema, eles podem ser incluídos no sistema operacional ou podem ser parte de aplicações individuais. *Windows Performance Monitor* solicita o valor atual dos contadores de desempenho em

determinados intervalos de tempo. Dados de rastreamento do evento são coletados a partir de provedores de rastreamento que são componentes do sistema operacional ou de aplicativos individuais. Estes relatam ações ou eventos. A saída de vários provedores de rastreamento pode ser combinada em uma sessão de rastreamento. As informações de configuração são coletadas de valores de chave no registro do Windows. Essa ferramenta pode gravar o valor de uma chave de registro em um determinado momento ou intervalo como parte de um arquivo de log.

- **Java MissionControl/Flight Recorder[38] –** Java *Flight Recorder* e *MissionControl* juntos representam um conjunto de ferramentas para coletar continuamente o tempo de execução do experimento e prover informações detalhadas que permitam a análise após a execução dos experimentos. Java *Flight Recorder* é uma estrutura de perfis e coleta de eventos embutida no JDK da Oracle. Ele permite que administradores e desenvolvedores Java reunam informações detalhadas de baixo nível sobre como o *Java Virtual Machine* (JVM) e o aplicativo Java estão se comportando. *Java MissionControl* é um conjunto avançado de ferramentas que permitem a análise eficiente e detalhada da extensa quantidade de dados coletados pelo Java *Flight Recorder*. Essas ferramentas permitem que os desenvolvedores e administradores coletem e analisem os dados de aplicativos Java executados localmente ou implantados em ambientes de produção.

4.2.5 REFINAMENTO DOS DADOS DA MEDIÇÃO

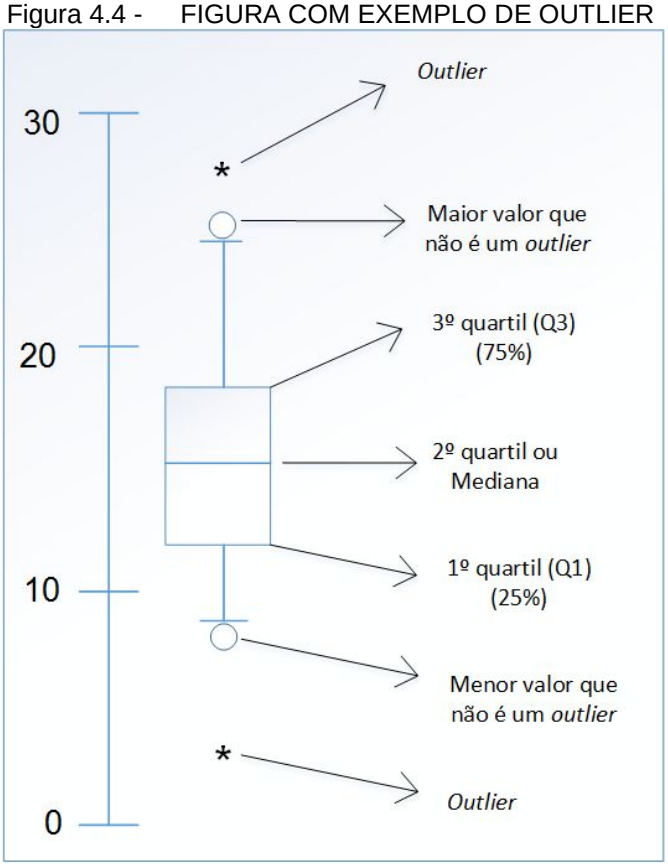
Um *outlier* é uma observação que se encontra a uma distância anormal de outros valores em uma amostra aleatória de uma população. Mesmo com essa definição é necessário um analista para decidir o que será considerado anormal. Antes das observações anormais serem apontadas, é necessário caracterizar as observações normais. Estatisticamente, um *outlier* é um erro cuja ocorrência é fruto de uma variável aleatória e é altamente improvável [39].

É importante detectar e observar a sua quantidade e, se necessário, retirá-los do conjunto de dados. Estes *outliers* contaminam a distribuição de probabilidade dos erros, e termina invalidando as hipóteses de distribuição, quando não são eliminados previamente.

Para detectar estes erros, podem-se utilizar alguns métodos de identificação como Gráfico de Box (Box-Plot), Modelos de discordância, Teste de Dixon, Teste de Grubbs, Z-scores, entre outros[40].

O gráfico de Box-Plot pode ser observado na Figura 4.4 e é construído da seguinte forma: calcula-se a mediana, o quartil inferior (Q1) e o quartil superior (Q3); Subtrai-se o quartil superior do quartil inferior = (L). Os valores que estiverem no intervalo de Q3+1,5L e Q3+3L e no intervalo Q1-1,5L e Q1-3L, serão considerados outliers podendo, portanto ser aceitos na população com alguma suspeita; Os valores que forem maiores que Q3+3L e menores que Q1-3L devem ser considerados suspeitos de pertencer à população, devendo ser investigada a origem da dispersão. Estes pontos são chamados de extremos[40].

Figura 4.4 - FIGURA COM EXEMPLO DE OUTLIER



4.2.6 ANÁLISE E APRESENTAÇÃO DE RESULTADOS

A etapa final de um estudo de desempenho via medição é analisar e apresentar os resultados aos usuários interessados. A análise dos resultados pode ser através de geração de estatísticas como a média, desvio-padrão, mediana, etc. Gráficos, tais como gráficos de linhas, gráficos de barras, gráficos de pizza e histogramas são comumente usados em resultados de desempenho. O gráfico também é uma boa maneira de enfatizar ou esclarecer um ponto, para reforçar uma conclusão, e para resumir os resultados de um estudo.

Considerando experimentos focados em serviços *Web*, é importante lembrar as métricas relevantes que já foram citadas neste trabalho: tempo de resposta, utilização do processador e taxa de transferência. A maioria das análises estatísticas e de avaliação, baseadas em suas métricas,podem apoiar o usuário no processo de negócios e na tomada de decisão.

Algumas análises devem ser realizadas com os dados gerados da medição como tirar a média dos dados, a mediana, achar o desvio padrão, pois todas estas informações geram informações que podem ser inseridas em gráficos.

4.3 FASE DE SIMULAÇÃO

Na fase de Simulação abordam-se os conceitos e técnicas utilizadas para a criação e validação do modelo. Através deste modelo, é possível utilizar simulação para gerar dados e informações que poderão ser usadas para fazer uma avaliação de desempenho mais completa do sistema.

Os dados que foram capturados na fase de medição, servem de entrada para esta fase de Simulação. Todas as informações necessárias para o entendimento desta fase estão divididas em 4 etapas, as quais serão abordadas nas próximas seções.

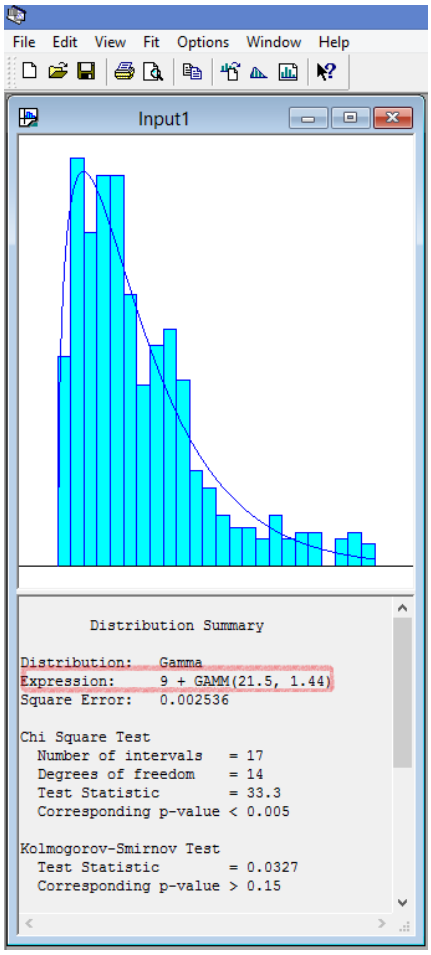
4.3.1 OBTENÇÃO DO COMPORTAMENTO DOS DADOS

O objetivo desta atividade é determinar a curva de distribuição teórica de probabilidade que melhor reproduz o comportamento do sistema. Os dados que serão utilizados nesta etapa foram obtidos na fase de medição. Outros dados estão distribuídos em torno de um valor médio, de acordo com uma distribuição estatística. As etapas básicas para se obter a curva de distribuição são:

- Processo de amostragem e coleta dos dados;
- Tratamento dos dados;
- Identificação da distribuição estatística.

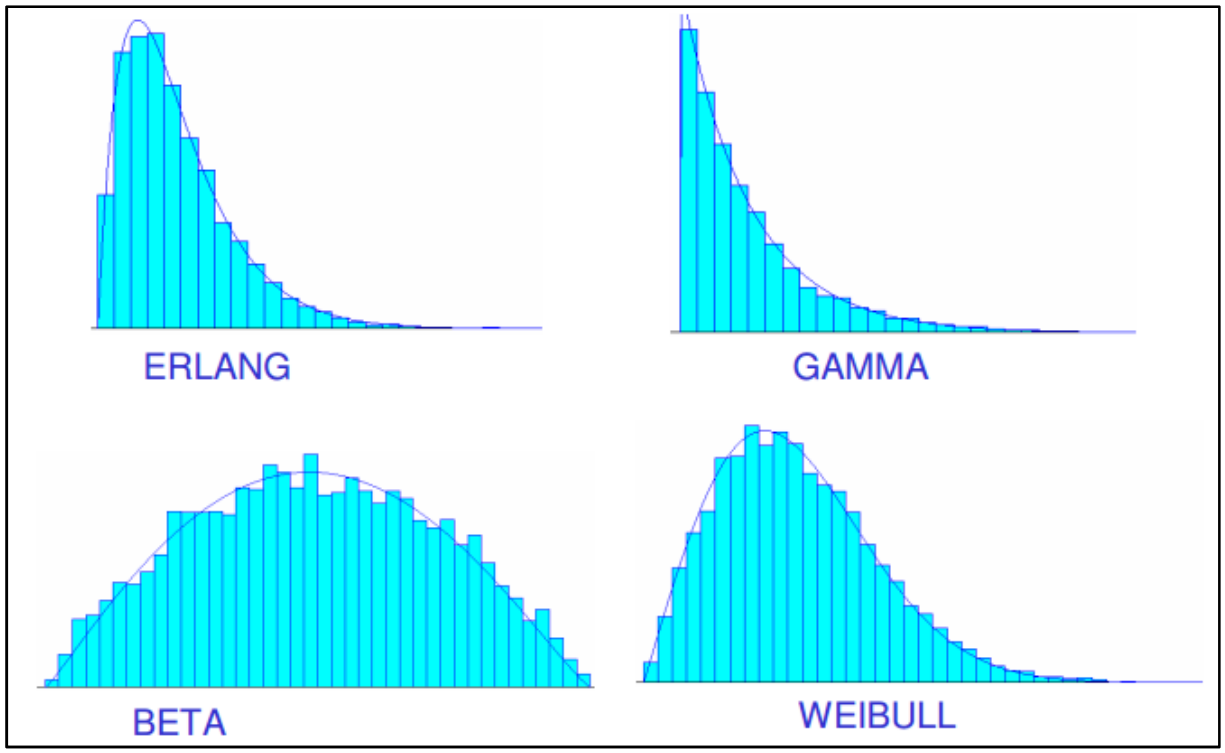
A etapa do processo de amostragem e coleta dos dados, foi obtida na última etapa da fase de medição. Após a obtenção destes dados, foi utilizado o tratamento de dados através da retirada dos outliers. Logo após estas etapas, é necessário identificar a distribuição estatística dos dados. Estas última etapa pode ser automatizada utilizando um software específico para este fim, como por exemplo o *Input Analyzer*, que é um software integrante do Arena[41]. Esta ferramenta fornece ajustes estatísticos para um conjunto de dados experimentais, como os dados deentrada, de tempo de processo, entre outros. Na Figura 4.5, ilustra-se a distribuíção de probabilidade indicada pelo *Input Analyzer*, os parâmetros dessa distribuição e o erro quadrático associado a essa expressão.

Figura 4.5 - EXEMPLO DO INPUT ANALYZER



A partir de um conjunto de dados históricos medidos de um sistema real, o *Input Analyzer* determina a melhor função e determina a curva de distribuição teórica de probabilidade que melhor reproduz o comportamento do sistema.As distribuiçõesestatísticas usadas em modelagem de processos são diversas, alguns exemplos de distribuições são a Erlang, Gama, Beta e Weibull.

Figura 4.6 - EXEMPLO DE DISTRIBUIÇÕES ESTATÍSTICAS



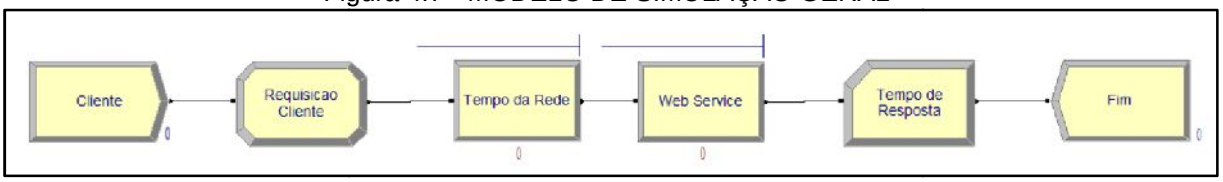
Pode-se notar como os dados se distribuem de forma distinta nas funções citadas. Cada tipo de distribuição da Figura 4.6 foi obtida de um conjunto de dados diferentes, e com isso foi gerada a melhor curva de distribuição teórica de cada conjunto de dados.

4.3.2 CRIAÇÃO DO MODELO DE SIMULAÇÃO

Assim que o sistema a ser simulado está bem definido, é necessário representá-lo por meio de um modelo, o qual irá representar uma visão particular do sistema. Sem que ocorram perdas significativas das características essenciais e importantes para a avaliação do desempenho, o modelo deve ser de fácil compreensão. A quantidade de informações incluídas deve variar de acordo com o propósito da construção do modelo. É recomendado que o modelo seja simples e que os detalhes sejam inseridos sistematicamente até que a complexidade desejada seja alcançada.

Com as ferramentas e ambientes computacionais hoje disponíveis, é possível construir modelos de simulação em computadores,mesmo com conhecimento restrito sobre linguagens e ferramentas de simulação, bem como sobre toda a lógica de programação e a matemática envolvida nesses programas. O mais importante é que o usuário tenha domínio sobre a natureza do sistema e do problema a ser tratado [11]. Na Figura 4.7, pode-se observar um exemplo de um modelo de simulação feito por a ferramenta de simulação, Arena [46].

Figura 4.7 - MODELO DE SIMULAÇÃO GERAL



A simulação computacional utiliza-se de modelos computacionais para esse propósito. Estes modelos computacionais voltados à simulação de sistemas dependem, fundamentalmente, da natureza do sistema a ser estudada. Sistemas podem ser definidos como “um conjunto de objetos, entre os quais se podem encontrar ou definir algum tipo de relação, que atuam e interagem (cooperam) com a intenção de alcançar um objetivo ou propósito lógico” (TAYLOR apud [11]). Para o propósito desta abordagem, podem ser definidos sistemas como um grupo de componentes que recebem estímulos ou entradas (*inputs*) e produzem respostas ou saídas (*outputs*). São os componentes e suas relações, tanto internas quanto externas ao próprio sistema, que determinam como este pode converter estímulos em respostas [11].

Para a criação do modelo de simulação, existem várias ferramentas que auxiliam este processo. Alguns exemplos de softwares de simulação: ARENA, OMNeT++ [42], etc.

4.3.3 VALIDAÇÃO/AVALIAÇÃO DO MODELO

Nesta fase de Validação/Avaliação deve-se demonstrar que o modelo proposto é uma representação do sistema a ser simulado, e deve responder a seguinte pergunta: “O modelo conceitual pode substituir, com um grau de erro aceitável, o sistema real?”. A avaliação pode ser considerada em dois casos: quando

o sistema modelado não existe, tendo-se apenas o projeto do sistema a ser simulado, ou quando o sistema simulado existe e pode ser medido. No primeiro caso, o objetivo da avaliação é estimar o desempenho do projeto ou mesmo avaliar projetos alternativos. Neste caso, o modelo pode ser avaliado baseando-se nos resultados esperados do projeto do sistema e cada abstração e suposição são justificadas. No segundo caso, o objetivo da análise é avaliar uma mudança proposta no sistema, e a avaliação é baseada na comparação dos resultados do modelo com as medidas reais do sistema [27], [28].

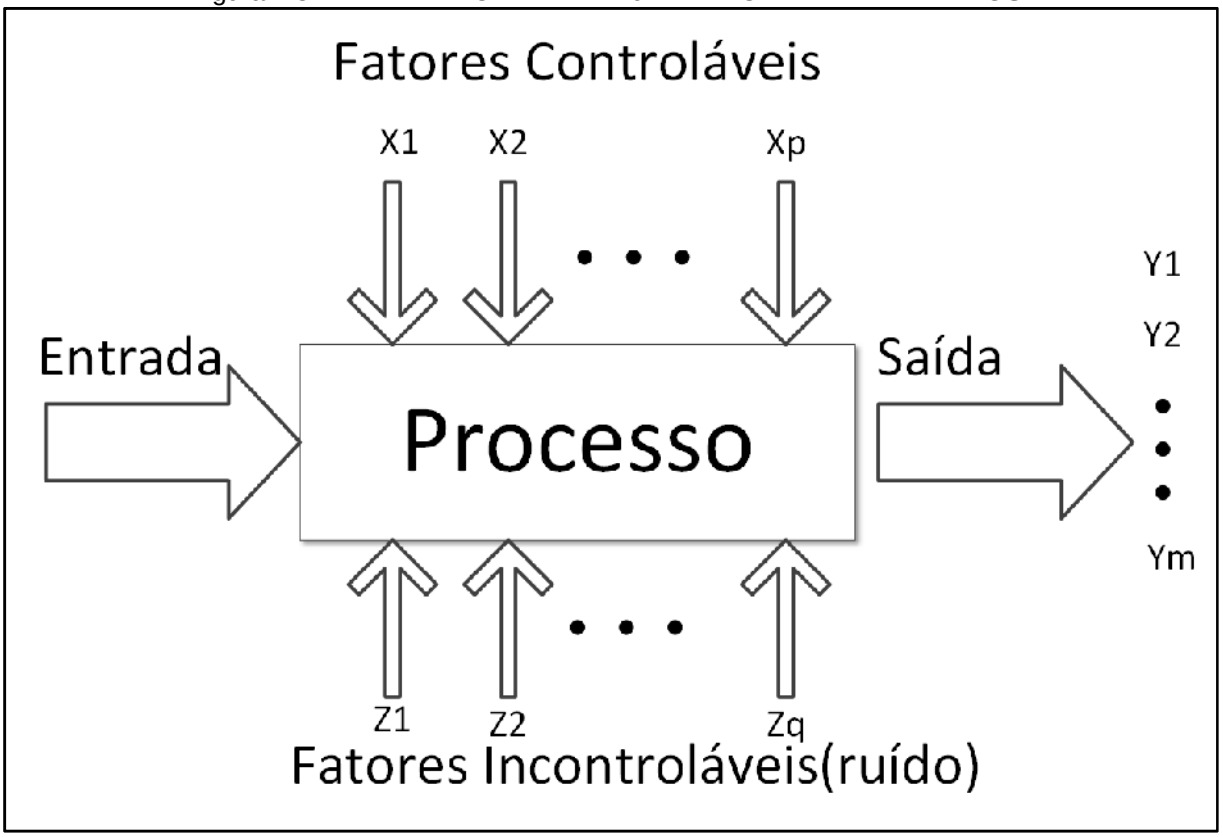
Embora as medições obtidas em sistemas reais seja o método mais confiável e preferido na validação de um modelo, nunca deve-se esquecer o fato de que se estará comparando observações realizadas em sistemas distintos, um é o real e o outro é simulado com base no real.

4.3.4 PLANEJAMENTO E EXECUÇÃO DA SIMULAÇÃO

Design ofExperiments (DOE) é um método sistemático para determinar a relação entre os fatores que afetam um processo e a saída do referido processo. Em outras palavras, ele é usado para encontrar relações de causa e efeito. Esta informação é necessária para gerir entradas do processo, a fim de otimizar a produção. É uma técnica muito importante para a indústria pois, seu emprego, permite resultados mais confiáveis economizando dinheiro e tempo. Já que a técnica de Planejamento de Experimentos requer uma quantidade exaustiva de cálculos, é fundamental o emprego de ferramentas para gerar os resultados.

Na Figura 4.8, pode-se notar como o planejamento de experimentos funciona. Primeiramente tem-se como exemplo um sistema computacional, neste caso os fatores controláveis poderiam ser, quantidade de memória, quantidades de discos rígidos, velocidade da rede, etc. Já nos fatores incontroláveis, pode-se utilizar a interferência que ocorre na rede. Como saída, tem-se os resultados da junção dos fatores controláveis e os incontroláveis.

Figura 4.8 - EXEMPLO DE PLANEJAMENTO DE EXPERIMENTOS



O experimento projetado ou planejado é um teste ou uma série de testes nos quais se induzem mudanças deliberadas ou estímulos nas variáveis de entrada do processo ou sistema, de tal forma que seja possível observar e identificar os efeitos nas respostas ou nas variáveis de saída.

Uma compreensão do DOE primeiramente requer o conhecimento de algumas ferramentas estatísticas e conceitos de experimentação. Apesar de um DOE poder ser analisado em vários *softwares*, é importante para os profissionais entenderem conceitos básicos do DOE para sua aplicação adequada. As técnicas de planejamento e análise de experimentos são utilizadas basicamente para melhorar as características de qualidade dos produtos ou processos de fabricação, reduzir o número de testes e otimizar o uso de recursos. O planejamento de experimentos descreve que esse objetivo geral pode ser dividido em outros objetivos secundários:

- Identificar as variáveis (fatores de controle) do processo que mais influem nos parâmetros de resposta de interesse;

- Atribuir valores às variáveis influentes do processo de modo que a variabilidade da resposta de interesse seja mínima;
- Atribuir valores às variáveis influentes do processo de modo que o efeito das variáveis não controláveis seja reduzido.

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a metodologia e os métodos que compõem a solução integrada para a avaliação de desempenho de serviços *Web*. Para ilustrar e avaliar o que foi proposto, no próximo capítulo será apresentado um estudo de caso mostrando na prática todos os passos descritos.

5 ESTUDO DE CASO E AVALIAÇÃO

5 ESTUDO DE CASO E AVALIAÇÃO

Este capítulo se destina a apresentar um estudo de caso com o intuito de ilustrar a aplicação da metodologia proposta neste trabalho e demonstrar a sua viabilidade. Para facilitar essa demonstração, este estudo de caso foi desenvolvido e será apresentado/avaliado seguindo exatamente o fluxo de atividadesproposto na metodologia.

5.1 ESTUDO DE CASO

O estudo de caso visa, principalmente, mostrar a adequabilidade da proposta deste trabalho, relacionada a abordagem para avaliação de desempenho, e sua explicação ao contexto deste estudo de caso. Implementada especificamente para a utilização nesse trabalho, o cenário é composto por dois serviços *Web*, e cada um deles contém um tipo de serviço *Web*, um com o SOAP e o outro com o REST.

Assim como a abordagem proposta, o estudo de caso foi dividido na aplicação das duas fases da metodologia apresentada no capítulo anterior, a fase de medição e a de simulação.

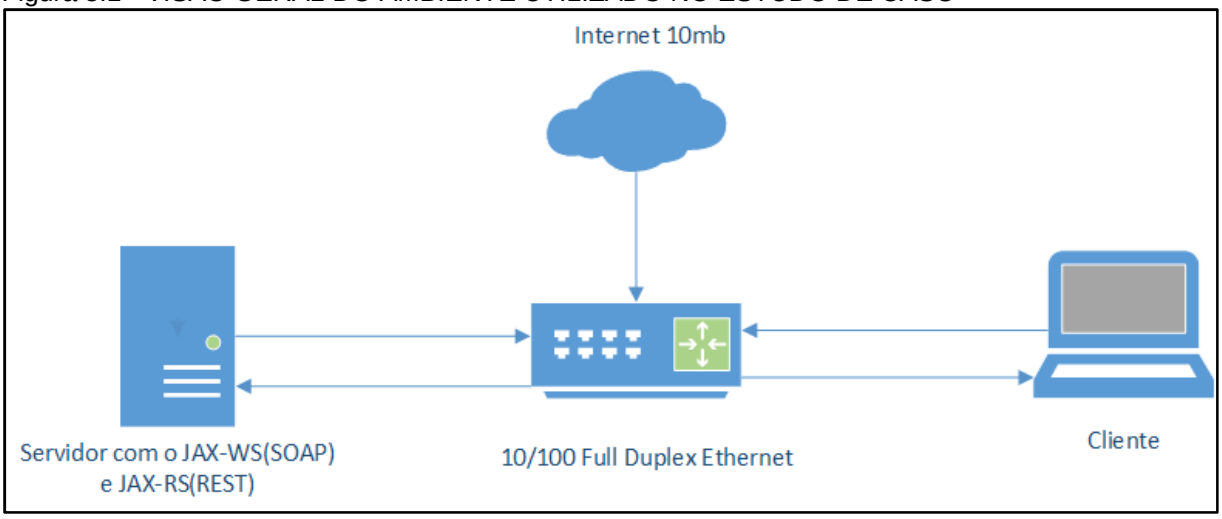
5.1.1 DEFINIÇÃO DE METAS E ENTENDIMENTO DO SISTEMA

Em primeiro lugar, devem ser definidosos objetivos do estudo e escolhido o que delinear no entendimento do sistema. Pode-se afirmar que os dois protocolos (SOAP e REST) são seguramente os protocolos de comunicação mais usados, e um dos fatores para a escolha destes é a comparação do desempenho de serviço *Web*. Dependendo do seu objetivo, pode-se utilizar um estudo apenas de um protocolo, por exemplo, e identificar as métricas que desejar. O objetivo deste estudo de caso é comparar o desempenho dos dois protocolos de comunicação de serviços *Web*, SOAP e REST, a fim de confrontar os seus resultados.

Na Figura 5.1 é apresentada a visão geral do estudo de caso. Neste estudo, dois serviços *Web*, o REST e o SOAP, foram instalados em um servidor. Ainda no servidor, as ferramentas de monitoramento Windows Performance Monitor, Java

MissionControl e Wireshark foram instaladas. A infraestrutura ilustrada na Figura 5.1mostra quando os pedidos são enviados através da rede a partir de um computador do cliente para o servidor *Web*.

Figura 5.1 - VISÃO GERAL DO AMBIENTE UTILIZADO NO ESTUDO DE CASO



A escolha do sistema afeta as métricas de desempenho, bem como as cargas de trabalho utilizadas para comparaçãodos sistemas conforme o cenário analisado. As informações sobre o ambiente utilizado no estudo de caso estão descritas abaixo:

- **Configuração Física da Máquina Servidor –** Processador Intel® Core ™ 2 Duo CPU @ 2.33GHz 2.33GHz, Memória RAM 5,00GB, Sistema Operacional Windows 7 64bits, HD 120GB 5400 rotações por minutos;
- **Configuração Física da Máquina Cliente -**Intel® Core ™ i7-4510U CPU @ 2.00GHz 2.60GHz, Memória RAM 8,00GB, Sistema Operacional Windows 8 64bits, HD 1TB 7200 rotações por minutos;
- **Rede do Servidor/Cliente –** 10mb;
- **Servidor *Web*/Aplicação –**Glassfish 4.1 [41];
- **Serviços *Web* –** SOAP, REST.

5.1.2 SELEÇÃO DE MÉTRICAS DE DESEMPENHOS DE SERVIÇOS *WEB*

Como visto na metodologia proposta, existem várias métricas que podem ser utilizadas para avaliação de serviços *Web*. Neste estudo de caso, foram utilizadas as seguintes métricas:

- Tempo de Resposta;
- Utilização do Processador;
- *Throughtput.*

A fim de medir o tempo de resposta e a latência dos serviços *Web*, existem várias ferramentas para executar esta operação, tal como o JMeter, SOAPUI, que já foram anteriormente apresentadas no Capítulo 3. Considerando os objetivos do estudo de caso, o JMeter foi adotada porque suporta tanto requisições REST como SOAP, além de ser *Open Source*. Para capturar a utilização do processador no lado do servidor, é usado o *Windows Performance Monitor* e o Java *MissionControl*.Nestas duas ferramentas foram extraídos os *logs* de acordo com o tempo que o servidor estava recebendo a carga de requisições HTTP.

Outra métrica que pode-se obter é o *Throughtput*que é definido como a taxa (pedidos por unidade de tempo) em que os pedidos podem ser atendidos pelo sistema. No caso de serviços *Web*, foi usado o número de pedidos dividido pelo tempo em milissegundos para se obter o *throughtput*.

5.1.3 SELEÇÃO DA CARGA DE TRABALHO DO SERVIÇO *WEB*

Para a avaliação proposta neste estudo de caso, uma das etapas importantes é a escolha da carga de trabalho a ser usada. Neste contexto, testes foram desenvolvidos com ferramentas de teste de cargas específicas, mais precisamente foi utilizada a ferramenta JMeter[34]. Nestes testes, os pedidos foram feitos com 8112 *bytes* cada, pois se o tamanho do pedido fosse muito pequeno, seria difícil capturar a latência e o tempo de resposta, porque na primeira resposta do servidor já traria a resposta completa. Neste caso, cada pedido é um usuário diferente, fazendo uma requisição no serviço *Web*. Os parâmetros escolhidosrelacionados a carga de trabalho foram:

a) Tipo de Protocolos - Neste caso, foram utilizados os dois protocolos de comunicação para serviços *Web* mais utilizados, REST e SOAP;
b) Tempo entre chamadas sucessivas – Neste estudo de caso, todas as requisições foram enviadas ao mesmo tempo para o servidor.
c) Número e tamanhos dos parâmetros de chamada – A quantidade de requisições mínima e a máxima foram respectivamente 100 e 500 requisições. O tamanho das amostras foi de 8112 *bytes* cada.
O fator chave escolhido para este estudo foi número de chamadas consecutivas e a comparação dos protocolos de comunicação de serviços *Web* SOAP e REST.

5.1.4 PROJETO E EXECUÇÃO DO EXPERIMENTO

Em primeiro lugar, é importante registrar as condições em que este estudo foi executado. Os dois serviços *Web* (REST e SOAP) foram instalados no mesmo servidor com as configurações apresentadas na Seção 4.1.1.

A linguagem de programação utilizada no script foi a linguagem Java. Para executar o experimento, o servidor de aplicação de código aberto *GlassFish* foi instalado. No estudo de caso, foi utilizada a versão do *NetBeans* IDE 8.0.2[43] sem Glassfish, pois ele já tinha sido instalado separadamente. Após a instalação destes softwares (Netbeans, Glassfish), foram instalados os dois serviços *Web*, a API Java para XML Serviços *Web* (JAX-WS) para SOAP e API Java para RESTful serviços *Web* (JAX-RS) para o REST.

O método utilizado neste estudo de caso foi o *POST* pois, normalmente é o mais utilizado para a criação de novos recursos. POST não é seguro nem idempotente, portanto é recomendável para solicitações de recursos não idempotentes. O método GET é usado para recuperar (ou ler) uma representação de um recurso.

No JAX-WS, uma operação chamada de serviços *Web* é representada com um protocolo baseado em XML, como SOAP. A especificação SOAP define o envelope da estrutura, regras de codificação, e convenções para a representação de chamadas de serviços *Web* e suas respostas. Essas chamadas e respostas são transmitidas como mensagens SOAP (arquivos XML) sobre HTTP. Apesar de mensagens SOAP serem complexas, a API JAX-WS esconde esta complexidade.

No lado do servidor, o desenvolvedor especifica as operações de serviços *Web*, definindo métodos em uma interface escrita na linguagem de programação Java. Logo após a criação foi feita a configuração do serviços *Web* com *Glassfish*. Com o REST também foi feita uma criação de um projeto Java, mas foi usado o JAX-RS, que é específico para REST. Já do lado do cliente, para testar se os serviços *Web* estavam funcionando corretamente, foi realizado um pedido diretamente pelo navegador da *Web* com o ip do servidor e com o endereço correto de cada serviço *Web*.

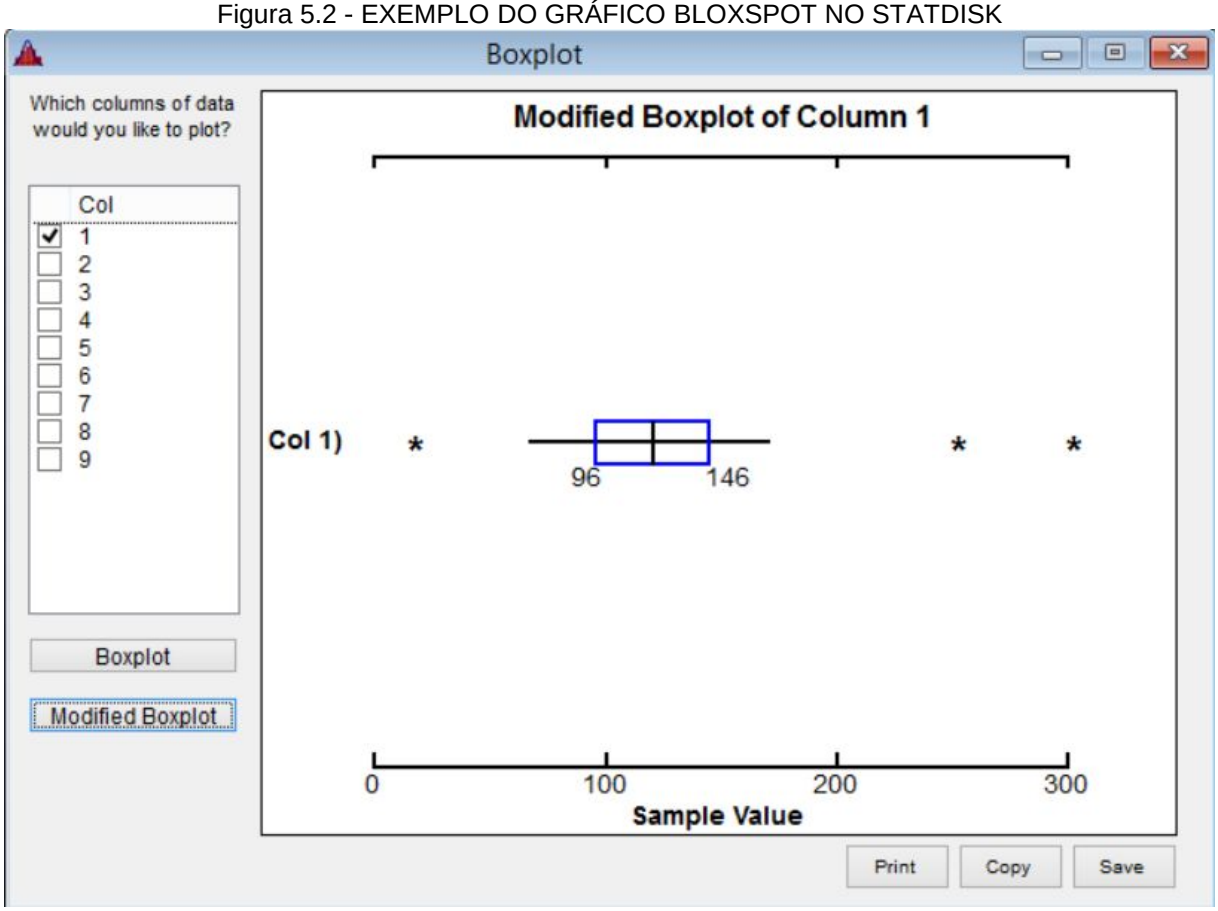
Antes de todos os testes serem realizados, o servidor foi isolado com apenas o processo do serviço *Web*, e a prioridade do processo foi modificada de “normal” para “tempo real”, o que significa que o processador vai dar prioridade máxima a este processo. Para identificar a quantidade de carga máxima que seria alcançada, foram feitos vários testes para chegar no número máximo de 500 requisições. Testes com pouca carga de trabalho foram feitos e observados, mas já que seria uma rajada de requisições simultânea, teriam que começar em no mínimo 100 requisições. Feita esta constatação, também foi observado que cada requisição, ocupava uma grande parcela na memória, sendo assim quando inserímos mais que 500 requisições, os resultados não seguiam o mesmo padrão das amostras anteriores (as amostras normais eram entre 35 a 150ms, com mais de 500 requisições passavam de 2000ms). Quando o cliente enviava mais do que as 500 requisições, os 8Gb de memória RAM da máquina do cliente eram esgotados fazendo com que fosse feito swap com o HD. Além disso foram feitas comparações entre os dois serviços *Web* SOAP e REST, com o número de requisições 100, 200, 300, 400 e 500, respectivamente.

5.1.5 REFINAMENTO DOS DADOS DA MEDIÇÃO

A importância de detectar e observar a quantidade de outliers já foi mencionada em detalhes no capítulo anterior. Nesta etapa será apresentado na prática como foi utilizada a remoção dos outliers dos dados coletados.

Assim que os dados da medição foram obtidos, todos os resultados foram inseridos na ferramenta *Statdisk*[44]. Nesta ferramenta, pode-se utilizar o gráfico *bloxspot*, onde são mostrados todos os outliers dos dados que foram coletados. Na Figura 5.2, observa-se que os dados que estão aproximadamente acima de 150 e abaixo de 20, têm boa probabilidade de serem *outliers*. Desta forma, todos esses dados com alta probabilidade de serem *outliers* foram retirados para termos uma análise mais acurada.

Figura 5.2 - EXEMPLO DO GRÁFICO BLOXSPOT NO STATDISK



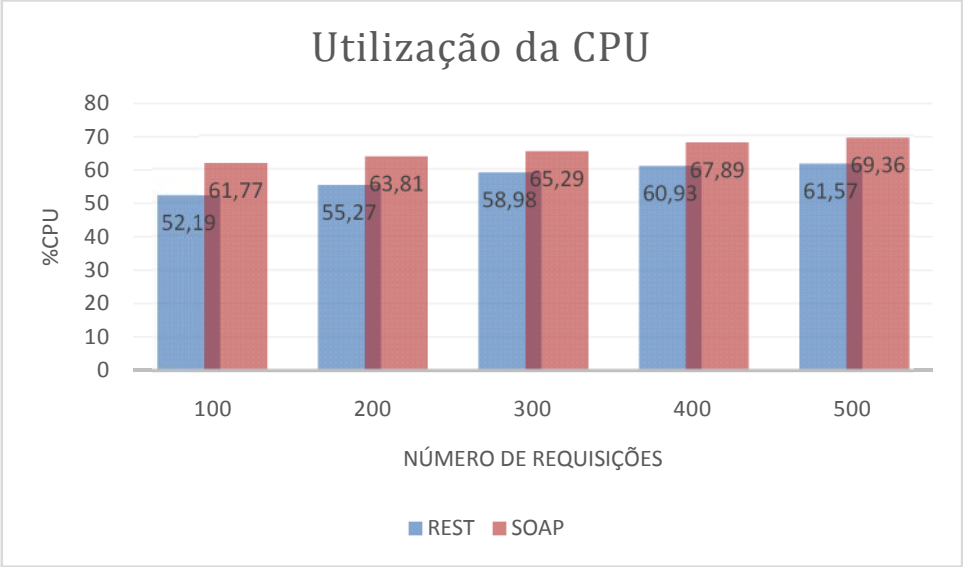
.

5.1.6 ANÁLISE E APRESENTAÇÃO DOS RESULTADOS

Inicialmente, a Figura 4.3 apresenta a média da utilização da CPU, expressa em termos percentuais. Para medir a utilização da CPU, foi utilizado o *Windows Performance Monitor* que fornece informações completas de todos os processos que estão em execução, a utilização da memória RAM, a utilização do disco, etc. É

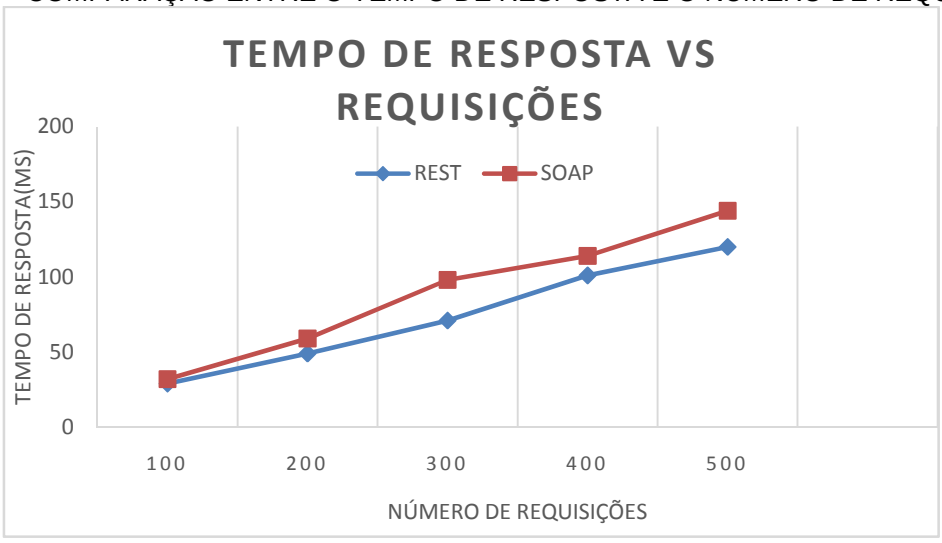
relevante ressaltar que, na Figura 5.3, existe sempre uma diferença entre os resultados de desempenho da CPU considerando os protocolos REST e SOAP. Dentre várias razões que podem ser apontadas para este fato, uma delas reside no fato de que o cabeçalho da requisição SOAP é maior se comparado ao REST; consequentemente, ele envia mais dados para o servidor processar.

Figura 5.3 - NÚMERO DE REQUISIÇÕES VS UTILIZAÇÃO DE CPU



A Figura 5.4 ilustra a comparação entre o número de pedidos e o tempo de resposta associado. Mais uma vez, pode-se notar que o SOAP apresenta uma pequena desvantagem em relação ao desempenho do REST.

Figura 5.4 - COMPARAÇÃO ENTRE O TEMPO DE RESPOSTA E O NÚMERO DE REQUISIÇÕES



A utilização do processadorem todos os testes foi maior para o serviço *Web* SOAP. No caso do número de pedidos em relação ao tempo de resposta, SOAP também foi um pouco maior, e isso é considerado uma desvantagem na avaliação de desempenho.

Na Tabela 5.1, são apresentados, a média do tempo de resposta e o throughput de cada serviço *Web* em relação as quantidade de requisições. Pode-se observar que na Tabela 5.1, o throughtput do REST foi ligeiramente maior, constatando-se então que a vazão dos dados é maior no REST em relação ao SOAP, outro ponto positivo para o REST.

Tabela 5.1 - RELACÃO DO TEMPO DE RESPOSTA COM THROUGHPUT

Clientes	**Medição REST(ms)**	**Medição SOAP(ms)**	***Throughput* REST(Clientes/ms)**	***Throughput* SOAP(Clientes/ms)**
100	29	32	3,44	3,12
200	49	59	4,08	3,38
300	71	98	4,22	3,06
400	101	114	3,96	3,50
500	120	144	4,16	3,47

5.2 FASE DE SIMULAÇÃO

A fase de medição é realizada através da observação constante de um sistema a fim de medir, no mundo real (fase de medição), como as métricas do sistema são afetadas pelas variações de fatores específicos.Nesta seção, o foco será mostrar na prática como foi realizada a confecção do modelo de simulação. Antes da concepção do modelo, há vários passos a serem seguidos, e eles serão descritos nas próximas seções. Além disso, também se abordará questões relacionadas ao planejamento dos experimentos de simulação e tratamento dos dados. A fase de medição, que foi obtida anteriormente, é fundamental para esta etapa, pois vai servir de entrada para todos os passos que serão seguidos mais a frente.

5.2.1 OBTENÇÃO DO COMPORTAMENTO DOS DADOS

O objetivo desta etapa é determinar a distribuição de probabilidade teórica que melhor reproduz o comportamento dos dados obtidos na medição. Esta etapa é crucial para o modelo de simulação, visto que é com a função obtida nesta etapa, que os dados são representados que servirá de entrada para os objetos no modelo de simulação. O *software Input Analyzer* foi utilizado para a obtenção dos resultados desta etapa. As funções das distribuições de probabilidade estão descritas de acordo com as Tabelas 5.2 e 5.3. Na Tabela 5.2, pode-se observar que os tempos da rede (Tempo de Resposta menos o tempo de Latência) são representados através de uma distribuição triangular, já os tempos de processamento são representados por meio de uma distribuiçãofunção normal.

Tabela 5.2- FUNÇÃO DO COMPORTAMENTO DOS DADOS REST

REST	**DESCRIÇÃO**	**PARÂMETRO**
E1	Entidade 1: Requisição solicitada ao servidor *Web*	100,200,300,400,500
P1	Processo: Tempo da Rede	Função: TRIANGULAR (10,5, 13, 33,5)
P2	Processo: Tempo de Processamento	Função: NORMAL (10,9, 5, 72)
R1	Recurso: Rede	1 Recurso Computacional
R2	Recurso: Jax-RS	1 Recurso Computacional

Na Tabela 5.3, que é a do SOAP, os tempos da rede e os tempos de processamento foram representados pelas distribuições weibull e beta, respectivamente.

Tabela 5.3 - FUNÇÃO DO COMPORTAMENTO DOS DADOS SOAP

SOAP	**DESCRIÇÃO**	**PARÂMETRO**
E1	Entidade 1: Requisição solicitada ao servidor *Web*	100,200,300,400,500
P1	Processo: Tempo da Rede	Função: 12,5 + WEIBULL (11,8, 1,73)
P2	Processo: Tempo de Processamento	Função: 10,5 + 25 * BETA(1,11, 2,57)
R1	Recurso: Rede	1 Recurso Computacional
R2	Recurso: Jax-WS	1 Recurso Computacional

5.2.2 CRIAÇÃO DO MODELO DE SIMULAÇÃO

Para a criação do modelo de simulação proposto neste trabalho, foi utilizado o *software* Arena[41]. Pode-se inserir todas as informações obtidas (funções do tempo da rede e tempo de processamento) até agora no modelo criado. Nas Figuras 5.5 e

5.6 têm-se os modelos de simulações dos serviços *Web* REST e SOAP. No módulo cliente, foi inserido a quantidade de requisições. A cada utilização do modelo, variou-se a carga de trabalho entre 100 e 500 requisições. No módulo Requisição cliente é criada uma variável para o cálculo do tempo de resposta. Nos módulos Tempo da Rede e *WebService* REST/SOAP, foram inseridas as funções das distribuições de probabilidade obtidas anteriormente na etapa do comportamento dos dados.

Figura 5.5 - MODELO DE SIMULAÇÃO REST

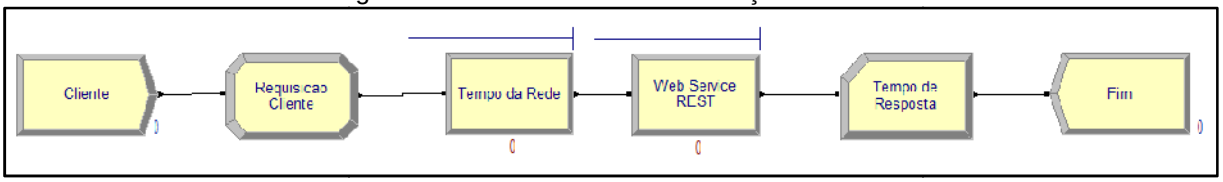
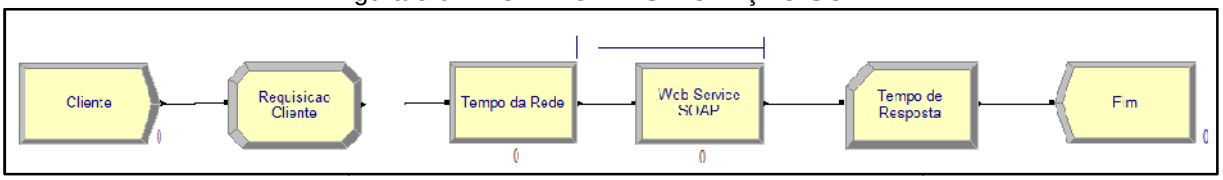


Figura 5.6 - MODELO DE SIMULAÇÃO SOAP



Foi utilizado Inicialmente o processo básico do Arena, o *Create*. Nele foi inserida a quantidade de clientes fazendo requisições. O processo inserido foi o *Assign* para representar a variável *Response Time*. A função da distribuição de probailidadeobtida a partir dos dados da medição foi inserida no processo “Tempo da Rede”. No módulo *Web* Services SOAP/REST, foi inserida a quantidade de servidores, neste caso 1, e a função que foi capturada na seção anterior. Inseriu-se também um processo tipo *Record* com o nome de Tempo de Resposta, onde foram gravados os resultados dos tempos (tempo da rede e o tempo do processamento) e por fim um processo Dispose, para retornar o resultado. Estes modelos de simulações, são baseados no modelo proposto na metodologia.

5.2.3 VALIDAÇÃO/AVALIAÇÃO DO MODELO

Para validar/avaliar o modelo, pode-se utilizar diversos tipos de critérios de decisões como o erro relativo, erro normatizado, teste t, entre outros. Neste estudo de caso foi utilizado o cálculo do percentual de erro relativo que é calculado através

da seguinte expressão: ((|Valor Medido - Valor Simulado|)x100)/Valor Medido. Nas Tabelas 5.4 e 5.5, são apresentadas as respostas obtidas tanto na medição quanto na simulação, e na última coluna o erro relativo. Outro fato para ser observado nas Figuras 5.4 e 5.5, é que neste estudo de caso foi considerado um erro relativo de 10% em todas as situações, o que ajuda a mostrar que o modelo pode ser usado, dentro de uma margem de erro, para se fazer inferências através do mesmo.

Tabela 5.4 - RELAÇÃO ENTRE TEMPO DE RESPOSTA, SIMULAÇÃO E O ERRO RELATIVO DO REST

REST	MEDIÇÃO	SIMULAÇÃO	ERRO RELATIVO
100	29	26,38	9,03
200	49	52,07	6,26
300	71	77,39	9
400	101	102,74	1,72
500	120	127,87	6,55

Tabela 5.5 - RELAÇÃO ENTRE TEMPO DE MEDIÇÃO, SIMULAÇÃO E O ERRO RELATIVO DO SOAP

SOAP	MEDIÇÃO	SIMULAÇÃO	ERRO RELATIVO
100	32	31,37	1,96
200	59	62,45	5,84
300	98	93,79	4,29
400	114	125,05	9,69
500	144	156,37	8,59

5.2.4 PLANEJAMENTO E EXECUÇÃO DA SIMULAÇÃO

O objetivo desta etapa é planejar os experimentos de simulação que deverão ser executados. Nesta etapa do estudo de caso, já se possui todos os resultados da medições e também já se tem o modelo validado através do erro relativo menor que 10% para a comparação entre os dados medidos e os dados obtidos através do modelo de simulação. No estudo de caso proposto neste trabalho os fatores escolhidos foram:

- Quantidade de Requisições;
- Número de Servidores;
- Velocidade da Rede;
- Protocolo de Comunicação Utilizado.

A variável que foi utilizada como resposta experimental para o modelo foi o tempo de resposta (ms). O software utilizado para este planejamento foi o *Minitab* v17, que é um software voltado para fins estatísticos e que oferece uma ferramenta específica para Planejamento de Experimentos (DOE). Assim que foram inseridos todos esses fatores(Requisições, número de servidores, velocidade da rede e o protocolo) no Minitab, foi gerado aleatoriamente todas as combinações de acordo com as mesmas, com todas as relações possíveis entre os fatores, conforme pode-se observar na Tabela 5.6. Os cenários obtidos pelo Minitab foram representados através do modelo de simulação e só então inseridos na coluna de tempo de resposta(ms). Na Tabela 5.6 pode-se observar os fatores que foram variados neste planejamento de experimentos. Por exemplo, na primeira linha tem-se 100 requisições, 5 servidores clusterisados, uma rede de 10Mb, o protocolo REST. Todas estas informações foram inseridas no modelo de simulação e a resposta foi que para atender as 100 requisições foram necessários 16,55 milissegundos.

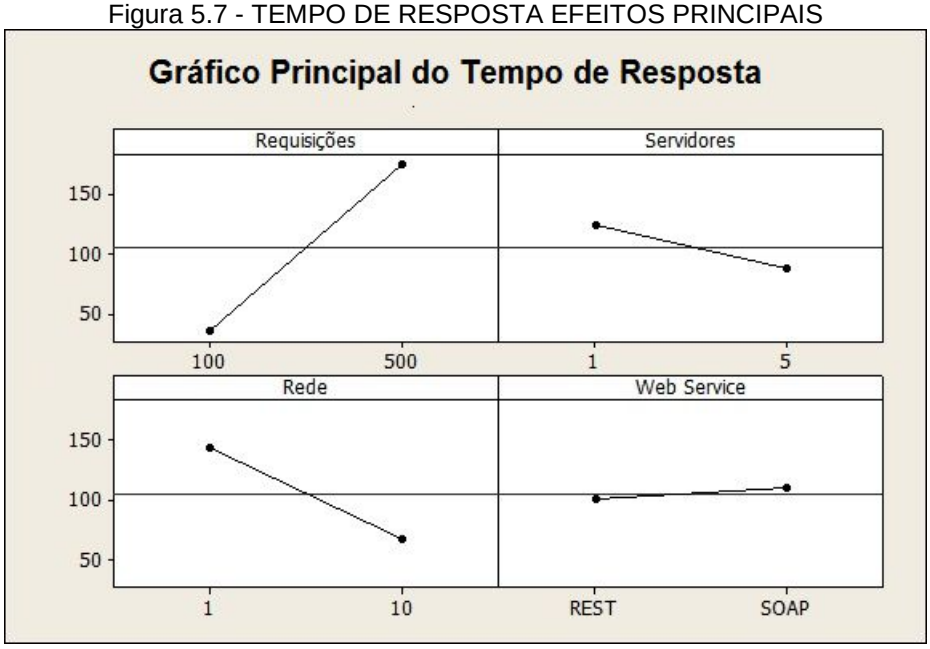
Tabela 5.6 - CENÁRIOS CRIADOS PELO PLANEJAMENTO DE EXPERIMENTOS

REQUISIÇÕES	SERVIDORES	REDE	PROTOCOLO	TEMPO DE RESPOSTA(MS)
100	5	10	REST	16,55
100	5	10	SOAP	16,63
100	1	10	REST	26,38
100	1	10	SOAP	31,37
100	5	1	REST	42,28
100	5	1	SOAP	43,17
100	1	1	REST	51,81
100	1	1	SOAP	57,60
500	1	10	REST	128,70
500	1	10	SOAP	156,37
500	5	1	REST	205,61
500	5	1	SOAP	209,58
500	1	1	REST	252,07
500	1	1	SOAP	284,43
500	5	10	SOAP	79,98
500	5	10	REST	81,40

Na Figura 5.7, pode-se observar o tempo de resposta em relação a todas as variáveis escolhidas inicialmente. Por exemplo, nota-se que o número de requisições é o que mais diferencia em relação ao tempo de reposta, pois o resultado com 100 e

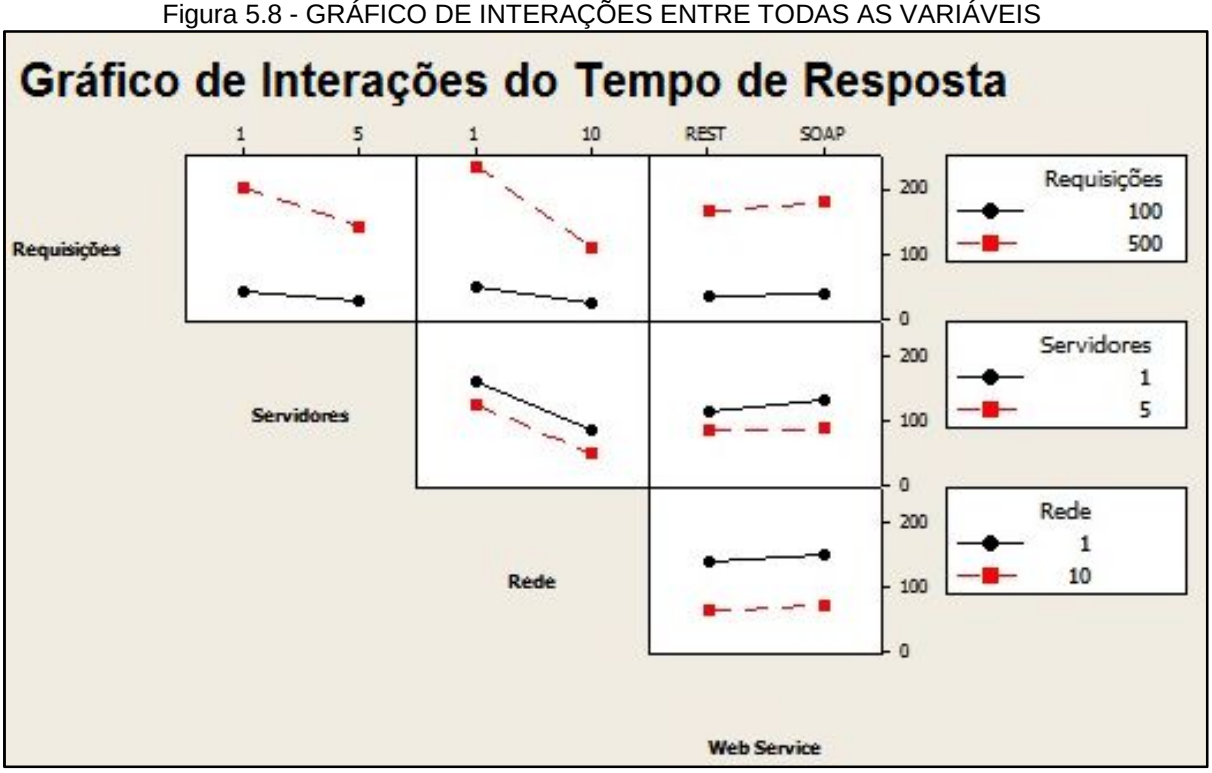
com 500 são bem diferentes. Já com o número de servidores não apresentou tanta diferença.

Figura 5.7 - TEMPO DE RESPOSTA EFEITOS PRINCIPAIS



Na Figura 5.8, o gráfico relaciona todas as interações entre os fatores e apresenta a relação que existe entre o tempo de resposta obtido. O que pode-se notar é que o gráfico que tem maior influencia é o de número de requisições com a rede.

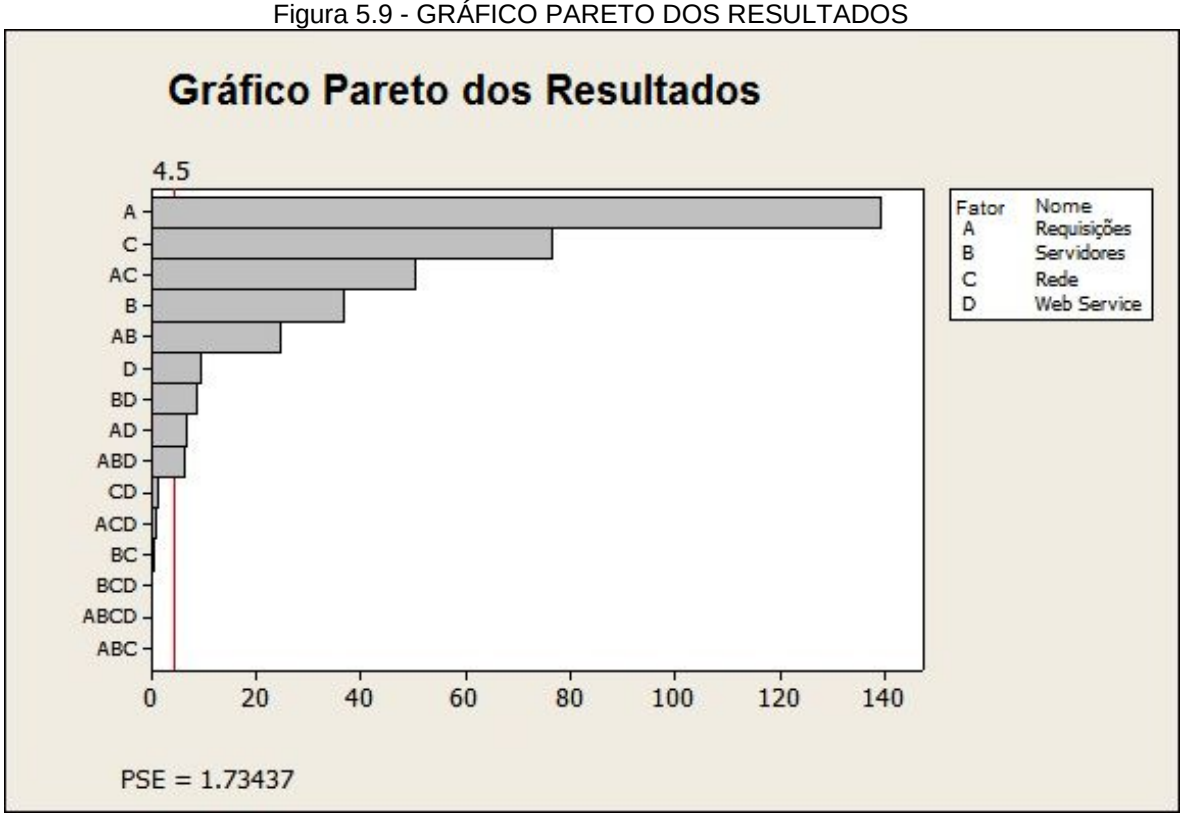
Figura 5.8 - GRÁFICO DE INTERAÇÕES ENTRE TODAS AS VARIÁVEIS



Fatores significativos como o número de requisições por exemplo, são aqueles que influenciam na resposta (tempo de resposta neste caso), conforme eles foram alterados de uma configuração para outra. O*Minitab* fornece uma tabela e um gráfico de Pareto[45] dos efeitos, tornando muito fácil a identificação dos fatores significativos.

O que pode-se notar no gráfico da Figura 5.9 é que o Fator A (número de requisições) é muito relevante, pois conta com quase 140% de significância. Já o Fator C (velocidade da rede) também apresentou grande importância com cerca de 80% de significância. A junção do Fator A, com o C, forneceu quase 50% de significância.

Figura 5.9 - GRÁFICO PARETO DOS RESULTADOS



5.2.5 AVALIAÇÃO

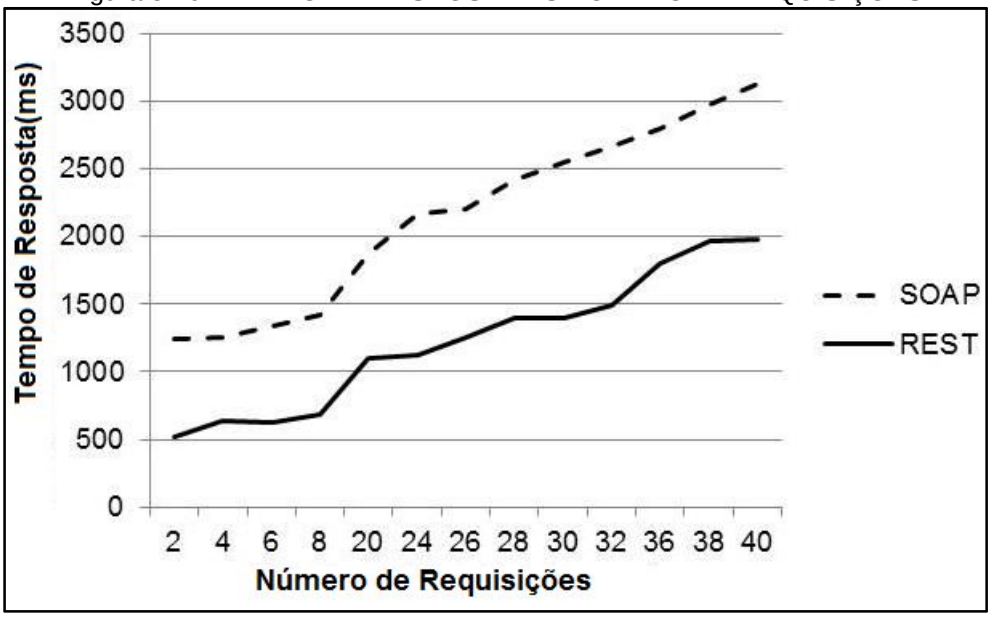
Com todos estes resultados obtidos, é relevante ressaltar que, existe sempre uma diferença entre os resultados de desempenho da CPU considerando os protocolos REST e SOAP, cujas diferenças podem serem vistas nas Figuras 5.3 e 5.4. Dentre várias razões que podem ser apontadas para este fato, uma delas reside

no fato que o cabeçalho da requisição SOAP é maior se comparado ao REST; consequentemente, ele envia mais dados para o servidor processar. Sendo assim, neste estudo de caso, tanto nos testes de avaliação do tempo de processamento quanto no tempo de resposta e também no *throughput*, o REST obteve resultados melhores em relação ao SOAP. Todas estas informações foram obtidas na fase de medição. Nesta mesma abordagem foi feito um modelo de simulação, e foram apresentados os detalhes importantes para representação do sistema a ser avaliado. Apesar do modelo ter sido representado na totalidade, houve um certo nível de abstração nesse modelo,a qualfoi cuidadosamente planejada. Os resultados da simulação ampliam os tipos de testes e a quantidade de cargas que pode-se obter do sistema.

Em relação a proposta da abordagem de avaliação de desempenho de serviços *Web*, ela propicia não apenas para estes dois serviços *Web* que foram mencionados no estudo de caso, mas para quaisquer outro serviço *Web* existente. Além de poder também comparar serviços *Web* e planejar como será o comportamento em diversas situações como por exemplo utilizando dispositivos móveis para testar os serviços *Web*. Uma observação a ser destacada, quando foram feitos os testes de medição, é que a quantidade máxima de requisições obtidas foi de 500 requisições, quando após este valor, a memória chegava ao seu limite, e não dava para prosseguir nos testes. Utilizando a abordagem proposta, pode-se simular um número bem maior de requisições.

Na mesma linha deste trabalho, o artigo de Kumar [9] mostra que os nossos resultados, em relação a medição do SOAP vs REST, foram compatíveis, mostrando que o SOAP levou mais tempo para atender as requisições. Pode-se observar na Figura 5.10, a comparação do tempo de resposta de acordo com o número de requisições HTTP. O resultado foi que o REST, em relação ao SOAP, obteve um tempo de resposta menor. Sendo assim, o resultado encontrado por Kumar, seguiu a mesma tendência dos resultados descritos no estudo de caso desta dissertação. A diferença foi que na metodologia proposta por Kumar, não é reaplicável a outros ambientes de medição.

Figura 5.10 - TEMPO DE RESPOSTA VS NÚMERO DE REQUISIÇÕES



5.2.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o estudo de caso que tanto ilustrou como ajudou na avaliação da proposta deste trabalho. O estudo de caso mostrou que a abordagem proposta permite que se possa fazer uma avaliação de desempenho em sistemas baseados em serviços *Web*.

6 CONCLUSÕES E TRABALHOS FUTUROS

6 CONCLUSÕES E TRABALHOS FUTUROS

6.1 CONCLUSÕES

O foco principal deste trabalho foi criar uma abordagem focada no desempenho de serviços *Web* onde duas técnicas de avaliação de desempenho bastante conhecidas, medição e simulação, atuam de forma integrada.A abordagem proposta neste trabalho é composta por uma metodologia e um modelo de simulação que podem ser usados, para diversos objetivos dentro de um estudo de desempenho de serviços *Web*.

A metodologia proposta neste trabalho é composta por um conjunto de atividades destinadas a avaliação de desempenho de serviços *Web*. Essas atividades compreendem desde o planejamento dos experimentos da medição até a simulação de casos mais robustos (ex.: com uma carga de trabalho maior) a análise de resultados.

Adicionalmente, este trabalho apresentou uma análise comparativa dos dois protocolos de comunicação mais utilizados em serviços *Web. No* estudo de caso proposto. Esse atualmente é um tema que vem sendo pesquisado dentro da comunidade, pois existe um interesse relacionado ao impacto no desempenho baseado na escolha do protocolo REST ou SOAP. Vários estudos relacionados foram apresentados no capítulo anterior e ilustram este interesse. O desempenho de serviços *Web* é geralmente avaliado com base em parâmetros como tempo de resposta do servidor ea taxa de throughput. O estudo desenvolvindo neste trabalho mostra que odesempenho do protocolo REST tende a ser melhor do que o desempenho do SOAPconsiderando as principais métricas estudadas(ex.: tempo de resposta e t*hroughput*). O protocolo REST obteve um desempenho melhor por ser um protocolo mais “leve”, envia menos dados. Já o SOAP tem uma tarefa a mais, pois tem que processar o enveloap SOAP que é em XML, trazendo um maior gasto de tempo neste processo.

É interessante observar que o RESTse mostrou mais rápido, em termos de tempo de execução, do que o SOAP, mesmo que a sua utilização ainda seja limitada pela IDE simples e pela falta de suporte e apoio ao desenvolvimento. Atualmente

existem muito mais materiais relacionados ao SOAP. O REST ainda não tem tantas informações específicas, e uma comunidade tão grande quanto a do SOAP.

6.2 CONTRIBUIÇÕES

A principal contribuição científica deste trabalho foi a proposta de uma abordagem integrada, composta por uma metodologia e um modelo de simulação, para a avaliação de desempenho de serviços *Web*. Esta abordagem, diferentemente de iniciativas existentes (ex.: [5], [9], [16]), integra as técnicas de medição e simulação, que são usadas de forma complementar. Essa integração resulta em diversas vantagens interessantes, pois cada uma dessas técnicas tem pontos fortes particulares. Por exemplo, a medição pode ser utilizada em um primeiro momento para um entendimento inicial do sistema e obtenção de medidas reais. Em um segundo momento, essas medidas reais são utilizadas para a construção de um modelo de simulação que permite a avaliação de outros casos que inclusive seriam mais complicados (ou até impossíveis) de serem trabalhados usando medição.
Outras contribuições deste trabalho são apresentadas a seguir.

- Metodologia de medição. Caso seja possível a análise do serviço *Web* apenas através da técnica de medição, a fase de medição da metodologia pode ser usada para guiar o estudo. Esta contribuição específica está inclusive publicada em um artigo fruto desta dissertação de mestrado [46], e pode servir como suporte para estudos de desempenho baseados em medição envolvendo serviços *Web*.
- Metodologia de simulação. Se houver a necessidade de aplicar a técnica de simulação para avaliação de cenários mais complexos, a fase de simulação da metodologia pode ser usada para guiar o estudo. Esta etapa é dependente da Fase de Medição, pois utiliza como dados de entrada os resultados da medição.
- Estudo de caso comparativo envolvendo os protocolos de comunicação SOAP e REST.

6.3 TRABALHOS FUTUROS

Diversos trabalhos futuros são vislumbrados no contexto deste trabalho. Uma possível melhoria deste trabalho é a inclusão de mais detalhes ao modelo de simulação proposto. Neste caso, teria como fazer todo o caminho da requisição, detalhando cada passo, podendo modifica-los a qualquer momento. Também pode-se desenvolver uma representação matemática para avaliação de serviço *Web*. Pode-se utilizar neste caso, formalismos como redes de Petri Estocásticas [26] ou Cadeias de Markov [24] para avaliar desempenho de serviços *Web*.

Uma outra possibilidade é a avaliação de outros cases de serviços *Web*, dentro desta metodologia proposta, como por exemplo, um case específico para ambiente mobile. Com um teste em ambiente mobile, pode-se avaliar por exemplo, qual o tempo de resposta dos dois tipos de serviços *Web*, SOAP e REST em um smartphone. Os dispositivos móveis apresentam uma grande necessidade de um bom desempenho, pois existe limitações de hardware e de rede (3G, 4G).

Outro caso importante seria a avaliação de requisitos não-funcionais como disponibilidade, segurança, entre vários outros. Com a disponibilidade, poderia calcular o tempo de indisponibilidade de um serviço. Já em segurança, pode-se testar entre os dois serviços *Web*, SOAP e o REST, qual apresentaria o melhor desempenho pois, no REST a segurança pode ser adicionada através do HTTPS, já no SOAP existe o *WS-SECURITY[47].*

REFERÊNCIAS

[1] Papazoglou, M. and Heuvel, W. “Service oriented architectures: approaches, technologies and research issues,” In VLDB Journal, v. 16, 389- 415. 2007

[2] W3C. “SOAP v. 1.2,” 2007, Available at http://www.w3.org/TR/2007/REC-soap12-part000704 27/ (last visit: 28th February 2015).

[3] Fielding, R.T. “Architectural styles and the design of networked-based software architectures. PhD thesis. Department of Information and Computer Science, University of California, Irvine.Cardoso, J. Business process control-flow complexity: metric, evaluation and validation,” 2000, International Journal of Serviços Web Research, v. 5, pp. 49-76. 2008

[4] Sérgio Nunes,Gabriel David, " Uma Arquitectura Web para Serviços Web", XATA 2005-XML:Aplicações e Tecnologias Associadas, 2005, último acesso em 05 de Maio de 2015: **http://hdl.handle.net/10216/281**

[5] Bertocci05] Vittorio Bertocci, “EndtoEnd Security,” http://blogs.msdn.com/b/vbertocci/archive/2005/04/25/end-to-end-security-or-why-youshouldn-t-drive-your-motorcycle-naked.aspx?CommentPOSTed=true#commentmessage,último acesso em 11 de Maio de 2015.

[6] P Markey, G Clynch, “A performance analysis of WS-*(SOAP) and RESTfulServiços Web for Implementing Service and Resource Orientated Architectures,” 2013, Available at http://arrow.dit.ie/ittscicon/1/ (last visit 31rd March 2015).

[7] PK Potti, S Ahuja, K Umapathy, “Comparing Performance of Serviços Web Interaction Styles: SOAP vs. REST,” 2012, Available at http://scholar.google.com/scholar?cites=14866246770289479857&as_sdt=2005&sciodt=0,5&hl=en&num=20 (last visit 31rd March 2015).

[8] Tiago Santos "URL: http://www.revistaproducaoengenharia.org/arearestrita/arquivos_internos/artigos/1-269%20-%20formatado%20em%206-8-13.pdf" Último acesso em 15/05/2015

[9] PK Potti, S Ahuja, K Umapathy, “Comparing Performance of Serviços Web Interaction Styles: SOAP vs. REST,” 2012, Available at http://scholar.google.com/scholar?cites=14866246770289479857&as_sdt=2005&sci

odt=0,5&hl=en&num=20 (last visit 31rd March 2015).

[10] Raj Jain, “The Art of Computer Systems Performance Analysis: techniques forexperimental design, measurement, simulation and modeling,” John Wiley,1991, ISBN: 0-471-50336-3

[11] FREITAS FILHO, P. J. de. Introdução à modelagem e simulação de sistemas:com aplicações em arena.2. ed. Florianópolis: Visual Books, 2008.

[12] Wambua "URL: http://www.researchgate.net/profile/Lawrence_Muchemi/publication/272067608_Comparative_Study_of_Rest_and_Soap_A_Case_Study_of_Registrar_of_Political_Parties_in_Kenya/links/54d9d9b80cf24647581f9a13.pdf" Último acesso em 15/05/2015

[13] VARNISH "URL: https://www.varnish-cache.org/" Último acesso em 19/05/2015

[14] POUND "URL: http://www.apsis.ch/pound/" Último acesso em 19/05/2015

[15] ZOPE "URL: http://www.zope.org/" Último acesso em 19/05/2015

[16] Ardagna, C.A.; Damiani, E.; Sagbo, K.A.R., "Early Assessment of Service Performance Based on Simulation," Services Computing (SCC), 2013 IEEE International Conference on, vol., no., pp.33, 40, June 28 2013-July. "URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6649675" Último acesso em 20/05/2015

[17] BENHARREF, A.; SERHANI, M. A.; BOUKTIF, “S. Online Monitoring for Sustainable Communities of Serviços Web,” Poster Session. 12th IFIP/IEEE, 2011

[18] Erl, T. “Service-oriented Architecture: Concepts, Technologies and Design,” Prentice Hall, 2005

[19] Erik Christensen, Francisco Curbera, Greg Meredith, and SanjivaWeerawarana, “WSDL 1.1 Document,” www.w3.org/TR/wsdl, last accessed March 7, 2011.

[20] OMG. "Common Object Request Broker Architecture: Core Specification (CORBA 3.0)". 2002

[21] SANTANA, R.H.C.; SANTANA M.J.; BARBATO A.K.; TRALDI O.A. "Avaliação de Desempenho: Modelagem e Simulação. Relatório Técnico, São Carlos: Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2004.

[22] FRANCÊS, C, R, L; SANTANA M. J; SANTANA R. H. C.;ORLANDI R. C.G.S."Tolls and Methodologies For Performance Evaluation of Distributed Computing Systems - A Comparsion Study" Proceedings of the 1997 Summer Computer Simulation Conference(SCSC'97), 1997.

[23] MENASCÉ, D. A.: ALMEIDA, V.A. F. Performance by Design: computer capacity planning by example. [S.1.]: Prentice Hall PTR, 2005.

[24] BOLCH, G. et al. Queueing Networks and Markov Chains: modeling and performance evaluation with computer science applications. [S.1.]: Wiley-Interscience, 2006.

[25] XIONG, K.; PERROS, H. Service Performance and Analysis in Cloud Computing. Services-I, 2009 World Conferenceon, [S.1.], 2009

[26] GERMAN, R. Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets. [S.1]: John Wiley& Sons, Inc. New York, NY, USA, 2000.

[27] BRUSCHI, S. M. "ASDA - Um Ambiente para Simulação Distribuída Automático." Tese(Doutorado). São Carlos: ICMC - USP, Novembro de 2002.

[28] CHANG, X. "Network Simulations with OPNET." Proceedings of the 1990 Winter Simulation Conference, 1999

[29] Apache "URL: http://www.apache.org/" Último acesso em 15/05/2015

[30] Internet Information Services "URL: http://www.microsoft.com/web/platform/server.aspx" Último acesso em 15/05/2015

[31] Lighttpd "URL: http://www.lighttpd.net/" Último acesso em 15/05/2015

[32] Nginx "URL: http://nginx.org/" Último acesso em 15/05/2015

[33] G-WAN "URL: http://gwan.com/" Último acesso em 15/05/2015

[34] Halili, Emily H, "Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites,", 2008

[35] MOODLE Moodle "URL: http://moodle.com/" Último acesso em 19/05/2015

[36] Fiddler "URL: http://www.telerik.com/fiddler" Último acesso em 15/05/2015

[37] Enbody, R, “Perfmon: Performance Monitoring Tool,” Michigan State University, 1999, http://www. cps. msu. edu/enbody/perfmon. html (last visit: 28th February 2015)

[38] Java MissionControl "URL: http://www.oracle.com/technetwork/java/javaseproducts/mission-control/java-mission-control-1998576.html" Último acesso em 15/05/2015

[39] VEVERKA, V. V.; MADRON, F. Material and Energy Balancing in the Process Industries: From Microscopic Balances to Large Plants. [S.1.]: Elsevier, 1997.

[40] Outlier "URL: http://www.estgv.ipv.pt/PaginasPessoais/psarabando/CET%20%20Ambiente%202008-2009/Slides/8.%20Outliers.pdf" Último acesso em 15/05/2015.

[41] SWET, R. J.; DRAKE G. R. "The Arena Product Family: Enterprise Modeling Solutions. "ProceedingofWinterSimulationConference (WSC 2001), 2001.

[42] OMNeT++"URL: https://omnetpp.org/" Último acesso em 19/05/2015

[43] Netbeans "URL: https://netbeans.org/" Último acesso em 15/05/2015

[44] Statdisk "URL: http://www.statdisk.org/" Último acesso em 15/05/2015

[45] Pareto "URL:http://www.portal-administracao.com/2014/04/diagrama-de-pareto-passo-a-passo.html" Último acesso em 15/05/2015

[46] Marcelo"URL:http://www.ijser.org/onlineResearchPaperViewer.aspx?Towards-a-Methodology-for-Performance-Measurement-of-Service-Based-Systems.pdf" Último acesso em 19/05/2015

[47] Web2.0security10] “Security for REST and Web2.0,”www.matsays.com/downloads/inf400/inf400-sp10-wk14b2.pdf, último acesso em 11 de Maio de 2015.