COMPUTAÇÃO DISTRIBUÍDA – COMPARATIVO DE DESEMPENHO ENTRE WEB SERVICE E RMI

Rogério Francia Farias [1]
Evandro César Freiberger[2]

RESUMO

Serviços de Acesso remoto e Arquitetura Orientada a Serviço têm sido muito discutidos nos últimos anos. Para tanto se torna imprescindível a avaliação de desempenho entre as tecnologias desenvolvidas e utilizadas nos dias de hoje. Apresenta-se neste artigo o desenvolvimento de uma aplicação que faz consulta a um banco de dados corporativo utilizando duas tecnologias de computação distribuída: *Web Services* e RMI. O objetivo é produzir, sob as mesmas condições, comparações de performance entre as duas tecnologias.

Palavras-chave: *Arquitetura Orientada a Serviço (SOA) - Computação Distribuída - Web Service - Invocação de Métodos Remotos – RMI.*

ABSTRACT

Remote Access Services and Architecture based on Service have been largely discussed a long time. So becomes indispensable the performance and security valuation between the technologies developed and used actually. This article presents the development of the application that consult the corporate database using two technologies of distributed computing: *Web Services* and RMI. The target is to produce, under the same conditions, comparison of performance between these two technologies.

Keywords: *Service Oriented Architecture (SOA) - Distributed Computing - Web Service - Remote Method Invocation (RMI).*

---

[1] Graduado em Processamento de Dados, pela UNIVAG – Várzea Grande/MT, Pós-Graduando pelo Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso, campus Cuiabá/MT.

[2] Msc. pela Universidade Federal Santa Catarina – UFSC.

INTRODUÇÃO

Hoje o mundo observa atentamente a evolução extraordinária da informática, desde a primeira geração dos computadores descrita como modelo de "*Von Neumann*" até nos dias atuais do mundo globalizado e a explosão meteórica da internet, muita coisa mudou. Vários modelos, arquiteturas, redes e linguagens surgiram, havendo assim, a popularização dos recursos tecnológicos e o aumento considerável da sua utilização por todas as partes da sociedade.

Devido ao avanço tecnológico das ultimas décadas e o surgimento da rede mundial de computadores, Brax & Leal (2001), afirmavam que os equipamentos tecnológicos foram empregados amplamente no mercado, surgindo assim, a necessidade de desenvolver equipamentos com maior capacidade de processamento, armazenamento e segurança, para os Softwares mais complexos, obtendo melhor compartilhamento de recursos e comunicação para dar suporte ao chamado mundo globalizado.

Porém, a demanda da capacidade de processamento é maior que as tecnologias desenvolvidas, como acontece com as diversas prestações de serviços à sociedade: serviço de meteorologia, entidades governamentais, sistemas financeiros, *e-Business, e-commerce* que transformam o mundo atual em um grande meio corporativo entre as áreas dos Governos, Indústria, Comércio e Serviços, impulsionando o surgimento de soluções como forma de melhorar a integração dos vários sistemas corporativos existentes, buscando a Interoperabilidade e homogeneidades das informações trocadas neste contexto.

Neste cenário globalizado e corporativo surge o principio de computação distribuída, que possibilita muitos sistemas locais de diferentes plataformas, trocarem informações em um mesmo banco de dados corporativo dando grande destaque a SOA - Arquitetura Orientada a Serviços. Esta Arquitetura, para Thomas (2009), tem o intuito de prover a interoperabilidade entre os sistemas já existentes e de diferentes tipos de linguagem que provém informações e que precisam de um padrão para ser manipulados pelas diversas instituições, proporcionando assim, vários benefícios.

Padronização essa, que segundo Brax & Leal (2001), é feita pela W3C e conta com cooperação de grandes empresas como *IBM*, *Microsoft*, *Oracle*, dentre outras, o que se faz necessária, uma vez que exista grande dificuldade desses sistemas se comunicarem.

O método de desenvolvimento deste artigo será de uma pesquisa de laboratório, que investigará o desempenho de duas tecnologias de acesso remoto, *Web Service* e RMI. Serão desenvolvidas essas duas aplicações utilizando os conceitos das tecnologias de acesso remoto que utilizam o princípio da computação distribuída e o conceito de SOA, em seguida será gerada uma busca a uma certa quantidade de registro e posteriormente será apresentado seus desempenhos, quanto a performance.

A relevância deste trabalho se dá ao fato de que, ao final poderão ser observados os desempenhos das duas tecnologias desenvolvidas, onde os números apresentados poderão servir de subsidio em trabalhos futuros, ou até mesmo em uma posterior escolha para implementação futura.

1. TECNOLOGIAS DE ACESSO REMOTO

Para Deitel (2005), várias tecnologias denominadas de *Middlewares* foram implementadas até o atual momento, porém uns foram mais sucedidos que outros em oferecer portabilidade e interoperabilidade. Utilizando do conceito de chamada de procedimento remoto – RPC (desenvolvido nos anos 80), estas ferramentas buscam e/ou fornecem informações por uma rede (internet, por exemplo) utilizando de uma interface padrão entendida por ambos os lados. Dentre os principais *Middlewares*, destacam-se: *Web Service* e o RMI.

2. 1 COMPUTAÇÃO DISTRIBUÍDA E SOA

Para melhor entendimento das tecnologias de *Web Servives* e RMI, necessita-se entender um pouco sobre computação distribuída ou Sistemas distribuídos e SOA. Segue um breve conceito sobre sistemas distribuídos que ontem eram uma forte

tendência e hoje já é realidade, na medida em que as redes ficam mais velozes e confiáveis.

2.1.1 Computação distribuída ou sistemas distribuídos

Segundo Coulouris (2007), os sistemas distribuídos são a interatividade de vários equipamentos e softwares de diferentes fabricantes, arquiteturas e linguagens interligadas através de uma rede. Pode-se dizer que a internet é um modelo de sistemas distribuídos, onde vários equipamentos e softwares heterogêneos se interagem remotamente através da grande rede, sendo oferecidos por varias entidades, inúmeros serviços como: *e-mail*, transferência de arquivos *(FTP)*, WWW, Voip, mensagens instantâneas e etc.

Já Tanenbaum (1994), afirma que nem todos os sistemas que se dizem ser distribuídos realmente são, principalmente quando estes tornam possível dizer quem é responsável por alguma tarefa. O modelo cliente/servidor pode-se afirmar que é um modelo para sistemas distribuídos, uma vez que, o acesso é feito através de requisições e todas as entidades envolvidas podem ser cliente e servidor: cliente quando solicita algo pela rede e servidor quando compartilha algo que lhe é solicitado.

Com o grande crescimento da internet os sistemas distribuídos foram se tornando mais comuns, uma vez que eles buscam desempenho, portabilidade, escalabilidade, conectividade, segurança, confiabilidade e tolerância a falhas de uma maneira conjunta, de modo a se tornarem ferramentas mais completas e que atendem as necessidades em comum. Deitel (2005), diz que o desafio principal em um projeto de sistemas distribuídos, é gerenciar a comunicação entre vários componentes envolvidos tendo como objetivo principal prover a interoperabilidade entre computadores e aplicações heterogêneas.

Defini-se hoje, que computação distribuída é a divisão de tarefas de um sistema grande em vários processadores independentes trabalhando como se fossem um só. Assim, os sistemas distribuídos utilizando de softwares que podem ser fortemente ou fracamente acoplados são os exemplos mais modernos de prestação de serviços.

2.1.1.1 Softwares Fortemente e Fracamente Acoplados

Todo o conceito de computação distribuída é fundamentado no acoplamento entre os sistemas, quanto menos acoplados, melhor pode ser integrados os sistemas, por causa da independência que existe entre eles. Santos (2007) define que os Softwares fortemente acoplados provêm um nível de integração e compartilhamento de recursos mais intenso e transparente ao usuário caracterizando sistemas operacionais distribuídos e conceitua ainda que, por outro lado, os Softwares fracamente acoplados permitem que máquinas e usuários de um sistema distribuído sejam fundamentalmente independentes e interagem de forma limitada quando isto for necessário. Porém, os softwares com menos acoplamento são softwares mais complexos e de difícil manutenção, por isso são menos utilizados.Todavia os softwares com maior acoplamento são menos complexos e de mais fácil manutenção. Por isso, são mais utilizados.

2.1.2 Arquitetura Orientada a Serviço - SOA

Para complementar o conceito de Computação Distribuída, a Arquitetura Orientada a Serviço (SOA) é um modelo de arquitetura que contempla as tecnologias *Web Servive* e RMI além de outras tecnologias de acesso remoto. Esta arquitetura de software provê comunicação de serviços disponibilizados através de interfaces de sistemas clientes com interfaces que disponibilizam serviços e são organizados por barramentos de serviços (*Enterprise Service Bus*). Esta arquitetura permite maior suporte tecnológico às empresas e permite o crescimento sistêmico e dinâmico da parte de negocio do sistema corporativo, sem trazer prejuízo ao desempenho da comunicação com o mundo exterior.

Thomas (2009, p.24) entende que:

> A SOA estabelece um modelo arquitetônico que visa a aprimorar a eficiência, a agilidade e a produtividade de uma empresa, posicionando os serviços como os principais meios para que a solução lógica seja representada no suporte à realização dos objetivos estratégicos associados à **computação orientada a serviços** (grifo nosso).

Portanto, esta arquitetura somente estabelece fundamentos baseados em características, princípios, paradigmas, modelos, linguagem de modelos, padrão e boas praticas, formando assim, uma ótima pilha de procedimentos que dão suporte a computação orientada a serviços.

A plataforma da computação orientada a serviço é mais complexa que as típicas, uma vez que possuem elementos que se relacionam de forma a proporcionar o melhor fluxo possível dos processos dos sistemas de informação.

Thomas (2009, p.24) diz ainda que:

> A computação orientada a serviços representa uma nova geração da plataforma da computação distribuída. Como tal, ela abrange muitas coisas, incluindo seu próprio paradigma de *design* e princípios de *design*, catálogos de modelo de *design*, linguagens-padrão, um modelo arquitetônico distinto, tecnologias e *frameworks* relacionados.

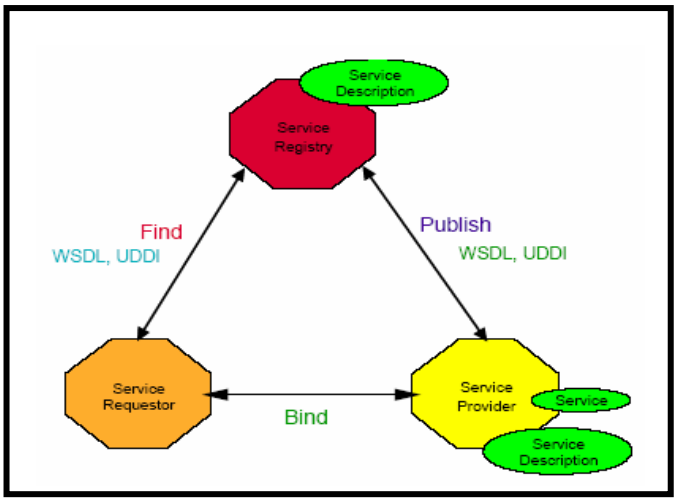
Pode-se dizer então que esta plataforma vem complementar as atuais da computação distribuída, adicionando novas camadas de design e outros atributos importantes.

2. 2 WEB SERVICES

Esta tecnologia também pode ser desenvolvida em linguagem Java e roda em qualquer sistema operacional, uma vez que, este utiliza a JVM como interpretador neste caso. O *Web Service* pode ser compreendido como uma tecnologia de acesso a serviços remotos através da troca de mensagens utilizando o padrão XML.

Ela traz como maior beneficio a independência de plataformas de *Hardware* e *Software* na qual a comunicação estabelecida entre dois dispositivos remotos: *Interface* Servidora e a *Interface* Cliente geram maior flexibilidade na relação entre linguagens de programação, Sistemas Operacionais, disponibilização dos serviços e o dispositivo que acessa o *Web Service.* Tudo isso é transparente aos usuários, uma vez que, utiliza-se as descrição de WSDL, que assim executa a implementação via *browser.*

Conforme Girard (2004), a tecnologia *Web Service* apresenta uma estrutura de arquitetura de um provedor do serviço: esta arquitetura é composta por um servidor e um cliente do serviço como descreve a figura do autor:



Fonte: *Extraído da Dissertação de Mestrado de Girard (2004)*
Figura 01 – Estrutura de comunicação *Web Service.*

Web Service utiliza a porta 80 (padrão), portanto, torna-se mais fácil sua implantação e configuração de *Firewalls*, *Proxis* e outros em redes corporativas e Internet.

2. 3 REMOTE METHOD INVOCATION - RMI

RMI, (*Remote Method Invocation*) ou em Português Invocação de Métodos Remotos. Esta tecnologia foi uma evolução do RCP (*Remote Procedure Call*) ou em Português: Chamada de Procedimentos Remotos, onde foi desenvolvido na década de 80 e que tem por finalidade executar procedimentos e funções de sistemas procedurais localizados em outros locais, porém, com a migração do modelo de desenvolvimento procedural para o modelo orientado a objetos, gerou alguns conflitos acarretando a necessidade de adequações ao novo modelo.

A tecnologia de Invocação de Métodos Remotos – RMI é uma solução que também utiliza a linguagem Java como *Web Sevice* e também provê comunicação entre sistemas remotos a partir da execução dos métodos de objetos localizados em diversos locais através da computação distribuídas orientadas a objeto.

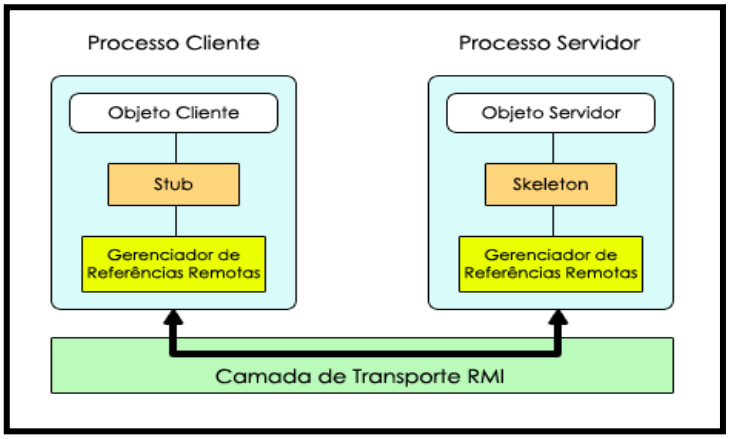
Ela tem como característica um Servidor que disponibiliza os serviços de acesso remoto aos métodos, através de classes denominadas: *Skeletons* e um dispositivo cliente que requisita estes serviços providos pelo servidor através das

classes chamadas *Stub*, tudo isso interligado por uma interface Java abstrata, juntamente com a camada de transporte RMI.

O mais interessante de tudo isso, é que essa estrutura de acesso remoto também é transparente aos usuários, como os *Web Services*, assim, toda essa execução remota é realizada como se fosse local.

O RMI utiliza a porta *default* 1099, ou o desenvolvedor pode configurar uma outra porta na implementação da aplicação, isso requer uma atenção especial por parte dos desenvolvedores e dos administradores de redes na configuração de ferramentas de proteção.

A figura a seguir, de Castro, Raeder e Nunes (2007), demonstra uma estrutura básica da tecnologia:



Fonte: Extraído do Artigo de Castro; Raeder e Nunes (2007)
Figura 02 – Estrutura de comunicação RMI

Observa-se que a estrutura do RMI é relativamente simples, onde tanto os processos cliente, quanto o processo servidor se comunicam facilmente. Sendo os responsáveis por essa comunicação é basicamente as classes *Stub* e *Skeleton*.

3. PESQUISA DE DESEMPENHO ENTRE WEB SERVICE E RMI – ESTUDO DE CASO

Foram desenvolvidas duas aplicações Java, a primeira foi um Web Service, que disponibiliza os serviços através da comunicação com trocas de mensagens

utilizando o padrão XML. Na segunda aplicação, foi desenvolvido um RMI, que por sua vez, disponibiliza um servidor de registros que pode registrar vários serviços.

Ambas compartilham as camadas de negócio e persistência, com o objetivo de propor o menor acoplamento possível entre as aplicações, assim, não haverá dependência entre as tecnologias utilizadas, possibilitando ainda qualquer outra tecnologia de acesso remoto utilizar dessas duas camadas para realizar o acesso a um Banco de Dados.

Neste caso, fora utilizado um Banco de dados de ocorrências da Polícia Militar do Estado de Mato Grosso com aproximadamente quatrocentos mil registros. A consulta realizada neste BD e feito em iguais condições de equipamentos e rede, buscando observar o desempenho em uma consulta em exatamente 150.000 (cento e cinquenta mil) registros.

Foi elaborado neste trabalho o diagrama de processo a seguir para demonstrar de forma simples a estrutura entre as camadas geradas pelas aplicações e seus pacotes, bem como a comunicação entre os componentes que o modelo utiliza:

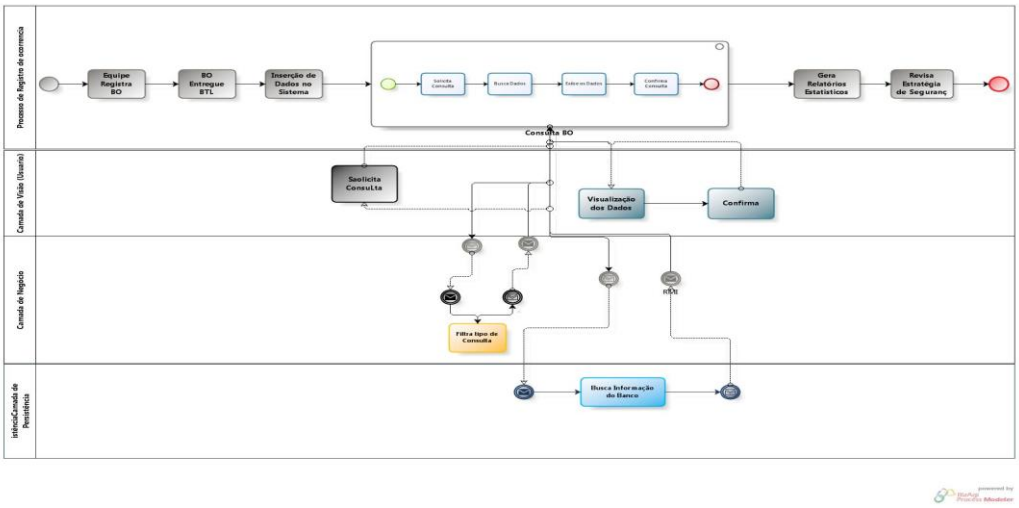


Figura 03 – Diagrama de Processo das aplicações

O diagrama de processo simula um fluxo dos processos de registro de ocorrências da PMMT, e o processo *Consulta BO* na camada de visão, demonstra que as interfaces clientes solicitam o serviço de conexão ao BD, disponibilizados pelas interfaces servidoras das tecnologias apresentadas neste trabalho. Este serviço

requerido é transferido às camadas de negócio que realiza a conexão com o BD na camada de persistência, onde esta realiza a consulta e aponta o resultado.

A utilização desta camada de negócio possibilita utilizar diversas tecnologias de acesso remoto, fazendo com que a conexão com o Banco de Dados seja sempre intermediada por essa camada, que proporciona a interdependência de tecnologias.

Também foi elaborado neste trabalho o diagrama de sequência para demonstrar melhor a comunicação entre essas camadas:

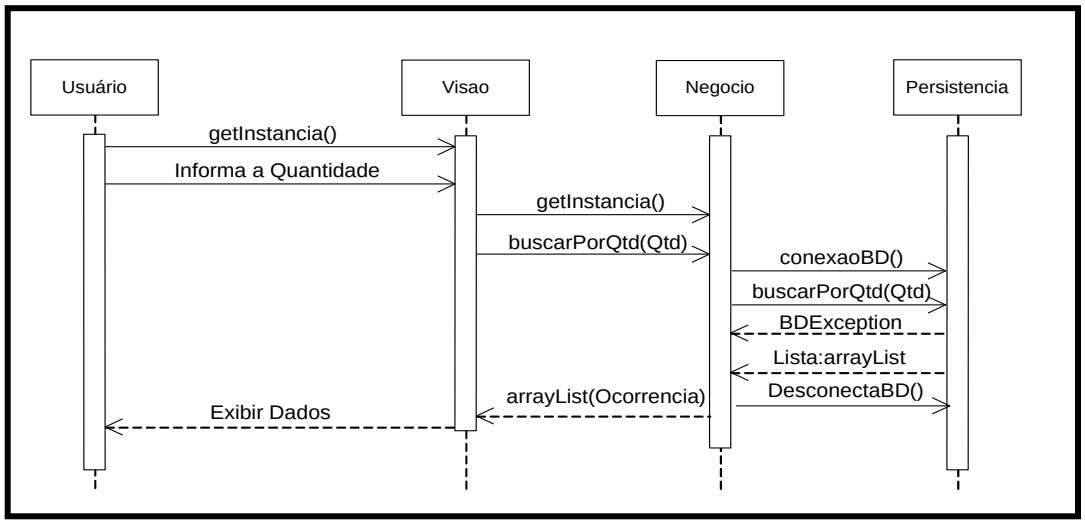


Figura 04 – Diagrama de Sequência das camadas

3. 1 ETAPAS DE DESENVOLVIMENTO

As aplicações apresentadas neste artigo foram desenvolvidas utilizando-se a IDE *JDevelop 11.1.1.2.0* da *Oracle* para o desenvolvimento, sendo que o *Web Service* utiliza também o *GlassFish 3.0* como servidor de aplicações, onde foram divididas em três camadas: *Visão*, Negócio e Persistência.

Na camada de visão, existem as interfaces clientes e servidoras das tecnologias de acesso remoto. Na camada de Negócio contém os pacotes de classes que chama o método de conexão com o BD da camada de persistência e os disponibiliza para registro das interfaces servidoras para acesso remoto das interfaces clientes e na camada de persistência, contem a classe do método que faz a conexão com BD.

3.1.1 Camada de visão – tecnologia web service

No desenvolvimento desta aplicação, observa-se que é simples a construção das *interfaces* cliente e servidor, uma vez que, a própria IDE utilizada, neste caso o *JDeveloper*, constrói a versão *default* e o desenvolvedor apenas configura os parâmetros necessários, como a descrição do endereço do servidor Web Service na descrição WSDL da interface cliente.

Foram criados dois projetos sendo do tipo *Web Service*; um para a interface servidora que importa os pacotes dos métodos das camadas de negócio e persistência, dando assim, o comportamento da aplicação, que no caso é a disponibilização do método de conexão com o banco de dados.

Posteriormente foi criado outro projeto, o da *interface* cliente, onde este é referenciado o endereço da *interface* servidor, que quando executado, importa a classe do painel para interatividade com o usuário.

3.1.2 Camada de visão – tecnologia RMI

Na aplicação do RMI, seu desenvolvimento é semelhante ao do *Web Service*, porém com algumas ressalvas. Observou-se neste projeto, que ele é mais complexo para se desenvolver, exigindo mais trabalho, atenção e conhecimento de acesso remoto e serviços. A diferença primordial para o *Web Service*, é que além do projeto conter os pacotes da *interface* cliente e a *interface* servidora, este tem um pacote contendo uma *interface* abstrata que prove a comunicação entre as *interfaces* cliente e Servidora RMI e uma classe que implementa desta *interface.*

Quando o projeto RMI é executado ele também importa os pacotes das camadas de negócio e persistência igualmente ao *Web Service*. A interface de comunicação entre as interfaces obtém os métodos que a interface servidora chamada Ca camada de negócio.

A maior dificuldade está para executar os *rmic* e *rmiregistry*, uma vez que, não se tomou muito cuidado com a configuração do ambiente de desenvolvimento. No caso deste trabalho, foi criado até um arquivo *.bat* para automatizar o registro do serviço ao servidor de registro RMI.

3.1.3 Camadas de negócio e persistência

As camadas de negócio e persistência estão inseridas em um mesmo projeto, e contém os pacotes *Negócio, Persistência* e *Vo.* Estes pacotes contêm os métodos de conexão com o BD, métodos que chamam a referida conexão, disponibilização para o registro das interfaces servidoras, na camada de visão. Estas camadas foram o grande diferencial entre outros projetos de tecnologias de acesso remoto observados.

4. APRESENTAÇÃO DOS RESULTADOS

No trabalho de desenvolvimento das aplicações e realização dos testes, foi observado que o tempo de resposta à consulta varia de acordo com a utilização do *cache* de memória e a utilização do processador, mesmo as aplicações sendo submetidas aos testes com mesmas condições de equipamentos e rede. Por isso, foram realizados varias consultas intercalando as tecnologias e finalizado com a média de vinte consultas para cada aplicação.

Poderá até ser feito uma análise dos resultados e comparar os desempenhos, porém em trabalhos futuros. Somente serão apresentados neste artigo os resultados dos desempenhos das tecnologias, que é o escopo do trabalho.

4.1 APRESENTAÇÃO DE DESEMPENHO DO WEB SERVICE

Nos testes realizados com o *Web Service*, pode ser observado a inconsistência dos tempos das consultas, variando entre o maior tempo: 1 mim, 22 s, 625 ms e o menor tempo: 21 s, 141 ms, obtendo a média de 26 s, 2559 ms.

Foi observado ainda que esta tecnologia obteve o maior entre os maiores tempos e o maior entre os menores tempos, onde estes não foram em ordem sequencial, havendo alternância entre os tempos.

4.2 APRESENTAÇÃO DE DESEMPENHO DO RMI

Nos testes realizados com o RMI, observou-se também a inconsistência dos tempos tomados nas consultas realizadas ao BD. Estes tempos variou-se entre o

maior tempo: 1 mim, 14 s, 078 ms e o menor tempo: 17 s, 828 ms, obtendo a média de 24 s, 5965 ms.

Foi observado ainda que esta tecnologia é mais robusta, tendo uma aproximação maior entre os tempos e obteve o menor entre os maiores tempos e também o menor entre os menores tempos, onde estes também não foram na ordem sequencial, havendo também um alternância entre os tempos.

CONSIDERAÇÕES

A partir do desenvolvimento do trabalho podemos considerar que este proporcionou uma experiência impar, no estudo de Redes de computadores e Computação Distribuída uma vez que, pôde-se estudar mais profundamente os conceitos e adquirir uma maior bagagem no desenvolvimento prático, podendo observar os pontos fortes e fracos de cada ferramenta e aprendendo com as dificuldades e particularidades de cada uma.

Conclui-se ainda que o desenvolvimento das camadas de negócio e persistência foram primordiais para garantir o máximo de reuso entre os sistemas e pode se tornar no futuro uma prática comum no desenvolvimento de sistemas distribuídos.

Após a realização dos testes de desempenho, observou-se que estas tecnologias vieram para ficar, tendo em vista, que o tempo gasto para a realização das consultas foram relativamente baixo perante uma consulta com tantos registros, podendo trazer com isso uma série de benefícios que poderão ser visualizado com uma analise mais detalhada das tecnologias.

REFERÊNCIAS BIBLIOGRÁFICAS

BRAX, P.M.; LEAL, G. J.; **Serviços Web e a Evolução dos Serviços em TI**; http://www.paradigma.com.br/biblioteca/servicos-web/view; acessado em 11/01/02010.

CASTRO, R.; RAEDER, M.; NUNES, T.; **RMI: Uma Visão Geral**; http://www.inf.pucrs.br/~gustavo/disciplinas/sd/material/Artigo_RMI_Conceitual.pdf; acessado em 26/07/2010.

COULOURIS, G. F.; DOLLIMORE, J.; KINDBERG, T. **Distributed Systems: Concepts and Design**. 3rd ed. Addison-Wesley, 2002.

COULOURIS, G. F.; DOLLIMORE, J.; KINDBERG, T.; **TITULO**; 1ª Edição; São Paulo; Bookman; 2007

DEITEL H. M.; DEITEL P. J.; CHOFFNES D. R.; **Sistemas Operacionais**; 3ª Edição; São Paulo; Pearson Prentice Hall; 2005

DUARTE DOS SANTOS, Bruno Martins; **Computação Distribuída**; http://www.webartigos.com/articles/2903/1/computacao-distribuida/pagina1.html, publicado em 2/12/2007, ultimo acesso 29/09/2009.

GIRARD, R. A. D'A.; **Framework para coordenação e mediação de Web Services modelados como Learning Objects para ambientes de aprendizado na Web**; http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0220942_04_cap_03.pdf; acessado em 22/07/2010.

TANENBAUM, A. S.; **Distributed Operating Systems**. 1st ed. New Jersey: Prentice Hall, 1994.

TANENBAUM, A. S.; VAN Maarten, **Sistemas Distribuídos: Princípios e Paradigmas**; 2ª Edição; São Paulo; Pearson Prentice Hall; 2007

THOMAS Erl; **SOA Principios de design de serviços**; São Paulo; Paerson Prentice Hall; 2009.