UNIVERSIDADE FEDERAL FLUMINENSE
CENTRO TECNOLÓGICO
CIÊNCIA DA COMPUTAÇÃO

JAIRO LINO DUARTE

PORTAL EASYGRID,
UMA EXPERIÊNCIA USANDO WEB SERVICES EM GRID

NITERÓI, RJ
2004

JAIRO LINO DUARTE
matrícula nº 299.31.066-7

PORTAL EASYGRID,
UMA EXPERIÊNCIA USANDO WEB SERVICES EM GRID

"Construção de um portal, utilizando Web Services para facilitar o uso de um Grid"

Dissertação apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

ORIENTADOR: PROF. VINOD REBELLO
IC / CTC / UFF

Niterói
2004

JAIRO LINO DUARTE
matrícula nº 299.31.066-7

PORTAL EASYGRID,
UMA EXPERIÊNCIA USANDO WEB SERVICES EM GRID

"Construção de um portal, utilizando Web Services para facilitar o uso de um Grid"

Dissertação apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Aprovada em Março de 2004

BANCA EXAMINADORA

Prof. Dr. Vinod Rebello
Universidade Federal Fluminense

Prof. Dr. Cristina Boeres
Universidade Federal Fluminense

Prof. Dr. Orlando Loques
Universidade Federal Fluminense

Niterói
2004

Sumário

Lista de Abreviaturas, Siglas e Símbolos

WEB SERVICES -- Serviços oferecidos via rede utilizando tecnologias da Internet

GRID -- Arquitetura Distribuída, tambem conhecida como computação em grade

Cluster -- Arquitetura de hardware que busca unificar um conjunto de computadores

CORBA -- Common Object Request Broker Arquiteture

Peer-to-peer -- Rede com comunicação direta entre clientes.

screen saver -- Software utilizado para prolongar a vida util dos monitores antigos

CPU -- Central Process Unit, peça de hardware responsavel pela maior parte do processamento de um computador.

WEB -- Curta referência a Internet

FTP -- File transfer Protocol, protocolo utilizado para transferir arquivos por uma rede.

HTML -- Hyper Text Makeup Language, linguagem para apresentação visual de informações muito utilizado na internet.

HTTP -- Hypertext transfer protocol

RPC -- Remote Procedure Call

XML --eXtensible Markup Language

RMI -- Remote Method Interface

WSDL -- Web Service Descitpion Language

UDDI -- Universal Desciptor

WorkStations -- Estação de trabalho.

Desktop -- Computador de mesa e de uso geral.

CGI -- Common Gatway Interface

JSP -- Java Server Page

SMTP --simple mail transport protocol

POP -- post office Protocol

SOAP -- Simple Object Acess Protocol.

TCP -- Transmission control protocol

UDP -- User datagram protocol

À MINHA FAMÍLIA, PELOS FUNDAMENTOS E ESTRUTURAS BASILARES QUE ME PERMITIRAM ALCANÇAR EXPRESSIVOS NÍVEIS DE CONHECIMENTO ACADÊMICOS.

AO ORIENTADOR VINOD, PELAS OPORTUNIDADES DE APRENDIZAGEM E POR ORIENTAR ESTA MONOGRAFIA.

A JULIANA HUANG, PELA AMIZADE E PELO CONHECIMENTO DA LÍNGUA PORTUGUESA UTILIZADO NESTE TRABALHO.

Resumo

Esta monografia visa a apresentar principalmente uma experiência no uso do Web Service em Grid, parte de um projeto de pesquisa sobre Grid, realizada no Laboratório de Pós-Graduação, do Instituto da Computação, na Universidade Federal Fluminense.

Ressalta-se que esta pesquisa, em particular, possui como finalidade a elaboração de um Portal, para facilitar o acesso e a utilização de alto poder computacional pelos usuários, tornando transparente problemas enfrentados atualmente, como segurança, comunicação e administração do Grid.

Dessa forma, a presente obra traz uma visão preliminar de Computação em Grid, de seu comparativo com outras formas de computação distribuída, de Web Services e da evolução desse trabalho.

CAPÍTULO 1 – INTRODUÇÃO

Há de se convir que o principal objetivo do projeto final corresponde à exposição dos conhecimentos adquiridos durante a vida acadêmica de forma personalizada. Diante dessa premissa, elaborou-se o presente trabalho, com meta de apresentar a aplicação de inúmeros conceitos de várias matérias, bem como a sua conjugação inter-disciplinar, aliado à atenuação da necessidade de pesquisa no ambiente universitário. Nesse sentido, essa obra, além de exibir aspectos do amadurecimento de conhecimentos computacionais, visará ajudar na resolução de um problema real, que consiste na dificuldade da utilização de um Grid como facilitar o acesso , executar aplicações e gerenciamento dos processos.

Preliminarmente, menciona-se que o projeto se desenvolveu durante uma pesquisa sobre Grid, realizada no Laboratório de Pós-Graduação, do Instituto da Computação, na Universidade Federal Fluminense. Ressalta-se que este projeto, em particular, possui como finalidade a elaboração de um Portal, para facilitar o acesso dos usuários que precisam de elevado nível de poder computacional. Para tal, deve-se desenvolver todo um sistema de suporte para interfaces com o usuário, de forma que esta interface possa simplificar o acesso e a utilização, tornando transparente diversos problemas que se enfrentam atualmente, como segurança, comunicação e administração do Grid.

Cabe esclarecer que a denominação Portal se deve à sua funcionalidade, que é a criação de um ponto de acesso a vários recursos. No projeto, o Portal toma forma de uma camada entre duas partes, de modo a servir como intermediário entre os programas de apoio aos usuários e os recursos oferecidos pelo Grid.

O portal oferecerá aos programas dos usuários um conjunto de serviços que, por sua vez, são convertidos em uma série de ações particulares no Grid. Acrescenta-se que a

vantagem do uso de serviços, em relação à utilização direta dessas ações pelo programa do usuário, consiste na simplicidade do uso do serviço, reduzindo a complexidade da primeira, desde a forma que essas ações são tomadas bem como as peculiaridades tomadas pelas ações em relação a cada Grid.

Numa primeira visão, entende-se por Grid como uma forma mais avançada de computação distribuída. Nota-se que um conjunto de tecnologias forma o Grid, que se complementam para proporcionar um significante poder de computacional que, sem o mesmo, seria árduo ou, por vez, impossível de ser alcançado pelos custos de aquisição e de manutenção. Diante dessa situação, vislumbraremos que a tecnologia Grid possibilitará que um maior número de pessoas se tornem clientes de um elevado poder de processamento, poderoso o suficiente para atender às suas necessidades, por um custo considerado acessível.

Não raro, encontram-se computadores gastando uma significativa jornada do dia processando poucas informações, bem como executando aplicativos de trivial exigência computacional, como o tradicional "*screen saver*". Dessa forma, infere-se que esses equipamentos consomem energia sem, no entanto, gerar informações de relevante utilidade. Esse cenário análogo, habitualmente, ocorre ao se empregar um computador para a produção de tarefas ordinárias do cotidiano, sendo costumeiro precisarmos, na maior parte de tempo da duração do trabalho, apenas de uma pequena fração da capacidade de CPU e de memória.

Releva-se que a aquisição de tais equipamentos visa ao cumprimento de uma pequena quantidade de tarefas importantes de forma rápida e eficiente. Como um exemplo imaginário, tem-se um servidor WEB[w3c] que deve suportar as requisisões dos usuários, durante duas horas, período de maior movimento do dia. Contudo, excluindo-se essas duas horas, os recursos do servidor se mostram subutilizados.

Verificam-se que duas notórias realidades, de conhecimento comum, que forçam uma mudança nessa forma de aquisição de recursos, a saber: Uma economia frágil que pressiona, no sentido de se empregar os orçamentos de Tecnologia da Informação de modo a majorar a eficiência, bem como a imprescindibilidade de se realocar esses recursos de forma dinâmica e inteligente, conforme as mutáveis necessidades dos usuários.

Com intuito de solucionar essas questões, surgem, nas comunidades intelectuais, filosofias inovadoras de computação, como a mudança no paradigma de organização e utilização dos recursos disponíveis. Dentre o rol das mais promissoras repostas, destaca-se uma dupla dotada de peculiaridades: Computação em Grid[globus] e "Web Services"[w3c].

Serão apresentadas visões em nível elevado, ressaltando os fundamentos sobre essas tecnologias promissoras, mostrando comparações com outras propostas, todavia, mantendo um nível distante com as implementações particulares de cada uma. Esclare-se que apesar do conteúdo não ser detalhado ao leitor, certamente irá obter um conhecimento amplo, capaz de posicionar os papéis desempenhados por cada uma das técnicas, de maneira a possibilitar que, posteriormente, possam iniciar uma própria pesquisa de soluções para se colocar em prática. De fato, a vantagem da escolha pelo alto nível de abordagem, se evidencia pelo fato da rápida e constante evolução das soluções, o que tornaria, assim, qualquer exemplificação detalhista desatualiazada, o que não ocorre com os fundamentos basilares de paradigmas tecnológicas.

Nota-se que o vocábulo cliente, no presente texto, se refere ao desenvolvedor de uma aplicação, que apresenta carência por recursos como, por exemplo, alta capacidade de processamento e armazenamento.

Destaca-se, ainda, que os problemas do uso de Grid e do desenvolvimento do Portal se referem ao modo pelo qual as aplicações utilizam um Grid, iniciando-se pela sua compilação, execução e obtenção dos resultados, de uma forma mais simples do que a atual visto que, até então, realizava-se pelo uso de linhas de comandos e de ferramentas como FTP.

Entedemos como computação distribuida o uso de componentes como software e hardware não localizados em um único ponto ou geográficamente distantes, e que ainda assim, são capazes de trabalharem em conjunto para a obtenção de qualquer tipo de resposta aos nossos problemas, como exemplo atual a coleta de informação sobre um estoque de produtos de todas as filiais de uma empresa ou de uma molécula em diversos bancos de dados genéticos.

A presente monografia está organizada da seguinte forma:

Nos capítulos 2,3 e 4 serão apresentadas as idéias Grid e Web Services e, nos seguintes, relatos e explicações sobre o que foi realizado no projeto final.

CAPÍTULO 2 – Computação em Grid

Neste capítulo serão demonstrados fundamentos da computação em Grid, evidenciando os motivos que levaram ao surgimento do Grid. Observe que Grid pode ser visto de várias maneiras e, nesta monografia, foi escolhida uma visão mais ampla, de forma a permitir uma cobertura mais idealista do Grid.

Descobre-se na Computação em Grid uma revolucionadora arquitetura computacional distribuída, que recentemente vem se desenvolvendo no contexto de pesquisa, a fim de responder às crescentes demandas por recurso, quer seja pessoal ou por equipamentos. Inúmeras vezes, esse nível de necessidade se mostra tão elevado que poucas instituições de ensino e pesquisa possuem finaciamento para adquirí-los. Percebe-se que essa mesma necessidade tem sido impulsionada a altos patamares no setor empresarial, já que, nas companhias, se adota a tendência de trabalhar de forma conjunta, sob uma visão de integração global, em vistas de maximizarem a eficiência de seus negócios.

De modo a resolver esse objetivo, da eficiência sobre a utilização, projetou-se uma maneira que possibilitasse criar uma rede, para a colaboração de conhecimentos e de recursos computacionais, de tal forma que, um maior número de instituições, geograficamente distantes, fossem capazes de empregar todo o poder necessário para as suas atividades, de modo técnico e economicamente viável.

Cumpre destacar que as instituições de pesquisa se posicionam na linha de frente, no desenvolvimento dessa nova idéia, desencadeando as primeiras gerações de Grids. Forma-se, assim, uma base para que outros seguimentos possam ter o primeiro contato, bem como auxiliar a consolidação das técnicas.

Ressalta-se ainda que o Grid, numa perspectiva simplista, representa uma rede colaborativa de conhecimentos e capacidades, criada e implementada como uma nova arquitetura de computação distribuída. Isto se deve a certas novidades que oferece, como respostas às limitações, apresentadas no próximo capítulo, das arquiteturas anteriores como Cluster, CORBA[omg], RPC, RMI[sun-rmi] e redes "peer-to-peer".

Na contemporaneidade, a arquitetura do Grid atravessa a fase de definições de padrões pelos meios acadêmicos, que é de vital importância, para que o meio comercial possa utilizar de forma rentável essa arquitetura. E, por sua vez, essas empresas auxiliam o meio acadêmico a progredir com o desenvolvimento. Estes padrões também ajudam na criação de ferramentas de desenvolvimento, manuntenção e criação novas e mais abragentes de tal forma que mais aplicações estão passando a usar Grid como uma técnica de implementação. Compreende-se como ferramentas tudo aquilo que ajuda ao desenvolvedor na construção de sistemas, como editores de textos, compiladores, bibliotecas e literaturas.

2.1 Grid aumentando a eficiência

Nos últimos anos, tem se presenciado uma enorme distinção entre o ritmo de desenvolvimento das tecnologias de Rede e de desenvolvimento dos processadores. De fato, numa rápida comparação, a capacidade das Redes dobra a cada nove meses, enquanto que, a dos processadores, a cada dezoito, ratificando esse descompasso de aprimoramento, conforme visto no gráfico apresentado na Figura 1.

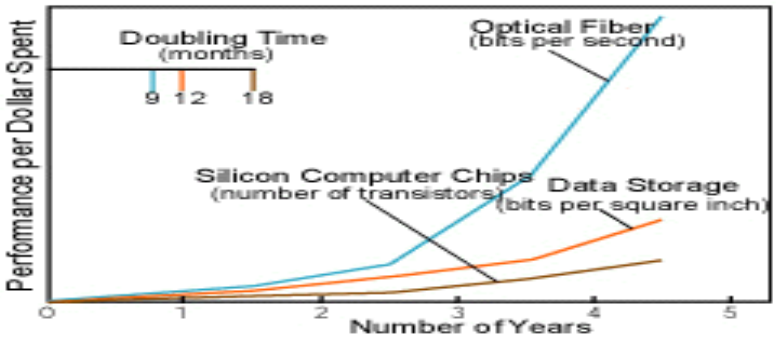

Figura 1

Performance per Dollar Spent = Desempenho por custo.

Optical Fiber = Fibra Ótica em bits, por segundo.

Silicon Computer Chips = Peça de silicio, por numero de transistors.

Data Storage = Capacidade de armazenamento, bits por "inch" ao quadrado de área.

Number of Years = Anos.

Doubling time = Tempo que demora para dobrar em meses.

Logo, com fins de atenuar essa lacuna, deve-se repensar o modelo vigente de utilização dos processadores, tendo em vista essas novas e avançadas redes. Encontra-se semelhante questão na relação das tecnologias de armazenamento com a dos processadores, que procovou a adoção de técnicas, de modo a compensar as deficiências dessas unidades.

Convém enunciar que, a Computação em Grid responde às carências de recursos, com a união virtual de vários processadores, para juntos alcançarem níveis inéditos de capacidades, impossíveis de se alcançar individualmente.

2.2 Por que usar computação em Grid.

Recentemente, diversas instituições e empresas passam por uma fase de reexame dos investimentos direcionados à área de tecnologia da informação, impelidas pelas constantes convulsões na saúde da economia global[compuworld]. Diante de tal quadro, infere-se que a máxima do mercado de adquirir novos equipamentos como forma de suprimento das necessidades crescentes de recursos, sofrer modificação, tendo como nova vertente a utilização dos equipamentos já existentes, permitindo o questionamento da eficiência na utilização. De modo a incrementar o retorno sobre os investimentos já efetivados na compra desses recursos, tendo como ganho vantagens sobre custos operacionais e reutilização da qualificação pessoal já adquiridos.

Com fins de se elevar o grau de aproveitamento dos recursos disponíveis na empresa, planeja-se um modelo promissor de alocação mais inteligente que, progressivamente, recebe um contingente maior de adeptos. Exemplifica-se uma política de alocação inteligente, que consiste em fornecer a quantidade de recursos para processos, de forma proporcional à sua produtividade e prioridaridade, otimizando a equação do ganho, que inclui a taxa de utilização dos recursos pelo tempo, bem como o retorno de cada processo produtivo, como variáveis. Em outras palavras, deseja-se atender preferencialmente os processos produtivos, que contribuem em muito para os resultados almejdos, além de manter a utilização dos recursos na sua totalidade, minimizando tempo em repouso dos computadores.

Pode-se entender por processo produtivo, tudo que precisa usar um recurso para obtenção de resultados como, por exemplo, uma aplicação científica, que simula reações quimicas ou um aplicativo, que realiza compras de ações no mercado, para clientes de grandes bancos.

Pode-se exemplificar como crescente necessidade a situações de algumas instituições que precisam de poder computacional para promoverem o uso de novas técnicas. Como exemplo uma industria petrolífera que desenvolveu um sistema que estima a localização dos melhores poços de petróleo no fundo do mar, mas que precisam de inumeras interações de algoritmos levando a uma grande necessidade de poder de processamento. Ou como no caso de institutos de fisica com aceleradores atômicos que produzem TeraBytesde informações [objetivity] em um tempo muito curto, que precisam ser armazenados e processados.

2.3 Tipos de Grid.

Salienta-se que a Computação em Grid busca resolver uma série de limitações de aplicações, fazendo com que cada tipo de Grid seja o resultado da tentativa de solucionar cada tipo de problema. A seguir, destacam-se três tipos de Grid, cabendo resaltar que estas não são mutuamente exclusivas, ou seja, podem ser combinadas.

- **Grid Computacional**

Este tipo de Grid procura oferecer uma grande quantidade de recursos de processamento, utilizando equipamentos de alta performance, de uso exclusivo das aplicações do Grid, que possui como exemplo atual o projeto TeraGrid que em 2004 foi capaz de oferecer 40 GigaFlops com 1 Petabyte de informações formado com recursos de cinco instituições norte americanas interligadas por uma rede capaz de transmitir 40 GigaBytes por segundo. Esses exemplo serve para mostrar ao leitor, o nivel de referência aos novos limites de poder computacional, pois tal capacidade não possui paralelos em sistemas localisados em um mesmo ambiente.

- **Grid de Oportunidade**

Já, este tipo de Grid, visa a aproveitar cada recurso não utilizado por um grupo de computadores não dedicados ao Grid, em que, normalmente, o proprietário de cada computador cede o controle do equipamento, pelo espaço de tempo que lhe for conveniente.

Usualmente, utiliza-se "WorkStations" para a construção desse Grid. Como exemplo temos o projeto Seti@home[seti] que oferece um "screen saver" que ao entrar em atividade, baixa um lote de dados de um site, que consiste em um bloco de sinais do espaço captados por uma antena especial, e utilizando alguns algoritmos procura por sinais de vida inteligênte nesses sinais. O que mostra uma forma financeiramente leve de se obter um grande poder

computacional, pois foram necessários investimentos apenas no desenvolvimento da aplicação "screen saver" e a aquisição de um servidor para disponibilisar os dados, tendo como de forma gratuita todos os computadores dos usuários de tal aplicação, de tal forma, que se fossem de uso exclusivo do Seti, seriam muitos caros.

- **Grid de Dados**

Um Grid de dados objetiva oferecer um serviço de armazenamento de informações entre diversas instituições e empresas que desejam compartilhar informações de forma segura e independentemente do lugar em que se encontram armazenadas. Habitualmente, se encontra uma mistura, de conceitos deste tipo de Grid, com redes peer-to-peer, visto que as duas arquiteturas objetivam à resolução de problemas parecidos.

Adiante, se analisa um comparativo entre as ambas, para mostrar as suas similaridades e suas distinções.

CAPÍTULO 3 - Grid e as formas anteriores de computação distribuída

Grid pode ser diferenciado das outras formas de computação distribuída, basicamente, pela sua parte mais fundamental, que corresponde à utilização eficiente de recursos distribuídos, heterogêneos, com diversas politicas de administração e com interligações fragéis, além de se mostrarem intermitentes entre os diversos participantes.

Como Grid pode ser focado de várias maneiras, é comum encontrar comparativos de Grid com as tecnologias anteriores utilizadas para se obter uma computação distribuída. Dispõem-se, a seguir, algumas análises que normalmente surgem quando a noção de Grid é apresentada, esses procuram trazer à tona as mudanças resultantes das idéias trazidas pelo Grid.

3.1 Como a computação em Grid se difere da computação em Cluster

Computação em cluster corresponde a um tipo bastante limitado de computação distribuída, sendo comum se confundir com Grid.

Elenca-se, primeiramente, a distinção de recursos com que cada um se mostra capaz a utilizar, Grid tenta abranger o maior número de recursos disponíveis, homegênios ou não, enquanto que o Cluster normalmente é capaz de utilizar de forma eficiente recursos homogênios. Entende-se como características que diferenciam um tipo de recurso, toda uma gama de necessidades de operação de cada um possui, como sistemas operacionais, fabricante dos componentes, versões de bibliotecas e politicas de administração e de acesso.

Vale acrescentar que se pode utilizar recursos heterogêneos em um Cluster, todavia, devido a sua natureza, as particularidades de cada recurso pouco influienciam no funcionamento global, isto é, um Cluster não se adapta de forma dinâmica e eficiente nesse modelo de utilização. Encontram-se implementações de Cluster que adotam um mesmo grupo de tecnologia em todos os pontos, vinculando, assim, a adição posterior de recursos à implementação dessas técnicas, onerando o custo da expansão do Cluster.

Por sua vez, Grids computacionais se mostram tão amigáveis, pela sua facilidade de incorporação de recursos, de modo que um ou mais Clusters podem ser enxergados como uma fonte adicional de recurso ao Grid. De fato, os usuários de Grid podem se utilizar de Cluster sem terem conhecimento disso, posto que o Grid trata de forma eficiente as suas fontes de poder computacional e, em se tratando de um Cluster, aproveita a sua caracterísiteca marcante, a de processamento altamente acoplado, empregando-a adequadamente.

Verifica-se outra relevante distinção em relação à distância e disposição em que esses recursos localizam-se fisicamente. Em Grids os recursos podem estar em qualquer lugar, quer seja na mesma sala ou em regiões distantes, bastando apenas a existência de uma rede conectando-os. E no lado do Cluster, os recursos devem permanecer próximos, ligados por uma rede de alto desempenho, sobre a mesma administração, e devem permanecer disponíveis de forma pré-estabelecida afim de se obter o máximo de performance. Percebe-se que, essa capacidade do Grid em usar recursos dispersos, se mostra altamente depedente da evolução tecnológica da redes de comunicação que as ligam. Com esse poder, Grid possui uma escalabilidade superior à de um Cluster, oferecendo assim, um novo paradigma de possíveis soluções aos problemas extremamente complexos e importantes.

3.2 Grid e CORBA

Observam-se que outras tecnologias poderiam tomar o lugar do CORBA, como MPI, RMI, COM[microsoft] e DCOM[microsoft]. Contudo CORBA, dentre esses, por possuir um modelo mais desenvolvido, tendo como característica a independência de plataforma e de linguagens de implementação, corresponde à que mais se assemelha ao que Grid se propõe a fornecer e utilizar.

Apesar de Grid ser uma arquitetura que abrange questões de arquiteturas de hardware e de software, e CORBA como uma forma de comunicação entre componentes distribuidos, os dois são comumente confrontados pelas pessoas como concorrentes, mas o que será

demonstrado é que Grid vai muito além do que CORBA procura resolver.

Pode-se considerar CORBA como uma arquitetura de comunicação que permite o desenvolvimento de aplicações distribuídas, compostas por diversos objetos que realizam uma rede de comunicação, com a finalidade de unificar capacidades e recursos dispersos fisicamente. Observa-se que CORBA representa uma tecnologia já estabelecida, possuindo uma extensa base instalada, incluindo projetos de interface gráfica para usuários como o GNOME[brlinux].

Por definição, CORBA promove o uso de orientação a objetos em todos os níveis, entretanto é possivel o uso de CORBA em linguagens que não oferecem tal paradigma. Nesse sentido, tem-se como exemplo notável a linguagem C, todavia neste caso se perdem algumas características oferecidas pelo CORBA como o polimorfismo. De fato, a adoção de orientação a objetos em CORBA se encontra tanto na implementação de componente ou recurso, bem como na forma pela qual se realizam as comunicações. E esse consiste em uma das principais diferenças entre CORBA e Grid, em que a necessidade de se melhor aproveitar o maquinário existente, novamente, induz ao Grid à utilização de tecnologias que não imponham filosofias de implementação, já que reduzem o contingente desses recursos abrangidos. Portanto, Grid não exige a adoção de orientação a objetos a todo o instante.

Atualmente, Grid está substituindo o papel da orientação a objetos na comunicação que possui em CORBA, oferecendo, como alternativa, a orientação a serviços, que representa uma forma de se obter nos recursos a fonte de respostas aos problemas. Entende-se por orientação a serviços como a forma pela qual se obtém os resultados, por exemplo, na orietação a objetos, se enxerga o recurso A como objeto A, além de cada capacidade do recurso A como métodos do objeto. Logo para se fazer uma tarefa com o recurso é necessário realizar sequências de chamadas de métodos, o que difere da orientação a serviços, em que a realização da mesma tarefa se dá uma única vez através de um serviço, deixando que o recurso realize toda a sequência de operações necessárias para o cumprimento dessa tarefa.

Por último, observa-se que a orientação a serviços influencia o modo com o qual os componentes permanecem ligados. No CORBA com a orientação a objetos, permanecem conectados de forma mais rígida e acopladas, entretanto, com Grid, se mostram indepedentes e maleáveis. Isto se deve, mais uma vez, à necessidade do Grid de reduzir as barreiras para melhor absorção de recursos.

3.3 Grid e Peer-to-peer

Boa parte das questões que Grid tenta suprir já se perduram nos meios tecnológicos há bastante tempo, de tal forma que já existem soluções para algumas. Dentre essas questões se destaca a necessidade de compartilhar pedaços individuais e indepedentes de informação, de modo que essas informações possam ser acessadas, replicadas e transferidas para todos que a desejarem, pela maneira mais fácil e com poucas restrições.

Redes Peer-to-Peer é uma das soluções bastante adotada, que corresponde ao compartilhamento de arquivos diversos, entre uma vasta quantidade de clientes fisicamente dispersos e interconectados por uma rede frágil, com baixa capacidade de transmisão.

Uma importante habilidade que as redes Peer-to-Peer estão adquirindo consiste na independência de serviços de controle centralizados, diminuindo, assim, as possibilidades de parada de funcionamento por falha em um único ponto, além de aumentar a escalabilidade, dado que o crescimento da parcela de participantes na rede depende da capacidade de gerenciamento de quem suporta o controle. Destaca-se um exemplo da fragilidade do controle centralizado, o caso Napster, que teve o serviço paralisado em razão de uma questão judicial. Nota-se que, nessa ocasião, bastou apenas o desligamento de alguns servidores para que todo o sistema de compartilhamento oferecido ficasse paralisado.

Em contraste, Grid comumente se utiliza de um controle centralizado, para garantir que somente clientes autorizados, sejam pessoas ou outros equipamentos, possam usá-lo. Nota-se que essa centralização não vincula o seu processamento a apenas um equipamento, sendo possível o uso do próprio Grid na distribuição dessa tarefa em vários pontos, coordenando que todos trabalhem em conjunto.

Tanto Grid como Peer-to-Peer detêm a capacidade de manipular recursos de maneira bastante flexível, podendo ser incorporados ou removidos em qualquer instante. Todavia, no caso específico do Peer-to-Peer, pode-se atingir a um patamar de utilização elevadíssimo, que se traduz, por exemplo, na capacidade de se aproveitar inúmeros recursos, mesmo aqueles dotados de curto de tempo de existência.

No tocante à variedade de recursos, Grid e Peer-to-Peer manipulam tipos distintos. Distingue-se em Peer-to-Peer, em princípio, duas espécies: a dos fornecedores de arquivos e a dos que possuem um catálogo de listagem dos anteriores. Quanto a Grid, esses recursos correspondem aos de qualquer modelo de equipamento com capacidade de processar informações ou que as forneçam. Nesse sentido, percebe-se que redes Peer-to-Peer se aplicam como uma fonte adicional de material para o Grid, agregando, assim, as capacidades e as vantagens particulares de todos.

3.4 Computação em Grid e Web Services

Como Grid busca extrair, otimizadamente, proveito das diversas máquinárias dispersas, imprescinde, assim, que seja ínfimo o limite de entraves na comunicação entre os componentes. Conforme já exposto, CORBA, apesar de seu perfil direcionado à computação distribuída, carrega consigo restrições que o invalidam na sua aplicabilidade total em Grid. Portanto, uma nova tecnologia de comunicação deve ser selecionada como técnica de comunicação e, nesse momento, Web Service se apresenta como um promissor candidato.

Observa-se que, inicialmente, Web Service não foi desenvolvido com fins de aplicabilidade em Grid, contudo, se percebe, trivialmente, a semelhança guardada no modo que cada um promove as comunicações entre as suas partes. No Grid, dois componentes trocam informações e, no Web Service, um cliente envia uma requisição à um servidor que, por sua vez, devolve uma resposta.

Como Web Services foi projetado para funcionar como meio de ligação entre duas partes, cuja tecnologia de implementação essas desconhem, sem, no entanto, impossibilitar o compartilhamento de recursos individuais.

O próximo capítulo oferecerá maior detalhamento do Web Services.

CAPÍTULO 4 – WEB SERVICES

Antes de se iniciar o desenvolvimento da filosofia por trás do "Web Service" deve-se esclarecer uma importante distinção entre "Web Services" e serviços Web, que podem parecer iguais, porém referem-se a diferentes objetos: o primeiro é uma forma de comunicação e uso de métodos remotos, sendo que o segundo se representa nas facilidades fornecidas pela internet, como páginas HTML, arquivos diversos, bate-papo e email.

Nos primórdios, pensou-se numa idéia simplória, pela qual dois componentes pudessem se comunicar, que se materializou em RPC (Remote Procedure Call), a denomida remota de procedimento, usando interfaces padronizadas em C e estruturas de dados representados pelo XDR(representação de dados externos) e, portanto, fornecia uma forma básica e limitada de intercomunicação em rede local. Sob a ótica dos desenvolvedores, sua estruturação, em grande parcela, se assemelha às chamadas de procedimentos do paradigma programação estruturada, em que blocos de código, tratados como um conjunto, realizam uma única função, que se apresenta referenciada por um nome comum.

No decorrer dos anos, com as pesquisas e a utilização prática, no coditiano de diversos círculos de desenvolvedores, outros modelos evoluíram a partir de RPC, como CORBA[omg] e RMI[sun-rmi] (em java). Todavia, em geral, manteve-se uma ligação forte entre os seus componentes.

Percebe-se que, recentemente, com o desenvolvimento e a popularização da Internet, as pessoas passaram a ter oportunidade de usar um simples cliente para o acesso de informações no formato HTML, de qualquer servidor Web, em qualquer localização, bastando apenas este estar conectado à rede internacional. Porém, tanto os servidores quanto os clientes podem ser implementados em qualquer dispositivo, sistema operacional e

linguagem, necessitando, tão somente, que cada um tenha a capacidade de adotar o protocolo HTTP para requisitar e responder por informações, interpretando HTML, o convencionado para descrevê-las.

Com tal observação , o próximo passo se concretizou na utilização dessas tecnologias da Internet, para possibilitar que as informações passasem todas a ser manipuladas por programas e, não mais, por seres humanos. Nessa linha de desenvolvimento, o pioneiro a surgir neste novo âmbito corresponde ao XML-RPC, um protocolo simples, construído para permitir uma invocação remota de um método, de maneira simples, que rapidamente evoluiu para SOAP (protocolo de acesso a objetos simples), permitindo um uso mais completo de recursos remotos, bem como dados mais complexos, tratamento de exeções e filosofia de utilização.

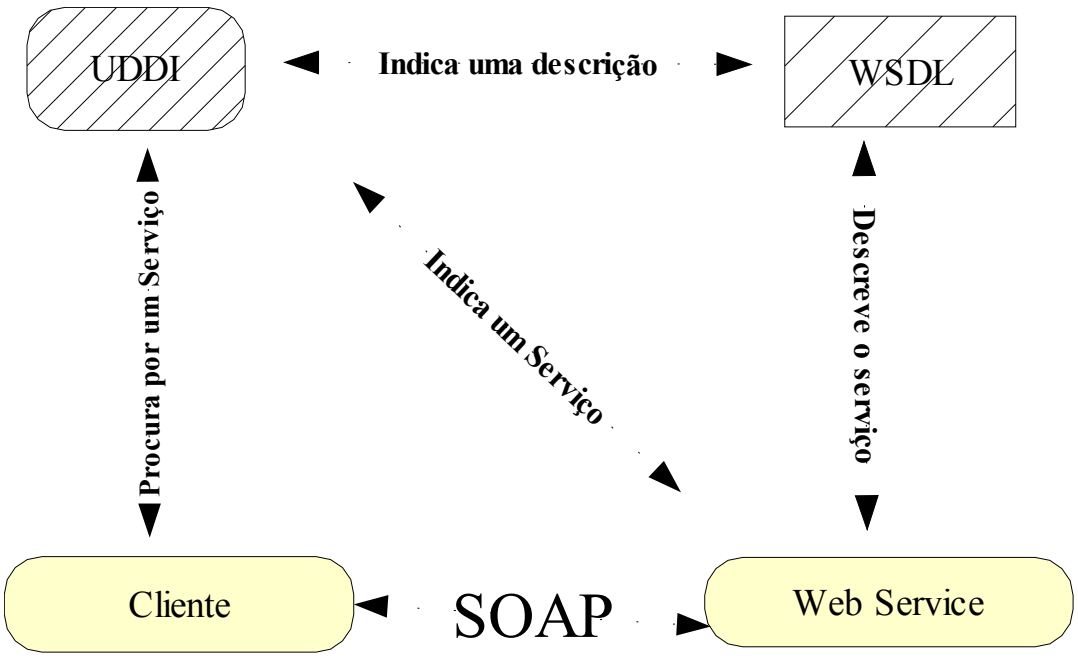

Figura 2

- UDDI – Diretório de serviços.
- WSDL – Descrição de um serviço.
- SOAP – Uso dos serviços.

Compreende-se, como SOAP, um protocolo leve para troca de informações em um ambiente distribuído. Esse consiste em três partes :

- uma descrição do conteúdo e do uso correspodente.
- um conjunto de regras de codificação para representar os dados.
- uma convenção de como promover chamadas remotas e suas respectivas respostas .

Cumpre relevar a participação de SOAP, em conjunto com outras tecnologias, como WSDL, que decreve um Web Service e, UDDI, que funciona como um catálogo de serviços, representam os alicerces basilares do Web Service. Acrescenta-se, ainda, que todas apresentam a sua contrução voltada para a finalidade de serem empregadas por qualquer servidor web, que se mostre capaz de manipular conteúdo dininâmico, como página do tipo PHP , CGI e JSP.

Indubitavelmente, essa flexibilidade abre oportunidade para a possibilidade de incluisão do suporte a Web Service à qualquer aplicação existente, para tal se requisita somente a elaboração de um nível de abstração em torno das chamadas existentes.

Ressalta-se o Web Service demonstra flexibilidade na maneira em que realiza o tráfego das informações, posto que se pode escolher protocolos como o HTTP, FTP, SMTP, ou algo personalizado para transportar o pacote SOAP. Percebe-se que a vantagem da adoção de alguns desses protocolos corresponde à utilização de sistemas de segurança atuais, em locais onde já existem serviços Web, que têm como exemplos: servidores de páginas, email e arquivos.

De fato, o reuso de protcolos é vantajoso, visto que as suas regras de segurança já se encontram definidas de forma a permitir o funcionamento dos serviços existentes que os utilizam. Dessa forma, ganha-se facilidade na implementação da segurança para os novos sistemas em Web Service.

Com o objetivo de clarificar alguns conceitos que norteiam Web Services, será destacado pela figura as camadas mais comuns.

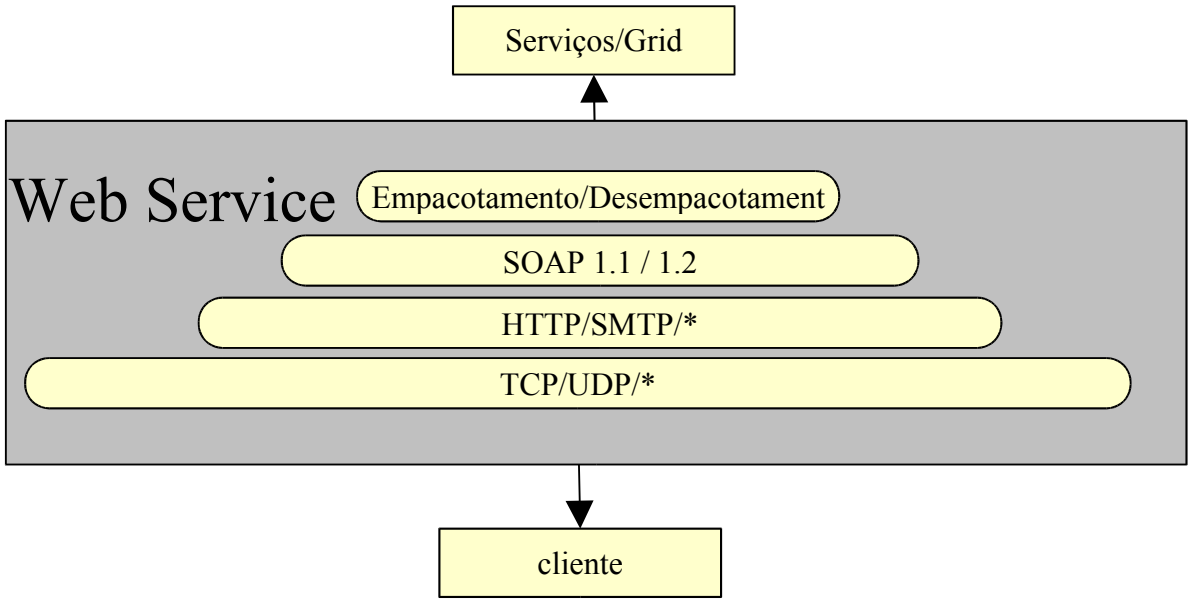

Figura 3

Segue uma analogia simplificadora das camadas internas do Web Service:

1. TCP/UDP/qualquer protocolo de comunicação – Carteiro.
2. HTTP/STMP/qualquer protocolo de requisição – Envelope.
3. SOAP – Carta com texto.
4. Empacotamento / Desempacotamento – As informações da carta adquiridas pela leitura ou escrita do texto.

Muito tem se comentado, ultimamente, sobre a performace do Web Services depois dos primeiros casos de uso real, em ambientes de produção, que mostraram retorno abaixo do nível esperado, deixando a desejar. No entanto, tal conclusão carece ser observada e analisada com cuidado, de forma a se procurar os motivos e as soluções. Com o resultado de tal trabalho, é facil verificar alguns motivos importantes, como por exemplo, o modo pelo

qual se usa o Web Service.

Esta forma de uso se origina do modelo pelo qual o Web Service evoluiu, o RPC, que, por sua vez, se utilizava das chamadas de procedimentos e, para a realização de alguma tarefa, fazia-se preciso várias interações entre os procedimentos. Esse comportamento, normalmente, ocorria em sistemas interligados por uma rede local com uma boa performance, logo o custo de comunicação entre cada equipamento se mostrava controlado e previsível. Em outras palavras, apesar de a chamada a um método remoto exigir um espaço de tempo maior do que a invocação do mesmo método localmente, isso não impossibilitava no final a obtenção de bons resultados, tendo em vista o ganho de se usar uma maior quantidade de recursos, que não seriam limitados a serem locais, podendo ser utilizados os distribuídos. Acrescenta-se que o impacto sobre o custo de se usar recursos remotos poderia ser minimizado, dado que normalmente se conhece esse custo e, com a adição de algumas mudanças, é possível diluir a perda de tempo do acesso.

Com o Web Service é muito diferente, pois se a rede de conexão for de baixa qualidade, como a Internet, o custo de se chamar uma sequência de procedimentos pode ser muito alto e imprevisível. Para resolver tais problemas cabe destacar a segunda palavra, Services, e com isso, repensar no modo de exposição das funcionalidades que desejamos tornar disponíveis a outros sistemas. Cabe, assim, agrupar todo o conjunto de chamadas que eram usadas pelo RPC, para realizar uma determinada tarefa, em uma única chamada Web Service, a fim de evitar o uso excessivo da rede visto que cada chamada gera acesso a rede, diminuindo assim o impacto negativo da baixa performance das redes inteligadas aos recursos.

Sintetiza-se, abaixo, a vantagens que podem ser obtidas do emprego de Grid com Web Services

- Maior compatibilidade, transmição das informações são codificados em texto.
- Melhor flexibilidade
- Compartilhamento de dados complexos entre as diversas plataformas de hardware e sistemas operacionais de forma mais fácil e produtiva.
- Possibilidade de usarmos a infra-estrutura atual de Internet para construirmos os novos serviços.
- Facilidade com relação aos sistemas existentes de segurança.

- Ao contrário de modelos como CORBA e DCOM cada parte, na prática, não precisa utilizar as mesmas implementações das ferramentas de comunicação, como o ORB no caso do CORBA, pode-se ter um ambiente heterogêneo nas ferramentas que se emprega para promover a comunicação entre os componentes.
- Uma arquitetura simples que possuiu um modelo relativamente pequeno, com o objetivo de apenas resolver as dificuldades mais comuns da computação distribuída.

Cabe destacar, ainda, o valor dessa última vantagem que resulta da tendência de se preferir manusear algo que possui poucos componentes muito úteis e reutilizáveis, acrescentando, posteriormente, somente as funcionalidades que realmente forem importantes ao caso concreto.

Ainda que extender os componentes possa parecer, numa primeira análise, trabalhoso, representa pouco esforço, se comparado com o incoveniente de termos objetos inúteis que vieram a reboque. Isso se evidencia, ainda mais, caso se considere o fato do Grid demandar uma grande quantidade de mão-de-obra para o seu desenvolvimento, muito superior ao exigido na adequação de componentes do caso anterior.

CAPÍTULO 5 - PRIMEIROS PASSOS DO PROJETO

Neste capitulo, numa visão genérica, relatam-se as realizações das fases do projeto final. Escolheu-se uma forma evolutiva, mostrando as tentativas de implementação, abordando as idéias que originaram cada passo e seus respectivos resultados. Dessa forma, abre-se a oportunidade ao leitor de escolher um dos caminhos já tentados. Porém, antes de iniciar os relatos de cada passo, se faz necessário destacar os preparativos que foram imprescindíveis de modo a possibilitar o início do desenvolvimento do Portal em si.

Durante o período foram realizadas atividades de pesquisa que permitiram a motivação, bem como a elaboração desse projeto. Para tal, iniciou-se um estudo de alto nível em diversas tecnologias, dedicando-se uma atenção reforçada para as mais recentes e promissoras, alia-se, ainda, o desejo de se adquirir conhecimentos valiosos para o mercado de trabalho em um futuro próximo.

Estas duas finalidades, a de procurar conhecimentos para ajudar no projeto e a do futuro profissional , em muitos momentos, entravam em conflitos, uma vez que algumas tecnologias interessantes não se mostravam adequadas ao projeto, isto é, em muitos momentos esse trabalho necessitava de uma solução particular, entretanto, no ponto de visto deste autor, pouco interessante no mercado de trabalho nacional, fato observado pela leitura de diversos artigos sobre programação encontrados na Internet e de anúncios das oportunidades de emprego em diversos meios.

No projeto se concentra uma série de desafios, sendo alguns listados abaixo:

5.1 Maturidade:

Como a tecnologia GRID é muito recente, além de apresentar um baixo grau de maturidade, apresenta uma série de problemas igualmente novos e particulares que foram poucos explorados. Em muitos momentos, encontrar material que ajudasse às suas resoluções se mostrou uma tarefa árdua.

Destacando alguns dos problemas, a variedade de formas que o Grid pode ser usado, administrado e desenvolvido. Dificultam o desenvolvimento do Portal, pois este de alguma forma, deve encontrar um modo de ligar um Grid aos seus usuários.

5.2 Heterogeneidade:

Heterogeneidade é um dos "efeitos colaterais" comum do Grid que busca incorporar o maior número de recursos disponíveis.

Inúmeras vezes, soluções foram abandonadas por serem aplicáveis a poucos sistemas. Como por exemplo o uso de ferramentas que trabalham com apenas um sistema operacional, visto que deveriam apresentar como característica fundamental serem genéricas.

5.3 Recursos Utilizados:

Tendo em vista este projeto ser uma parte integrante do EasyGrid, o primeiro ficou limitado aos recursos utilizados pelo segundo. Essa limitação foi um fator determinante na escolha do ambiente de desenvolvimento e de produção.

5.4 Recursos: Ambiente de desenvolvimento

Como o sistema operacional oferecido pelo laboratório de pós-graduação, que hospeda

o projeto EasyGrid e, por sua vez, hospeda este projeto final , corresponde ao GNU/Linux, toda a escolha das ferramentas deveriam funcionar nesse sistema operacional. Uma das características do sistema GNU/Linux consiste na pouca quantidade ou qualidade de ferramentas gráficas, obrigando, assim, os desenvolvedores a apresentarem um alto nível de domínio sobre todos os passos do desenvolvimento e manutenção do sistema em produção, provocando uma curva de aprendizagem de lento crescimento.

5.5 Ambiente de produção:

Cabe esclarecer como ambiente de produção o conjunto de hardware e software que será desenvolvido e utilizado a fim de dar vida ao projeto no momento que este alcançar o estado maduro o suficiente para ser utilizado pelos usuário do Grid.

O projeto EasyGrid não possuía nenhum recurso para o financiamento de software na época em que este projeto final foi iniciado, traduzindo-se num fator de bastante relevância na escolha do servidor de aplicações, que é a combinação entre hardware e software que buscam oferecer um sistema de suporte as aplicações, que no caso deste projeto seria o portal.

Tendo em consideração as condições apresentadas acima, as seguintes ações foram tomadas:

5.6 Escolha da distribuição do Sistema operacional:

Uma vez que o sistema operacional já foi determinado, como sendo o GNU/Linux, restou apenas determinar a escolha de uma distribuição Linux. Salienta-se que essas escolhas se devem ao fato deste projeto e o EasyGrid se encontrarem hospedados no laboratório da pós-graduação e, como esse local é freqüentado por outros alunos, que compreendem os de mestrado, doutorado e pesquisa, o sistema escolhido deveria também ser adequado para esses alunos. Outro fato determinante, é o fato das ferramentas utilizadas pelo EasyGrid, por hora, somente funcionam neste sistema operacional.

Não houve a escolha de apenas uma distribuição, mas sim, de duas - o sistema Red Hat 7.3 e uma variante, o Mandrake 9.1. O primeiro sistema optado foi Red Hat, por sua

ampla aceitação no mundo, sendo inclusive, base de diversas outras distribuições. O segundo sistema, o Mandrake, é umas das distribuições que se origina do Red Hat, porém este expandiu as características de sistema Desktop de modo mais maduro que a distribuição anterior, tornando-se, assim, uma excelente escolha para o laboratório.

Cumpre destacar algumas habilidades do Mandrake, como por exemplo a capacidade de gerenciamento mais avançado dos pacotes e de suas dependências, seleção de programas que o acompanham, uma grande comunidade de usuários que oferecem dicas e pacotes específicos, além de configurações padrões do ambiente gráfico melhores que o Red Hat.

5.7 Escolha da Ferramenta/Linguagem de desenvolvimento:

O próximo passo consiste na escolha da dupla: ferramenta e linguagem de desenvolvimento. Deve-se considerar a possibilidade de ser executados em sistemas heterogêneos, a flexibilidade deve ser um dos quesitos fundamentais.

Como ponto de partida, optou-se uma ferramenta de desenvolvimento familiar, o Kylix[Borland] 2.0. Essa possui uma característica vantajosa, que corresponde à sua facilidade de programação ser baseada em bibliotecas, podendo ser utilizadas em diversos sistemas operacionais, como o Windows[Microsoft] e Linux[kernel.org].

Esta ferramenta foi selecionada como ponto de partida, visto que estava em rápido desenvolvimento, além de já possuir uma forma madura, bem como boas características, a saber: a possibilidade de utilização da tecnologia de Web Services, uma interface de desenvolvimento integrada e a utilização de uma linguagem muito familiar para o desenvolvedor do projeto.

Uma outra opção seria utilizar a linguagem C++, o que no primeiro momento seria uma escolha mais natural para um projeto cujos clientes utilizarão programas escritos nesta mesma linguagem. Contudo, a linguagem foi abandonada pela falta de ferramentas de desenvolvimento que facilitassem a sua utilização e pela falta de intimidade do desenvolvedor do projeto com a linguagem, aliado ao fato da sua tendência futura de estagnação no mercado.

A terceira opção corresponde à linguagem Java, que possuía ótimas qualidades, dentro delas se pode destacar: habilidade de trabalho em sistemas heterogêneos, bastante promissor no mercado, disponibilidade de versáteis ferramentas de desenvolvimento e capacidade de trabalhar com Web Services. Em contrapartida, a linguagem Java não foi escolhida, por dificuldades de se encontrar as ferramentas no Linux que fossem preparadas para Web Services.

5.8 Arquitetura construída durante os primeiros passos.

•5.8.1 Cliente magro / Servidor Web

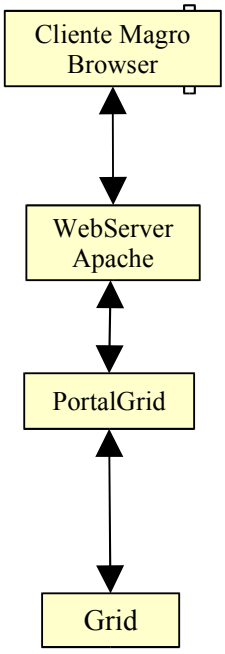

Figura 4

Na primeira tentativa de desenvolvimento do Portal, se utilizou de uma arquitetura bastante popular no âmbito da Internet, que corresponde ao uso de um "Browser": um tipo de cliente magro, cujo adjetivo magro significa que o programa cliente possui código apenas de exibição de informações, e de um servidor Web de páginas dinâmicas. Basicamente, o Portal é representado por uma página, cujo conteúdo seria produzido por um módulo, ligado como

uma extensão ao servidor de páginas. Acrescenta-se que as ações do Portal no Grid são realizadas na forma de chamadas de sistema por programas utilitários do Grid.

Para realizar tal ligação, optou-se pela tecnologia CGI (Common Gatway Interface) que é um modo pelo qual o Servidor de páginas se comunica com os programas externos, de forma que esses possam tomar parte no processamento de uma requisição de um cliente, oferecendo, assim, uma maior capacidade às tarefas que respondem a essas requisições. Note que se entende por capacidade tudo aquilo que as tarefas precisam ser capazes para realizar suas funções, como acessar um banco de dados, transferir um arquivo ou controlar algum processo do sistema operacional. Logo, o servidor era configurado para repassar todas as requisições cujo URL (Universal Resource Locator) apresentasse como final o nome do Portal, para uma aplicação externa que efetivamente ofereceria os serviços do Portal em Web Services.

A forma pela qual o programa externo oferecia o serviço compreende-se de:

- Recepção do pacote HTTP/SOAP repassado pelo servidor.
- Construção de objetos que armazenavam as informações contidas no pacote.
- Chamada de um método invocado pelo pedido dentro do pacote, repassando os objetos recém construídos como parâmetros.
- Processo do pedido
- Criação de uma resposta
- Conversão da resposta em um pacote HTTP/SOAP
- Envio do pacote de resposta para o servidor.
- Envio do pacote do servidor para o cliente, cujo pedido deu origem a esse processo.

Encontram-se pontos positivos, abaixo listados:

- Simplicidade de implementação.

Para utilizar essa arquitetura bastava estender uma classe oferecida pelo Kylix. Tal classe era responsável por traduzir as informações entre o servidor Web e o Portal através de objetos passados como parâmetros.

- Uso da estrutura atual de servidores Web.

Foi utilizado o servidor Apache[apache.org], um servidor de uso livre

possuidor de uma vasta gama de documentação, realidade esta de muita utilidade para o esclarecimento das capacidades e das limitações de uma aplicação CGI, como as que o Portal teria ao usar o servidor Web Apache.

- Utilizar Clientes já disponíveis, como um Browser comum de Internet.

Como clientes capazes de acessar o Portal, destacam-se : Netscape, Internet Explorer e Opera.

Entretanto, deve se considerar os pontos negativos a seguir:

- As técnicas de implementação das bibliotecas gráficas do Kylix não funcionam sem o uso de um servidor gráfico,no mundo Unix conhecido como modo headless, que deve estar ativo e com uma configuração de segurança especial, na máquina que hospeda o Portal. A necessidade de se utilizar essas bibliotecas gráficas se devia ao objetivo de gerar gráficos dinâmicos com diversas informações, como por exemplo mostrar um histórico dos recursos utilizados pelo aplicativo do cliente.

- Vulnerabilidade ao servidor

Causada pela forma a qual as requisições são encaminhadas do servidor Apache para os programas externos, visto que não há como garantir que uma requisição será atendida pela mesma instância do Portal.

Tal fato se deve à realidade habitual dos servidores serem implementados com múltiplos processos filhos, podendo cada um ter uma própria instância do Portal. Isto tem como resultado um aumento da escalabilidade, entretanto, como o projeto precisa de um ponto de controle central, a fim de se permitir a manutenção dos estados das aplicações de cada cliente entre as requisições, ter vários processos independentes faz com que cumprir tal necessidade mais difícil.

Convém mencionar que obrigar ao servidor apache a manter apenas um processo filho é desencorajado pelos desenvolvedores do servidor, dado que isso acarretaria na perda da escalabilidade, performance e estabilidade, já que, anteriormente, cada processo filho era recriado em certas condições, limpando assim a memória e evitando qualquer desperdício.

Elaborou-se um protótipo por meio dos resultados desta arquitetura, que consiste numa pagina web dinâmica, capaz de executar comandos do servidor, listar arquivos, transferir os arquivos do servidor para o cliente e a criar figuras, que simulam a representação das dependências dos programas do cliente. Destaca-se, ainda, que a criação dessas figuras expuseram algumas fragilidades das bibliotecas gráficas do Kylix, que dificultavam a realização confiável desse trabalho.

Como conclusão desta arquitetura, tem-se que :

- O Kylix oferece uma forma fácil de usar Web Services utilizando o servidor Apache como repositório, mas limitou a capacidade que a arquitetura Cliente/Servidor oferece para cumprir as necessidades do Portal.
- O desenvolvimento do protótipo mostrou que essa arquitetura é mais adequada para aplicações que não precisam guardar informações entre requisições.

- **5.8.2 Cliente Gordo / Mestre – Escravo**

Depois da arquitetura anterior ter se mostrado incapaz de oferecer os requisitos necessários, essa nova foi construída a partir da anterior, modificando o Portal, quebrando-o em duas partes, para fazer com que apenas uma pequena parte fosse acoplada ao servidor e mudanças na parte do Cliente.

Como cliente, determinou-se que este passaria a ser mais capaz na interação com o usuário e nas habilidades de manipulação gráfica, que iriam remover totalmente a necessidade do servidor em utilizar qualquer biblioteca gráfica, mostrando-se como uma fonte de problemas na primeira arquitetura.

Nesta nova arquitetura, o servidor é composto por duas partes, um Mestre que irá receber as requisições como o Portal na arquitetura anterior, todavia não irá mais processá-la, apenas enviando-a a segunda parte, o escravo, fazê-lo. Desta forma, pode-se forçar que apenas um processo do servidor receba todas as chamadas de requisições, de qualquer cliente. O Mestre ainda tinha o papel de criar novos Escravos quando fosse necessário ou finalizar os Escravos.

Com essa nova abordagem ainda é possível permitir que os processos do servidor

fossem recriados, visto que todos os estados das requisições, de cada cliente, estariam replicados nos escravos, de modo que, quando necessário, o Mestre apresentaria a capacidade de reconstruir os estados. Para tal reconstrução os escravos e o Mestre iriam estabelecer uma rede de comunicação.

Essa rede teria como idéia inspiradora as salas de bate papo populares na Internet, que permitem aos usuários participarem de grupos de conversas, onde é possível enviar ou receber uma mensagem de um ou para todos os participantes do grupo de forma simultânea. Cada grupo é independente do outro, porém cada usuário pode participar de vários grupos ao mesmo tempo. No projeto essa comunicação se daria de seguinte forma:

- O Mestre sempre vai escutar as mensagens enviadas pelos clientes por uma porta de comunicação fixa.
- As mensagens dos escravos podem conter:
    - Porta pela qual cada Escravo espera receber mensagem do Mestre. Isso é necessário para que os vários Escravos utilizem a mesma interface de rede sem conflitos.
    - Informações de estados como o início de uma operação ou a sinalização do término de uma tarefa.
    - Disponibilidade para trabalho.
- As mensagens de disponibilidade para trabalho que os escravos enviam ao Mestre são empilhadas pelo Mestre.
- Quando o usuário manda uma tarefa, usando o cliente que por sua vez manda para o servidor, a parte Mestre do servidor desempilha uma mensagem disponibilidade para trabalho e envia a tarefa do usuário ao Escravo que mandou a mensagem.
- Quando o Mestre é reiniciado pelo servidor Web esse espera que os Escravos tentem reconectar e, conforme cada um se reconecta, o Mestre pede ao Escravo que mande as informações sobre o que cada um esta fazendo.
- Quando o Escravo perde a conexão com o Mestre, este entra em um loop, com uma nova thread, tentando restabelecer a conexão.

Esta rede permitia que os escravos fossem hospedados em máquinas diferentes, aumentando assim a escalabilidade, tendo como uma futura possibilidade os escravos serem distribuídos utilizando o Grid. Isso expandia as fronteiras pois, dessa forma, o Grid passou a ser um recurso que o Portal poderia utilizar para aumentar a capacidade de processamento de

pedidos dos usuários.

Outro fato marcante durante a vida dessa arquitetura foi a evidente falta de documentação, que fosse completa o suficiente, para permitir o desenvolvimento da comunicação eficiente entre o Mestre e seus Escravos, que resultou em grande frustração e desperdício de tempo. Evidentemente, detectou-se tal carência durante tentativa de utilização dos componentes de comunicação do Kylix que, à primeira vista, possui todas as capacidades necessárias para o uso eficiente, contudo a documentação interna e os nomes dos procedimentos são por demais simples a ponto de não esclarecerem a forma pela qual cada um deveria ser utilizado. Tal implica a não implementação da rede de comunicação.

O resultado dessa arquitetura, que marcou o fim do uso do Kylix, consiste na construção de um protótipo do servidor e uma demostração de um cliente, que foram capazes de enviar comandos para o compudador e, este, era executado por um shell real, através de chamadas de bibliotecas do sistema operacional, que hospedava a parte servidor do portal. Além disso, apresentava a capacidade de realizar listagem de arquivos e suas transferências, usando apenas Web Services. Já, o Cliente, consistia num formulário com caixas de textos e botões que, por meio de interações com o usuário, enviava os pedidos, exibia as repostas e permetia a transferência de arquivos entre o cliente e o servidor em ambas direções. Note que esse cliente era multi-plataforma, rodava tanto em Windows (arquitetura x86 32bits) quanto em GNU/Linux (arquitetura igual).

Por causa das limitações encontradas note que o Portal ainda não possuía o lado Escravo e, tendo como efeito, que todo o trabalho ainda era feito pelo Mestre, todavia, existia um objeto Escravo que encapsulava toda as suas funcionalidades, o este objeto Escravo, tornarseia um processo indepedênte se a arquitetura não fosse ambandonada. A escolha de não implementar o escravo como processo separado logo de inicio foi pela tantativa de simplificar a arquitetura enquanto os problemas relacionados com os componentes de comunicação não fosse resolvidos, e mantendo o Escravo como objeto interno do Mestre tal comunicação seira feita por chamada a métodos. Note que a comunicação mesmo por chamada de métodos seguia a idéia que os Escravos estariam em processos separados, para tal todas as informações eram passadas via parametros de métodos.

Como conclusões dessa arquitetura, tem-se:

- Quebrar o Portal em duas partes, Mestre e Ecravo deu ao Portal alta escalabilidade,

resistência a falhas e abriu novas oportunidade de implementação dos Escravos, como a possibilidade de utilizar um Grid para executar várias cópias dos Escravos.

- A implementacão da rede de comunicação do Mestre e seus Escravos se mostrou, juntamente com a restrição de tempo, por demais complexa para ser adotada no projeto.

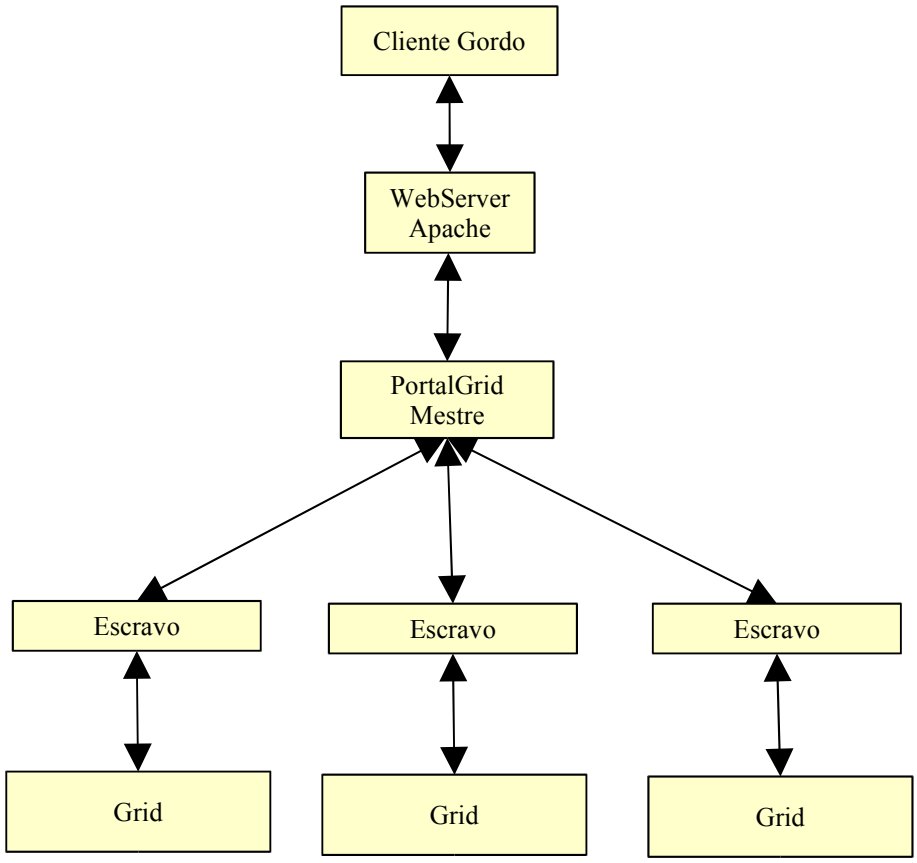

Figura 5

Na figura 5 temos:

- Clientes

  Uma interface sob forma de um formulário que irá interagir com o usuário

- Servidor

  Um servidor Web que irá receber requisições por HTTP e, por causa das configurações, os pedidos para o Portal (identificado através da url) são enviados para a parte abaixo.

- Portal / Mestre

  Um componente ligado ao Servidor acima como um "Shared Object" no Linux

ou "DLL" no Windows, que faz o tratamento da qualidade do pedido do cliente e, conforme o resultado, procura um Escravo para realizar a tarefa pedido pelo cliente.

- Escravos

Processos que realizam os pedidos fazendo chamadas ao Sistema Operacional por meio de interações com um shell.

- Grid

Um conjunto de aplicativos e de equipamentos que recebem ordens dos Escravos para tomar alguma ação em cima dos aplicativos dos usuários, como ações dentre outras pode-se ter a inicialização, a busca de informações de estado ou a finalização.

CAPÍTULO 6 - AMADURECIMENTO DO PROJETO

Durante a segunda metade do desenvolvimento do projeto final foram tomadas alterações revolucionárias em relação às técnicas de implementação, o que resultou na concretização integral das funcionalidades planejadas para o projeto, destinando-lhe um rumo mais adequado.

De fato, listam-se, a seguir, algumas mudanças que merecem destaque:

6.1 Escolha da Ferramenta/Linguagem de desenvolvimento:

Neste ponto, encontra-se a maior mudança realizada, que consiste na escolha da linguagem Java em detrimento do Kylix, como ferramenta de implementação do projeto.

Diversos motivos foram responsáveis por tal atitude, a mencionar:

- Estagnação da ferramenta Kylix, já que, aparentemente, foi abandonada pela sua produtora, em contraste com o crescente desenvolvimento da linguagem Java, incluído o uso em Grids, e suas ferramentas.

- Falta de material didático que auxiliasse no uso dos componentes do Kylix, destoante do farto material em Java, que em muitas vezes se mostravam independentes das ferramentas de desenvolvimento.

Questões legais, visto que o produto resultante do Projeto só poderia fornecer o código fonte para os clientes se estes possuíssem ou adquirissem o Kylix, resultando em

elevação de consumo dos recursos financeiros, em decorrência de restrições das

licenças de uso e redistribuição.

•Superação de barreiras que impediam a escolha de Java durante o inicio do projeto, como: encontrar um ambiente maduro, amigável e de uso gratuito; ferramentas específicas para Web Service e a habilidade do desenvolvedor, em questão, com a linguagem.

6.2 Arquitetura construída durante o amadurecimento.

- **6.2.1 Cliente Gordo / Web Container**

Esta arquitetura foi o resultado de pesquisas, pelas quais sistemas em Java poderiam ser construídos, usando todas as capacidades e características que a linguagem oferecia em relação a anterior. Como esperado, a pesquisa apresentou bons resultados que se traduziram no uso de algumas ferramentas, disponíveis de forma gratuita, todavia bastante madura e avançada.

Esta arquitetura é muito semelhante à primeira, diferenciando-se no fato do cliente ter adquirido mais responsabilidades e o servidor ter passado por uma reestruturação interna.

Essa nova abordagem consiste no desenvolvimento da capacidade, implicando em um custo aceitável de integrar o servidor Web com o portal, de unificar as funcionalidades do servidor em seu próprio corpo, portanto, removendo a necessidade de instalação e configuração do servidor como um programa a parte.

Nota-se que, apesar de existir essa integração, o grau de acoplamento entre o Portal e o

servidor Web interno não é grande, devido às características do Java, existe a possibilidade de se remover o servidor interno e voltar a usar de forma externa, sem efeitos colaterais.

Durante o processo de desenvolvimento, verificou-se, inúmeras vezes, que a decisão de mudar a linguagem de implementação para Java foi acertada, visto que em vários momentos, o projeto ganhou como opções um rico meio de soluções, cada dificuldade detinha várias alternativas e, em muitas ocasiões, a documentação dessas era facilmente encontrada na Internet.

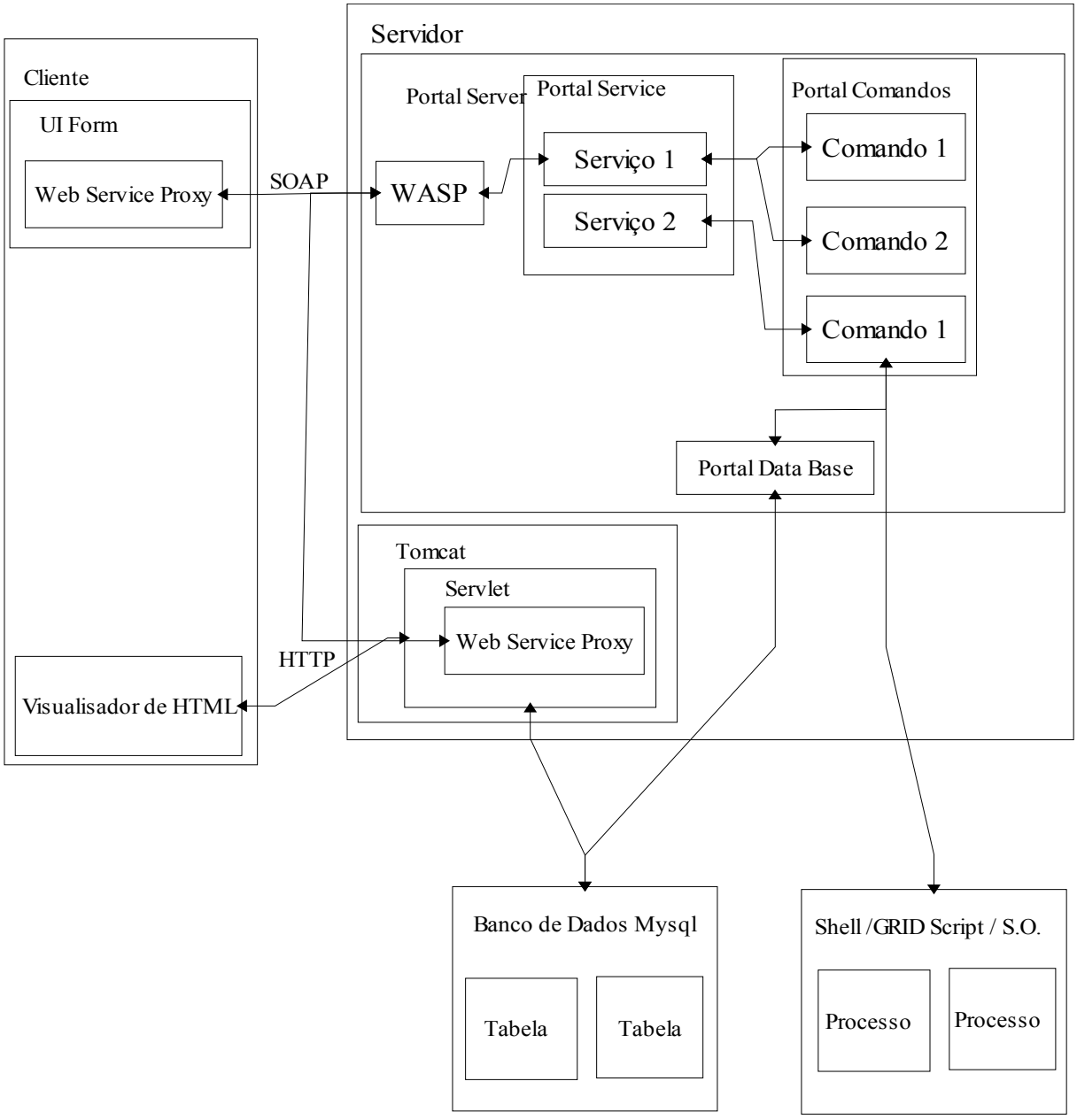

FIGURA 6

Destacam-se partes da arquitetura representada pela figura acima figura 6:

- Cliente – Agora pode se utilizar duas interfaces gráficas para acessar o Grid
    - O primeiro é utilizar Web Services para realizar as ações desejadas no Grid, para tal o Cliente irá utilizar um 'proxy' que através de métodos tem o papel de um intermediário entre o Cliente e o Portal. Todas as funcionalidades gráficas e de interações com o usuário ficam na máquina Cliente.
    - A segunda opção é utilizar uma interface Web que irá oferecer uma forma de acesso simplificado. A diferença com a interface anterior é o fato do 'proxy' ficar dentro do servidor Web que por sua vez está na máquina servidora. Como foi visto na primeira arquitetura, a capacidade gráfica e de interação da interface Web é mais limitada.
- Servidor:
    - WASP – Componente que promove a comunicação em Web Service com os clientes
    - Tomcat – Servidor Web que serve como intermediário para os clientes magros (visualizadores de paginas html) e o WASP
    - Portal Service – Serviços que os usuários podem acessar como Web Service, normalmente existe um mapeamento de um para um entre Serviço e Comando. Logo se pode ver um Serviço como um Comando acessível via Web Services.
    - Portal Comandos – Objetos que realizam ações no Grid, como executar um script ou consultar o banco de dados para obter informações.

Portal Data Base – Objetos responsáveis pela comunicação do Portal com algum banco de dados.

CAPÍTULO 7 – Implementação DA última arquitetura

Antes de iniciarmos a explanação, se faz necessário o esclarecimento mais profundo sobre os componentes da arquitetura.

7.1 SOAP

Segue uma breve seqüência de exemplos simples com a finalidade de mostrar o papel que o protocolo SOAP tem dentro do Web Service.

Como exemplo vamos definir o mundialmente famoso “Alo mundo” que vai ser exposto como Web Service.

```
public interface Hello
    public String sayHelloTo(String name)
```

A seguir segue um exemplo de como esse serviço poderia ser invocado por um arquivo XML.

```
<?xml version="1.0"?>
<Hello>
 <sayHelloTo>
 <name>John</name>
</sayHelloTo>
</Hello>
```

A seguir, um exemplo de uma resposta para a requisição anterior:

```
<?xml version="1.0"?>
<Hello>
   <sayHelloTo>
         <name> Hello John</name>
  </sayHelloTo>
</Hello>
```

Essa é uma forma muito simples de descrever pedidos e respostas, todavia não é suficientemente completo para resolver problemas do cotidiano. Esses problemas podem corresponder à definição exata do tipo de cada parâmetro, da forma de codificação do pacote e da chamada do serviço.

Pedido

```
<SOAP-ENV:Envelope
                xmlns:SOAP-ENV="
                http://schemas.xmlsoap.org/soap/envelope/"
                xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Header>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ns1:sayHelloTo
            xmlns:ns1="Hello"
           SOAP-ENV:encodingStyle="
http://schemas.xmlsoap.org/soap/encoding/">
         <name xsi:type="xsd:string">John</name>
      </ns1:sayHelloTo>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Nota-se que:

- *"xmlns:xsd="http://www.w3.org/1999/XMLSchema"* declara de qual XML-schema os tipos de dados foram criados.
- *"<ns1:sayHelloTo>"* determina o método que o usuário deseja utilizar.
- *"xmlns:ns1="Hello""* indica o parâmetro do método anterior
- "*<name xsi:type="xsd:string">John</name>*" atribui um valor a esse parâmetro.

Resposta

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"> <SOAP-ENV:Body>
<ns1:sayHelloToResponse xmlns:ns1="Hello" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <return
xsi:type="xsd:string">Hello John, How are you doing?</return>
</ns1:sayHelloToResponse> </SOAP-ENV:Body> </SOAP-ENV:Envelope>
```

Erro

Observa-se que Web Services possui uma forma bem definida para o tratamento de erro, desse modo, exibe-se, a seguir, o exemplo de um possível erro:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
     <faultcode>SOAP-ENV:Server</faultcode>
     <faultstring>Server Error</faultstring>
      <detail>
          <e:myfaultdetails xmlns:e="Hello">
              <message> Desculpe, mas hoje eu não posso dizer oi. </message>
              <errorcode>    1001      </errorcode>
          </e:myfaultdetails>
      </detail>
   </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

7.2 WSDL

```
<message name="myMethodRequest">
   <part name="x" type="xsd:String"/>
</message>
<message name="empty"/>

<portType name="PT">
   <operation name="sayHelloTo">
      <input message="myMethodRequest"/>
      <output message="empty"/>
   </operation>
</portType>
```

Neste exemplo de WSDL foi demonstrado apenas uma parte do que seria o documento inteiro para evidenciar como o método *sayHelloTo* pode ser representado.

Acrescentam-se algumas informações para uma descrição mais completa de um serviço por WSDL, como a definição do Namespace, um ponto de referência para a nomenclatura dos tipos que os parâmetros podem assumir. Exemplifica-se, assim, o momento quando se deseja saber o significado da palavra "Fila" , que em Português no Brasil difere do em Portugal, especificando o dicionário do país a utilizar.

7.3 Java Container

Para o desenvolvimento de uma aplicação que disponibiliza as funcionalidades para outros programas clientes, sobre forma de Web Services, é importante existência de um componente, implementado de forma interna ou não a esta aplicação, que será o responsável por receber o pacote SOAP enviado pelo cliente, por meio de algum protocolo de comunicação, como HTTP, bem como desempacotar as informações, realizar uma pesquisa sobre o conteúdo do pedido, a fim de localizar e invocar um responsável para atendê-lo. De forma inversa ocorre o mesmo fluxo, do servidor em direção ao cliente que mandou o pedido.

Neste projeto, foi feita uma seleção por tal componente, visto que já tinha sido implementado, bem como ser de utilização gratuita, pois o desenvolvimento deste componente seria muito complexo e demorado. Dentre as opções, duas possuem maiores destaques, o projeto APACHE AXIS e o SYSTINET WASP.

Cumpre exibir o Apache Axis, projeto de código aberto, que visa a fornecer ferramentas básicas, possibilitando o desenvolvimento de aplicações em Web Services. Como características favoráveis, destacam-se a gratuidade e simplicidade, cujo foco consiste no recebimento de uma requisição do servidor de transporte, o responsável pelo processamento do protocolo, como HTTP, para a transmissão, entre o cliente e o servidor pela Internet, de um pacote SOAP, enviado pelo cliente e, posteriormente, tratar esse pacote, criando um contexto de execução, que representa um conjunto de dados a ser usado por outros objetos, sendo seguida pela invocação de um manipulador, incumbido de responder especificamente o pedido contido no pacote SOAP recebido. Define-se, como manipulador, aquele que permite ao desenvolvedor exportar qualquer funcionalidade que a aplicação desenvolver.

Ressalta-se, ainda, a dependência do Axis de um outro servidor de transporte evidencia a sua capacidade limitada, que corresponde apenas à manipulação de pacotes SOAP. Assim, força-se o desenvolvimento ou a incorporação de um outro componente para implementar o protocolo de transferência. Como exemplos de servidores que poderiam ser utilizados, temos o Apache Tomcat e o Jboss.

Outra alternativa pesquisada foi o WASP SYSTINET, um sistema modular, com inúmeras facilidades de uso, como ferramentas de administração e desenvolvimento, uma boa performance, dotada de características avançadas de interoperabilidade e segurança. Como demonstração de sua qualidade, este servidor foi utilizado como implementação de referência pelo W3C para o padrão 1.2 do protocolo SOAP. Este ainda detém algumas características adicionais, abrangendo a capacidade de funcionar como servidor independente, sendo prescindível a existência de outros servidores de transporte, com fins de receber as requisições, ou como componente adicional de servidores de aplicações, cabendo exemplificar o “IBM Websphere” e “Sun ONE Application Server”.

Nesse contexto, o componente escolhido para dar o suporte ao projeto foi o WASP, visto que disponibiliza ferramentas adicionais de desenvolvimento que se integram com ambiente de programação escolhido, o Borland Jbuilder Personal 9, cabe destacar que essas ferramentas estão disponíveis para outros ambientes como o Eclipse e NetBeans. Menciona-se, ainda, que este possui um tipo de licenciamento gratuito, que abarcar o uso em servidores com um processador, não podendo ser distribuído a terceiros.

O Portal encontra-se dividido em quatro pacotes, com a finalidade de separar a

funcionalidade e diminuir o acoplamento entre os objetos, destacando a seguir:

- Server

  Este pacote possui a classe Server, que é responsável por iniciar um servidor interno de Web Services.

- Client

  Contém o código de simulação de um cliente, com o propósito de testar o servidor de Web Services e os serviços.

- Services

  Contém a Classe responsável por exportar os comandos como Web Services.

- Comandos

  Possui o conjunto de classes que fazem a conversão de serviços para o conjunto de ações.

A seguir, se detalha sobre a maneira que cada componente realiza as suas atribuições. Cabe lembrar que a convenção da nomenclatura segue o padrão de nomenclatura do Java, que corresponde ao nome da classe acrescida do prefixo com o nome do pacote a qual essa classe pertence, como no exemplo “MeuPacote.AlgumaClasse” .

- **Server.PortalServer**

Classe responsável por configurar e inicializar o servidor WASP. A configuração compõe-se de, determinação da porta de comunicação a qual o servidor irá atender, os nomes dos serviços e as classes ou objetos que serão responsáveis pelo atendimento de cada serviço.

```
String serverURL = new String("http://localhost:6060");
Wasp.startServer(“/portal”);
Registry.publish(servicePath, new Portal());
```

- **Service.Portal**

Esta classe é a implementação principal dos serviços do projeto, com o papel de oferecer um mapeamento dos pedidos feitos pelos usuários, através de chamada a serviços, para classes que possuem as regras de negócio. Convém mencionar que essas regras, neste caso, representam as funcionalidades necessárias ao uso do Grid, como transferência de arquivos e execução de programas no Grid.

Esta classe foi implementada com a intenção de separar as funcionalidades seguindo a arquitetura de 3 camadas, onde a classe Portal seria a segunda, que serve como intermediador entre o cliente e o comandos do grid, promovendo assim um alto grau de isolamento entre eles. Esta característica fica evidente na capacidade, por exemplo, de utilizarmos a mesma classe que implementa as regras de negócio em uma aplicação com páginas em JSP.

```
public void directrun( String[] args )
    Direct cmdDirect;
    cmdDirect = new Direct();
    cmdDirect.run( args );
```

Direct como exemplo é uma classe que implementa a execução de um comando em um shell no servidor

- **Service.PortalException**

Se durante o processamento de qualquer serviço uma situação de erro ocorrer, este será convertido em uma exceção, sendo enviado ao usuário para informar o erro, de modo a oferecer a oportunidade de identificar a razão, pelo o conteúdo da exceção. Percebe-se que as exceções do Java se diferenciam do padrão SOAP, que entende a exceção como uma mensagem de falha. Para tal o servidor WASP realiza o mapeamento adequado do PortalException para a mensagem de erro SOAP. Cabe ao cliente fazer a conversão de volta pela melhor maneira que a linguagem de implementação permitir.

```
public class PortalException extends Exception
    public PortalException() { }
    public PortalException(String message) { super(message); }
    public PortalException(Exception ex) { super(ex); }
```

- **Comando.Direct**

Este é o exemplo de implementação de um serviço que, neste caso, permite ao usuário executar comandos específicos em um "Shell" no servidor.

Os comandos são armazenados em um banco de dados relacional que, neste projeto, corresponde ao Mysql, um servidor de código aberto, de uso gratuito, com classes de acesso em java, interface gráfica para administração e já disponível na distribuição GNU/Linux Mandrake.

Elenca-se, a seguir, o conteúdo da tabela de comandos:

- hasparam –Determinação se o comando vai aceitar o parâmetro enviado pelo usuário.
- Redirect - Condição para redirecionar toda a saída "STDOUT" para um arquivo.
- Mask - prefixo no nome do arquivo de redirecionamento para este comando
- Nome – Identificação que o usuário vai usar, para invocar o comando, note que este é um nome "fantasia", como por exemplo "desligar_comp".
- Call – Identificação para a qual o Portal irá utilizar na execução do comando no "Shell", evidencia-se que é o caminho completo para o executável no sistema do servidor, como exemplo "/sbin/poweroff".

Vale destacar os motivos que levaram ao uso de um banco de dados relacional para armazenar informações de funcionamento do Portal. Durante o desenvolvimento notou-se a necessidade de construir um modo de armazenar informações dinâmicas do Portal, como dados dos usuários, histórico de tarefas executadas, uma estrutura de controle que permitisse a utilização do sistema apenas pelos usuários autorizados, de preferência a nível granular, a

ponto a permitir a definição de permissão a cada comando para cada usuário e que precisava ser modificado durante a fase de produção do Portal. Algumas idéias como utilização de uma hierarquia de diretórios e arquivos foi tentada, porém, logo abandonada, pela dificuldade em implementar todos os detalhes. A escolha de utilizar um banco mostrou-se promissora e bastante adequada, destacando os motivos a seguir:

- Habilidade em mapear as informações do Portal facilmente em tabelas
- Utilização de conhecimentos já adquiridos nos trabalhos do curso.
- Fonte de dados passível de ser utilizada por outros sistemas
- Rica disponibilidade de material didático.
- A existência de ferramentas capazes de manipular os dados do banco sem a necessidade de se usar o Portal, o que facilitaria a administração.

- **Comando.Icomando**

Interface à qual todos os comandos irão implementar. Essa classe objetiva definir um modelo de uso uniforme, por meio de métodos e parâmetros iguais. Repara-se que isso facilita o desenvolvimento da classe Services.Portal, posto que o mapeamento de todos os serviços em objetos do pacote Comando será uniforme, com a vantagem de obrigar o melhor desacoplamento entre as classes.

CAPÍTULO 8 - Conclusões

Inúmeros eventos marcaram este projeto final que colocaram à prova todos as experiências e conhecimentos adquiridos durante a vida acadêmica. Pela primeira vez, essas habilidades foram utilizadas plenamente e, visto que o projeto teve uma grande liberdade de pensamento e expressão, em que a experimentação para a aprendizagem era limitada apenas pela criatividade e ambição de seu autor.

Dentre as tecnologias pesquisadas podemos destacar algumas conclusões:

- Grid

Grid representa uma idéia, cujo maior ganho, pode ser o desenvolvimento de um novo e importante paradigma, questionando a forma com a qual trabalhamos e desenvolvemos sistemas em conjunto com os recursos computacionais.

Não existe um único caminho que possa ser trilhado para a implementação, pode e se deve escolher, quando for melhor, apenas um sub conjunto de características de computação em Grid, que podem trazer o maior ganho para as atividades produtivas.

- Web Services

Com ou sem Grid, esta tecnologia terá um papel muito importante para o futuro da computação, abrindo o caminho para sistemas integrados, buscando o acesso de capacidades distribuídas e possibilitando que recursos dispersos se unificaquem de forma a melhorar os processos produtivos, assim como na sua criação.

A futura sociedade, cada vez mais globalizada, tende a obrigar a aproximação de pessoas e habilidades e, nesse meio, Web Service , ou a sua evolução, irá prosperar.

- Java

Esta iniciou sua vida como linguagem de objetivo primeiro de ser executado em eletrodomésticos, que ao longo do tempo, evoluiu ao ponto de se tornar uma promissora fonte de soluções, com a sua forma coloborativa de evolução. É capaz de atrair inumeras iniciativas de usuários que tentam expandir as capacidades e redefinir os limites do Java.

Deve-se destacar que, apesar de existir linguagens alternativas com performace superior, Java e a sua Máquina Virtual que dá vida ao código, estão evoluindo de tal forma a diminuir a diferença com as outras linguagens. Na geração 1.4 a grande diferença de Java com as alternativas de "maior performance" , a linguagem "C" , esta nas bibliotecas de manipulação matemática, contudo vale destacar que atravez de JNI (Java Native Interface) é possível usar bibliotecas em "C" para a realização dessas manipulações. Informações e comparativos de performance podem ser encontrados dos "sites" "www.osnews.com" e "www.theserverside.com".

- Outras formas de computação distribuída

Apesar da escolha do Web Service em favor de tecnologias tradicionais como CORBA e RMI, ressalta-se que essas ainda são muito úteis, principalmente se o objetivo do sistema a ser construído consiste em promover a comunicação entre recursos em uma mesma rede ou, em que se necessite, de um alto grau de desempenho, quando os recursos humanos e de equipamentos normalmente já são integrados. Tecnologias antigas possuem como um grande atrativo a sua maturidade, levando a um alto grau de eficiência no uso dessas. É facil encontrar materiais didáticos, cuja qualidade leva ao desenvolvimento por caminhos muitas vezes trilhados, usualmente esclarecendo todas as qualidades, limitações e seus contornos.

De modo global, o Projeto obteve sucesso diferenciado em cada um de seus diveros objetivos, apresentando um balanço final bastante positivo, no sentido de se proporcionar um ambiente de pesquisa sobre novas tecnologias, contudo não alcançou êxito total na intenção de oferecer uma ferramenta ao projeto EasyGrid. Percebe-se que, dentre os motivos, encontra-se a constante modificação dos "scripts" necessários para execução e controle, além dos prazos de disponibilização em uma época não compativel com o andamento do projeto final. Convém ressaltar o sucesso na elaboração desta última arquitetura, apesar de implementado apenas as funcionalidades principais, oferece características necessárias aos requisitos do Easygrid, portanto, em futuros trabalhos, será possivel o sucesso integral deste objetivo

ferramental.

Entre as futuras modificações possíveis da última arquitetura, tem-se o ressurgimento da idéia de quebrar o Portal em Metre e Escravos, que foi apresentada pela segunda arquitetura já apresentada. Com a utilização de JMS (Java Message System) e de Jini, seria possível o desenvolvimeno da rede de comunicação, que foi inviabilizado quando ainda se utilizava o Kylix, porém essas tecnologias são bem documentadas, não sendo difícil achar tutoriais e exemplos na internet, como no site da Sun e do Javaworld, de modo que as barreiras de uso não se repitiriam.

Como conhecimento adquirido pelo projeto final, permanece a idéia de um futuro promissor para tecnologias baseadas nos conceitos de Grid e Web Services, restando evidente a adequação do uso Web Services em Grid, com a finalidade de criar modos de acesso às funcões e recursos do último.

Vale resaltar, ainda, que o projeto final não criou um Portal completo, pois dentre os motivos, constam as grandes e rápidas mudanças nas duas tecnologias que cercam o Portal, como definições de novos padrões em Web Service e em Grid. Como exemplo, se destaca a criação do projeto OGSA (Open Grid Services Architecture) definido pelo OGSI (Open Grid Services Infrastructure) que é produzido pelo grupo "Global Grid Forum", tendo como principal implementação o "Globus Toolkit 3" que, resumidamente, é a integração entre Web Services e Grid para a criação de Grid Services.

Logo, Grid Service pode ser considerado um "fork" de Web Services, devido à constante batalha entre fabricantes, como Microsoft , IBM e SUN, impeliu ao Web Services possuir muitas extensões de funcionalidades, que também são necessárias ao Grid, todavia boa parte não apresenta código aberto, ou melhor, são licenciados por cada fabricante mediante um custo financeiro. Resulta, assim, a necessidade da criação de padrões de extensões indepedentes e abertos, correspondendo ao objetivo que o Grid Services visa a alcançar.

8.1 Considerações finais

Por diversas vezes tecnologias baseadas em Java prósperas foram descartadas em favor de outras mais aplicáveis aos requisitos deste projeto, contudo, algumas das não escolhidas, oferecem um grande leque oportunidades em pesquisas dentro da área de computação distribuída.

Uma interessante proposta é da confecção de uma Tese de Mestrado, defendendo o uso de técnicas como JINI, JXTA e diversas outras do rico celeiro de idéias Java.

Menciona-se que JINI é um conjunto de ferramentas e arquiteturas que promove a computação distribuídas de objetos Java, com conceitos avançados de operações transacionais, utilização transparente de recursos remotos e meios para divulgação e pesquisas de serviços.

Já, JXTA, representa um conjunto de protocolos para a construção de uma rede Peer-To-Peer puramente baseado em Java.

Em suma, o presente trabalho apresenta possibilidade significantes para a elaboração de pesquisas posteriores, posto que explora um ponto local, dentre a extensa rede de facilidades e de benefícios tecnológicos que o Grid se mostra propenso a oferecer. Dessa forma, esta potencialidade desta inovação arquitetural se traduz numa fonte de soluções para as diversas questões expoentes no âmbito tecnológico e mercadológico, conforme explanado no corpo deste texto.

RECURSOS CITADOS

IBM-International Business Machines Corporation

http://www.ibm.com/developerworks/grid/library/gr-web/index.html

http://www.ibm.com/developerworks/library/gr-heritage/

http://www.ibm.com/developerworks/webservices/library/ws-arc3/

http://www.ibm.com/developerworks/webservices/library/gr-visual/

Reinventing the Wheel? CORBA vs. Web Services , Aniruddha Gokhale , Bharat Kumar e Arnaud Sahuguet

http://www2002.org/CDROM/alternate/395/

Revista Scientific American

http://www.scientificamerican.com/article.cfm?chanID=sa006&colID=1&articleID=000E9724-331A-1C71-84A9809EC588EF21

Revista eletrônica Java World

http://www.javaworld.com/javaworld/jw-03-2001/jw-0330-soap.html

RECURSOS CONSULTADOS

Projeto Teragrid

[teragrid]http://www.teragrid.org/

Instituto de pesquisa Seti

[seti] http://www.seti.org/

Systinet, empresa de soluções em infraestrutura de Web Services

[systinet] http://www.systinet.com/

SUN MicroSystens, Empresa responsável pelo Java

[sun] http://www.sun.com/

[sun-rmi] http://java.sun.com/products/jdk/rmi/

Borland, soluções para desenvolvimento de software

http://www.borland.com/

Mandrake Linux , distribuição Linux

http://www.mandrakelinux.com/

Padrão desenvolvido pela OMG

[omg] http://www.corba.org/

Globus Alliance, define e implementa padrões de Grid.

http://www.globus.org/

Revista eletrônica com um rico conteúdo de material sobre Java

http://www.javaworld.com/

Fundação de apoio a projetos Open Source

http://www.apache.org/

Site nacional com notícias e artigos para Linux

[br-linux]http://br-linux.org/

Revista sobre Informática.

[compuworld]
http://computerworld.uol.com.br/AdPortalV3/adCmsDocumentoShow.aspx?Documento=27215

Objetivity - Fornecedor de Banco de Dados

[objetivity]
http://www.objectivity.com/NewsEvents/Releases/archived_releases/1997/CERN.html