Universidade Federal da Bahia
Escola Politécnica / Instituto de Matemática
Programa de Pós-graduação em Mecatrônica
Mestrado em Mecatrônica

JUREMA ROCHA BARRETTO

USO DE GRADE COMPUTACIONAL PARA IDENTIFICAÇÃO DE INDIVÍDUOS ATRAVÉS DE IMPRESSÃO DIGITAL EM LARGA ESCALA

DISSERTAÇÃO DE MESTRADO

Salvador
2010

JUREMA ROCHA BARRETTO

USO DE GRADE COMPUTACIONAL PARA IDENTIFICAÇÃO DE INDIVÍDUOS ATRAVÉS DE IMPRESSÃO DIGITAL EM LARGA ESCALA

Dissertação apresentada ao Mestrado em Mecatrônica da Escola Politécnica e do Instituto de Matemática, Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Mecatrônica.

Orientadora: Profa. Dra. Fabíola Greve
Co-orientador: Prof. Dr. Genaro Costa

Salvador
2010

Dedico esse trabalho aos meus pais, cuja educação, amor e exemplo de dignidade fez eu me tornar quem sou.

Determinação, coragem e autoconfiança são fatores decisivos para o sucesso. Não importa quais sejam os obstáculos e as dificuldades. Se estamos possuídos de uma inabalável determinação, conseguiremos superá-los. Independentemente das circunstâncias, devemos ser sempre humildes, recatados e despidos de orgulho.

—DALAI LAMA

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus pelo dom da vida e pelas oportunidades que coloca no meu caminho.

Em seguida, agradeço aos meus pais por todo carinho, amor, dedicação, pela educação dada e pelo exemplo de pessoas dignas e do bem. Em especial à minha mãe cuja cumplicidade é inexplicável.

A Pedro, meu namorado, pelo amor, carinho, companheirismo, força e principalmente paciência e compreensão nos momentos de ausência.

A Fabíola, minha orientadora, pelos ensinamentos que me fez crescer bastante.

Ao amigo e co-orientador, Genaro Costa, que pra mim é um exemplo de um grande profissional. Obrigada, pelas dicas, pelos incentivos, pela paciência e orientação final.

Aos meninos do Gaudi, cuja participação na implementação foi fundamental para o projeto. Amadeu, Italo, Wilson e Filipe, muito obrigada.

A todos os amigos que sempre torceram por mim, entenderam a minha ausência e que, junto comigo, aguardam a conquista desse sonho.

Por fim, agradeço à Fapesb pelo financiamento do projeto e pela bolsa que recebi.

RESUMO

Sistemas automatizados de identificação através da impressão digital (AFIS) têm sido bastante utilizados como alternativas aos recursos tradicionais, tais como senhas e identificação em papel. Uma das suas maiores aplicações está na segurança pública, na identificação civil e criminal de indivíduos. Processos automatizados empregados no reconhecimento de duas impressões digitais (um para um) são simples e eficientes. O grande entrave consiste em efetuar verificações de um para muitos. A depender da ordem de grandeza da base de dados, esse procedimento pode levar muito tempo, devido a grande necessidade de processamento requerida. Para minimizar o tempo de busca, costuma-se adotar gerenciadores de banco de dados especializados e computação de alto desempenho. O que dificulta, por exemplo, a implantação de um sistema de reconhecimento criminal no âmbito estatal, mais ainda no âmbito nacional. Devido a particularidades de cada Estado, como por exemplo, disponibilidade de recursos financeiros, além do aumento progressivo da base de dados desse tipo de informação. Alternativas como o uso de programação paralela e distribuída, e *cluster* de computadores podem ser adotadas para obter menor tempo de busca. Porém, tornam-se ineficientes quando se trata de grandes bases de dados que têm crescimento constante e que possuem restrições de segurança para assegurar a privacidade dos dados compartilhados. As grades computacionais, dentre outros recursos, viabilizam a execução de aplicações sobre processadores dispersos geográfica e administrativamente proporcionando maior capacidade de processamento (o que implica na redução do tempo de resposta) e melhor custo-benefício que as plataformas centralizadas tradicionais. Ao mesmo tempo, ao utilizá-las, é possível compartilhar grande quantidade de dados distribuídos e dispor de meios para implementar recursos de segurança e autenticação afim de manter o sigilo e independência dos dados. Essa dissertação apresenta uma arquitetura para integração de dados de impressão digital dispersos geograficamente através do uso de grade computacional para atender as demandas de alto processamento computacional. Para atingir esse objetivo, provê o compartilhamento de informações através de um ambiente seguro, consistente, independente e de baixo custo, fundamentando-se em soluções de código aberto e hardware heterogêneo. Para validar a arquitetura, foi desenvolvida uma aplicação para identificação criminal através do uso de impressões digitais que pode ser executada em plataformas como, computadores pessoais (*desktop* ou *notebook*) e dispositivos móveis - mais especificamente em PDA's. Os resultados obtidos, a partir de testes realizados em alguns cenários representativos, comprovam que o uso de grade computacional melhora o tempo de resposta para esse tipo de aplicação e habilita o uso de sistemas AFIS em larga escala.

Palavras-chave: Impressão Digital, Grades Computacionais, Busca Distribuída

ABSTRACT

Automated fingerprint identification systems (AFIS) have been widely used as an alternative to traditional methods, such as passwords and identification by paper. One of their major applications is in public security, in the civil and criminal identification of citizens. Automated processes applied in the recognition of two fingerprints (one to one) are simple and efficient. The major obstacle is to compare one fingerprint with many others. Depending on the size of the database, this process may take a long time, because of the high performance computing required. To minimize the search time, specialized database managers and supercomputers are usually adopted. Although the costs of these resources have cheapened, their performance are affected when intensive computing is required. Alternatives such as parallel and distributed programming, as well as cluster computing could be applied to reduce search time. But, they become inefficient to large databases that are constantly growing and have security restrictions to ensure privacy to the shared data. Grid computing, in turn, enables applications to run on processors dispersed geographically and administratively with the goal to provide greater capacity of processing (wich implies to reduce the response time) and more cost-benefit than traditional platforms. Moreover, a grid allows the sharing of large amounts of distributed data and provides ways to implement security and authentication in order to maintain the privacy and data independency. This dissertation presents an architecture for integrating data of fingerprints geographically dispersed through the use of grid computing to support the demands of high computacional processing. For this, provides information sharing through an environment secure, consistent, independent and with low cost, based an open source solutions and heterogeneous hardware. To validate the architecture, we developed an application for criminal identification through the use of fingerprints that can run on platforms such as personal computers (desktop or notebook) and mobile devices - more specifically PDA's. The results from tests performed in some representatives scenarios attest that grid computing provides a better response time for this type of application and enables the use of AFIS systems in large scale.

Keywords: Fingerprint, Grid Computing, Distributed Search

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AFIS	*Automated Fingerprint Identification System*
ANSI	*American National Standards Institute*
API	*Application Programming Interface*
CE	*Computer Element*
DRM	*Distributed Resource Management*
DRMAA	*Distributed Resource Management Application API*
FAR	*False Acceptance Rate*
FBI	*Federal Bureau of Investigation*
FRR	*False Rejection Rate*
GGF	*Global Grid Forum*
GIIS	*Grid Index Information Service*
GRAM	*Globus Resource Allocation Manager*
GRIS	*Grid Resource Information Service*
GSI	*Grid Security Infrastructure*
GSM	*Global System for Mobile Communications*
IAFIS	*Integrated Automated Fingerprint Identification System*
ID	Impressão Digital
IETF	*Internet Engineering Task Force*
ITL	*Information Technology Laboratory*
J2ME	*Java 2 Micro Edition*
JPEG	*Joint Photographic Experts Group*
MDS	*Monitoring and Discovery Service*
MOM	*Message-Oriented Middleware*
NIST	*National Institute of Standards and Tecnology*
OGF	*Open Grid Forum*
OGSA	*Open Grid Service Architecture*

OGSA-DAI	*Open Grid Services Architecture - Data Access*
OGSI	*Open Grid Service Infrastructure*
OV	Organização Virtual
PDA	*Personal Digital Assistant*
PKI	*Public Key Infrastructure*
RAM	*Random Access Memory*
RFT	*Reliable File Transfer*
RTF	*Reliable File Transfer*
SDK	*Software Development Kit*
SGBD	Sistema Gerenciador de Base de Dados
SGML	*Standardized Generalized Markup Language*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SSL	*Secure Socket Layer*
UDDI	*Universal Description Discovery and Integration*
W3C	*World Wide Web Consortium*
WAN	*Wide Area Network*
WSDL	*Web Service Definition Language*
WSQ	*Wavelet Scalar Quantization*
WSRF	*Web Service Resource Framework*
XML	*eXtensible Markup Language*

CAPÍTULO 1

INTRODUÇÃO

O processo de reconhecimento através da impressão digital consiste em comparar a imagem capturada com as existentes em um banco de dados. Essa comparação, antigamente, era feita apenas pelo processo manual, onde era necessária a utilização de lentes de aumento para uma melhor análise da imagem. Este processo demandava muito tempo e a depender da quantidade de imagens a serem comparadas se tornava inviável. Além disso, a captura da digital era feita apenas pelo método com tinta em papel cuja resolução é baixa. Atualmente, são utilizados sensores específicos para captura da impressão digital que garantem alta resolução e qualidade da imagem, o que facilita a busca de forma automatizada.

O uso automatizado da impressão digital, como mecanismo de identificação de indivíduos, tem sido cada vez mais explorado. No Brasil, pode-se citar a sua utilização para a confecção de passaportes, do registro geral e da carteira de motorista, na identificação criminal em alguns Estados e mais recentemente, no processo eleitoral de algumas cidades. A agilidade e segurança providas por esse tipo de procedimento é o que tem motivado o investimento cada vez maior nessa área. Quando o processo de reconhecimento envolve buscas em bases únicas e com poucos registros, o investimento em tecnologias de grande porte, como supercomputadores e software especializado, poderá vir a ser viável; porém, quando envolve aplicações com grandes proporções de dados (milhões de registros), torna-se custoso investir nesse tipo de tecnologia, como é o caso do seu uso para identificação civil e criminal no Brasil.

Sistemas automatizados de identificação através da impressão digital (AFIS) podem ser utilizados em diversos âmbitos. Quando empregados no reconhecimento de duas impressões digitais (um para um) são simples e eficientes. O grande entrave consiste em efetuar verificações de um para muitos. A depender da ordem de grandeza da base de dados, esse procedimento pode levar muito tempo, devido a grande necessidade de processamento requerida. Alternativas, como o uso de programação paralela e distribuída e *cluster* de computadores podem ser adotadas para obter melhor desempenho e redução do tempo de busca. Porém, tais alternativas podem vir a ser tornar insuficientes quando a busca se dá em grandes bases de dados que têm crescimento constante e, além disso, possuem restrições de segurança para assegurar a privacidade dos dados compartilhados. Tais fatos dificultam, por exemplo, a implantação de um sistema AFIS para reconhecimento criminal em escala nacional. Nesse contexto, a infraestrutura de grade computacional pode ser utilizada para solucionar esse problema.

As grades computacionais viabilizam a execução de aplicações sobre processadores heterogêneos e dispersos geografica e administrativamente. Através do seu uso é possível

compartilhar grandes quantidade de dados (independente de onde estejam localizados), e dispor de meios para implementar recursos de segurança e autenticação afim de manter o sigilo dos dados. Os recursos são compartilhados através de uma organização virtual dinâmica e multiinstitucional, onde diversos domínios administrativos coexistem podendo ter suas próprias políticas de utilização. Sendo assim, a utilização dessa infraestrutura apresenta-se como uma boa opção para o processo de identificação de indivíduos através da impressão digital, numa larga escala.

Este trabalho apresenta uma arquitetura para busca intensiva de dados de impressão digital dispersos geograficamente, a partir da utilização de uma infraestrutura de grades computacionais. A arquitetura proposta permite o compartilhamento de informações através de um ambiente seguro, consistente, multi-plataforma, independente e fundamentado em soluções de código aberto e hardware heterogêneo. A partir da arquitetura foi desenvolvida uma aplicação para identificação criminal de indivíduos através do reconhecimento de impressão digital. Com o intuito de avaliar a viabilidade da arquitetura proposta e da aplicação subjacente desenvolvida, foram realizados testes de análise de desempenho em cenários representativos, envolvendo máquinas e hardware heterogêneos, além de infraestruturas de rede diferentes. Os cenários utilizados nos testes envolveram busca centralizada em uma única máquina, em ambiente de cluster e na grade computacional. Uma análise comparativa do resultado dos testes, nesses cenários, foi realizada e comprovou que o uso de grade computacional para esse tipo de aplicação melhora o seu tempo de resposta e viabiliza o uso de sistemas AFIS numa escala mais ampla.

1.1 MOTIVAÇÃO E OBJETIVOS

Os Estados brasileiros utilizam essencialmente a impressão digital como recurso para identificação dos indivíduos. No Estado da Bahia, até pouco tempo atrás, o reconhecimento através da impressão digital na Polícia Técnica era feito, exclusivamente, pelo processo manual através do auxílio de lentes de aumento. A busca e a comparação manual de impressões digitais, aliado ao grande volume de arquivos de fichas reduz o escopo do reconhecimento e torna esse procedimento bastante lento. A própria impressão em tinta no papel pode fazer com que o desenho digital não fique nítido, afetando a qualidade da imagem e tornando a verificação da impressão digital extremamente difícil. O procedimento manual somente atende às necessidades de busca do tipo um para um, sendo inviável efetuar buscas sem ter um conjunto reduzido de suspeitos.

As diversas limitações decorrentes de um processo de identificação manual, aliada a necessidade cada vez maior de controle de segurança, motivaram o governo federal a iniciar em 2004 um projeto de automatização do processo de reconhecimento de impressões digitais, empregando a tecnologia AFIS, através da utilização do software GRIAULE (GRIAULE, 2008a). Tal tecnologia é hoje adotada em diversas instituições públicas de vários países. Para tanto, foi efetuada a compra de um sistema importado da França, a um custo de US$ 36 milhões (dados de (GRIAULE, 2008b)), a ser aplicado na informatização das etapas necessárias para a identificação de criminosos, que serão úteis na investigação

policial e na identificação de autoria de delitos.

Diante de estudos realizados em aplicações para reconhecimento através da impressão digital, foram identificadas as seguintes limitações:

• No Brasil, para obter informações civis e criminais de todos os Estados é necessário fazer acesso a uma base centralizada, a base da Polícia Federal situada em Brasília.

• Tempo de busca muito alto para o processo de reconhecimento de impressão digital em bases de grandes dimensões, o que implica na necessidade de utilizar supercomputadores e software especializado, tal como bases de dados;

• Muitas soluções utilizadas atualmente são proprietárias e específicas de fabricantes, além de não suportarem crescimento em escala;

• Até onde sabemos, no Brasil, não existem soluções propostas na academia ou na indústria que viabilizem o reconhecimento automático de impressão digital em bases de dados de grandes proporções e na escala de um país.

Diante dessas limitações, as seguintes motivações fizeram surgir o desenvolvimento desse trabalho:

• Apresentar uma solução que garanta crescimento em escala e melhoria de desempenho para o processo de reconhecimento de indivíduos através da impressão digital;

• Utilizar software e hardware de código aberto para reduzir custos e permitir o uso abrangente e democrático da solução apresentada;

• Possibilitar o reconhecimento unificado de identificações digitais envolvendo as bases dos diversos Estados do Brasil, em alternativa ao acesso centralizado à base da Polícia Federal;

• Desenvolvimento de uma solução inovadora, através do uso de grades de computadores, para viabilizar a busca ágil de impressões digitais numa base de imagens de grandes proporções, além de possibilitar crescimento em escala e melhoria de desempenho;

• Projetar uma solução que obedeça a requisitos de segurança e sigilo dos dados imprescindíveis para este tipo de aplicação;

• Apresentar uma arquitetura que poderá ser adaptada para buscas de outros dados biométricos, tais como o reconhecimento de face, íris e voz.

1.1.1 Objetivos

Este projeto tem como principal objetivo estudar e avaliar o uso de grades computacionais para o reconhecimento de impressões digitais afim de propor um sistema que se fundamente em soluções de código aberto e hardware heterogêneo. Seus objetivos específicos contemplam:

• Estudar o processo de reconhecimento de impressões digitais desde a captura da

imagem, através de sensores, até os algoritmos que realizam a verificação;

• Modelar o processo de reconhecimento de impressão digital numa grade computacional segura e confiável. O que envolve a modelagem da base para manipulação dos dados e identificação das tarefas a serem distribuídas e executadas na grade computacional;

• Desenvolver uma arquitetura que permita o compartilhamento de informações de impressões digitais através do uso da tecnologia de grade computacional baseado em modelo de camadas funcionais independentes mas que se relacionam para prover o serviço;

• Validar a arquitetura proposta através da implementação de um protótipo de um sistema automatizado de identificação de indivíduos através da impressão digital e que contemple desde a captura da imagem até o reconhecimento a partir da base de dados modelada na grade computacional.

1.2 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está dividida em sete capítulos que apresentam o estado da arte das tecnologias utilizadas, a arquitetura e implementação da solução proposta, apresentação e análise dos resultados obtidos, e a conclusão e trabalhos futuros.

O Capítulo 2 apresenta o estado da arte relacionado à impressão digital. Para isso apresenta: o histórico da sua utilização, os métodos utilizados no processo de reconhecimento, os elementos constitutivos da impressão digital, dando ênfase às minúcias que são imprescindíveis na comparação, as classes relacionadas a uma impressão digital e as formas de captura da imagem da impressão digital. Além disso, explica o funcionamento de um sistema automatizado de identificação pela impressão digital e padrões relacionados. Por fim, detalha o NFIS, software desenvolvido pelo FBI para identificação automatizada através da impressão digital, dando destaque aos pacotes utilizados nesse projeto.

O Capítulo 3 detalha a tecnologia de grade computacional e apresenta seus conceitos, características e arquiteturas, bem como padrões recomendados, além da infraestrutura necessária para o seu funcionamento. São destacadas as tecnologias utilizadas na construção do projeto, tais como: o *middleware* de grade Globus Toolkit 4, o OGSA-DAI – utilizado para permitir o compartilhamento de dados oriundos de diversas fontes de dados (e de diferentes fabricantes), os escalonadores GridWay e Torque – responsáveis por definir quais os recursos mais adequados para prover um bom desempenho para aplicação num ambiente de grade computacional e de *cluster*, respectivamente.

O Capítulo 4 apresenta a arquitetura proposta para o reconhecimento de impressões digitais através do uso de grades computacionais. Inicialmente são apresentados os objetivos, premissas e restrições da arquitetura. Em seguida, a arquitetura é detalhada, sendo apresentada suas camadas com respectivos componentes, que englobam a definição das interfaces de acesso aos dados, os componentes do Globus Toolkit 4 utilizados, a definição do processo de escalonamento das tarefas na grade e o modelo de acesso a dados.

O Capítulo 5 apresenta um estudo de caso para validar a arquitetura proposta no

Capítulo 4. O estudo de caso trata de uma aplicação para identificação criminal através da impressão digital. Inicialmente é detalhada a descrição do problema da identificação criminal com a solução proposta, seguida do cenário da distribuição dos dados que apresenta os passos necessários para execução da aplicação. Na sequência são explicados os serviços de grade desenvolvidos e as interfaces implementadas. Por fim, é apresentado o modelo definido para a distribuição dos dados para que seja possível fazer acesso às informações disponibilizadas na grade computacional.

O Capítulo 6 descreve os testes realizados e os resultados obtidos para avaliar o protótipo desenvolvido com base na arquitetura proposta no Capítulo 4. Para isso, são apresentadas a infraestrutura utilizada para efetuar os testes, são detalhados os cenários definidos para os testes com as respectivas avaliações dos resultados.

O Capítulo 7 conclui o trabalho, apresentando os objetivos atingindos e as contribuições alcançadas com o desenvolvimento desse projeto seguido das sugestões de trabalhos futuros para dar continuidade ao mesmo.

CAPÍTULO 2

IMPRESSÃO DIGITAL

As impressões digitais humanas foram descobertas em um grande número de artefatos arqueológicos e itens históricos. Porém, só a partir de 1684 é que foram usadas como base científica para garantir a individualidade das pessoas (MALTONI et al., 2005). Registros históricos (COSTA, 2001) apresentam a utilização da impressão digital como meio de identificação desde o ano 650 onde eram utilizadas nos processos de divórcio, para firmar acordos e pelos analfabetos para legislação de papéis. A partir de 1300 os chineses passaram a utilizar a impressão digital para identificação de crimes. Desde então, várias pesquisas e estudos foram feitos relacionados à impressão digital (desenhos digitais, formação anatômica das impressões digitais, entre outros). Em 1882, foi lançado em Paris, por Alfonse Bertillon, o Sistema Antropométrico que foi considerado o primeiro sistema científico de identificação. Em 1888 a fim de estabelecer um sistema de identificação mais seguro que a antropometria, Francis Galton lançou as bases científicas da impressão digital. Em seguida, vários sistemas e nomes foram propostos até que em 1900, Edward Richard Henry publicou, na Inglaterra, seu livro entitulado "Classificafication and Uses of Fingerprints"onde apresentou um sistema com quatro tipos fundamentais de impressões digitais explicadas na Seção 2.3. Em 1901 o sistema de Henry foi adotado oficialmente na Inglaterra pelo Serviço de Polícia Metropolitano (*Scotland Yard*).

A biometria é utilizada para medir as características que definem a individualidade das pessoas (COSTA, 2001). Essas características podem ser tanto físicas quanto comportamentais. Impressão digital, face, íris, retina, assinatura manuscrita, dentre outros, fazem parte dos elementos que constituem os métodos biométricos para reconhecimento de indivíduos. A escolha do método biométrico a ser utilizado deve levar em consideração o propósito da aplicação a ser utilizada.

A impressão digital é um dos métodos biométricos mais utilizados atualmente para identificação humana, por se tratar de um método simples de ser implantado, não gerar muitos custos para utilização, além de apresentar ótimos resultados em termos de precisão, segurança e aceitação. Isto deve-se a duas propriedades fundamentais: (i) **persistência**: uma impressão digital não muda com o tempo e, (ii) **individualidade**: cada pessoa tem uma única impressão digital (ZEGARRA; LEITE; TORRES, 2006). A Figura 2.1 mostra a distribuição dos métodos biométricos mais utilizados em 2005(dados de (VIOLA, 2006)).

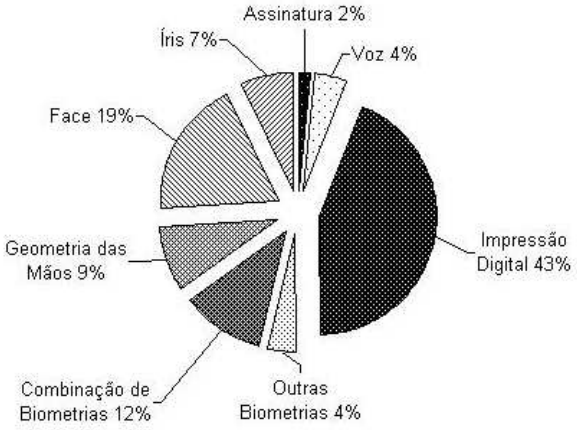

Figura 2.1. Métodos Biométricos mais Utilizados (VIOLA, 2006)

2.1 BIOMETRIA

A utilização da biometria nas organizações, no processo de autenticidade das pessoas, tem alcançado um avanço bastante significativo. Isto deve-se ao fato de que as características biométricas não podem ser roubadas, esquecidas e dificilmente podem ser forjadas, em contrapartida aos métodos tradicionais utilizados, como a utilização de senhas. Segundo (MALTONI et al., 2005), qualquer característica física e/ou comportamental de um indivíduo pode ser usada como um identificador biométrico. Para isso, é necessário que satisfaça aos seguintes requisitos:

• Universalidade: cada pessoa deve ter um conjunto de características;

• Unicidade (distinção): indica que quaisquer duas pessoas devem ser suficientemente diferentes em relação aos seus identificadores biométricos;

• Permanência (imutabilidade): as características biométricas de um indivíduo devem ser suficientemente invariantes (respeitando os critérios de comparação) ao longo do tempo;

• Critério Quantitativo: indica que as características biométricas podem ser medidas quantitativamente.

De acordo com os critérios citados acima, a impressão digital é um método biométrico, pois: (i) todas as pessoas possuem uma impressão digital, o que demonstra o critério da universalidade; (ii) as impressões digitais são únicas e permanentes, ou seja, as digitais apresentam um conjunto de características pessoais que é formado na gestação (aos 7 meses) e assim permanecem por toda a vida - a não ser que ocorra amputação dos dedos ou corte bastante profundo; (iii) através da comparação de pontos característicos, denominados minúcias (as minúcias serão explicadas na seção 2.2.1) é possível fazer a identificação de uma pessoa. Isto satisfaz o critério quantitativo.

Além dos requisitos de universalidade, unicidade, permanência e medidas quantitativas, citados acima, existem, na prática, outros critérios de fundamental importância, a saber:

• Desempenho: refere-se ao comportamento do sistema em relação a robustez, velocidade e exatidão do reconhecimento;

• Aceitabilidade: refere-se a aceitação, pelas pessoas, de que um identificador biométrico seja utilizado para garantir a sua individualidade;

• Intervenção: reflete o quão fácil é fraudar o método de reconhecimento. Por exemplo, a impressão digital pode ser falsificada com um simples molde do dedo em silicone ou gelatina, que são capazes de reproduzir minúcias com alto grau de fidelidade; O reconhecimento por voz, por ter um número de fatores de análise pequeno, pode ser copiada sem necessitar de grandes recursos (cantores *covers* imitam as vozes dos artistas).

2.1.1 Reconhecimento de Impressões Digitais

O reconhecimento através das impressões digitais pode ser empregado em diversas áreas, como: sistemas de segurança, identificação de criminosos, transações financeiras, controle de acesso a locais restritos, controle de frequência de funcionários, acesso em redes corporativas, validação de documentos, autenticação de portadores de cartões, comprovação de identidade, entre muitas outras aplicações. A depender do contexto da aplicação, um sistema biométrico, de reconhecimento, pode ser denominado sistema de verificação ou sistema de identificação. E em relação ao domínio da aplicação, os sistemas biométricos podem operar como um sistema *on-line* ou um sistema *off-line*.

Um sistema de **verificação** autentica a identidade de uma pessoa através da comparação da característica biométrica capturada com o seu próprio molde (*template*) pré-armazenado no sistema. O *template* consiste de dados numéricos, que identificam a localização dos pontos característicos (ver Seção 2.2.1), gerados a partir da imagem da ID adquirida. Um sistema de verificação executa a comparação um-para-um confirmando se o indivíduo em questão é quem estar dizendo ser. Já um sistema de **identificação** reconhece um indivíduo através da comparação numa base de dados de *templates*. Nesse sistema a comparação é de um-para-muitos identificando se o indivíduo em questão pertence a um determinado grupo, como o de criminosos, por exemplo.

Um sistema ***on-line*** requer que o reconhecimento seja executado de forma rápida, e uma resposta imediata é obtida. Normalmente, os sistemas *on-line* são totalmente automáticos e requerem que as características biométricas sejam capturadas usando um digitalizador com sensor apropriado. Um sistema ***off-line*** normalmente não requer que o reconhecimento seja executado de maneira imediata e um tempo de resposta relativo é tolerado. Sistemas *off-line* são, normalmente, semi-automáticos, e a aquisição biométrica pode ser através de um digitalizador *off-line*. As formas de captura de uma impressão digital serão detalhadas nas Seção 2.4.

2.2 ELEMENTOS CONSTITUTIVOS DE UMA IMPRESSÃO DIGITAL

O método de identificação humana através da impressão digital é denominado de dactiloscopia (do grego *d'akytos* (dedos) e *skopein* (examinar)) ou datilograma (KAZIENKO, 2003). Para o processo de identificação, a dactiloscopia utiliza os elementos constitutivos do desenho digital, a saber (KAZIENKO, 2003), alguns identificados na Figura 2.2:

• Linhas Pretas: são as linhas impressas do datilograma que correspondem às cristas papilares ou estrias;

• Linhas Brancas: referem-se as regiões (vales) que separam as linhas impressas de um datilograma;

• Poros: pequenos orifícios localizados sobre as linhas impressas do datilograma;

• Minúcias ou Pontos Característicos: são peculiaridades que ocorrem nas cristas papilares e que podem distinguir uma impressão digital de outra;

• Linhas Albodactiloscópicas: formadas através da interrupção de duas ou mais cristas papilares. Surgem devido ao enrugamento da pele. Essas linhas não devem ser confundidas com as linhas brancas;

• Delta: triângulo formado pelas cristas papilares. Pode ser formado por dois processos: pela bifurcação de uma linha simples, ou pela brusca divergência de duas linhas paralelas. Desempenha função importante na impressão digital porque é utilizada como referência para a determinação de uma classe (ver Seção 2.3);

• Núcleo: ponto localizado na área central da impressão digital. Junto com o delta constituem os pontos singulares utilizados para classificação de uma impressão digital.

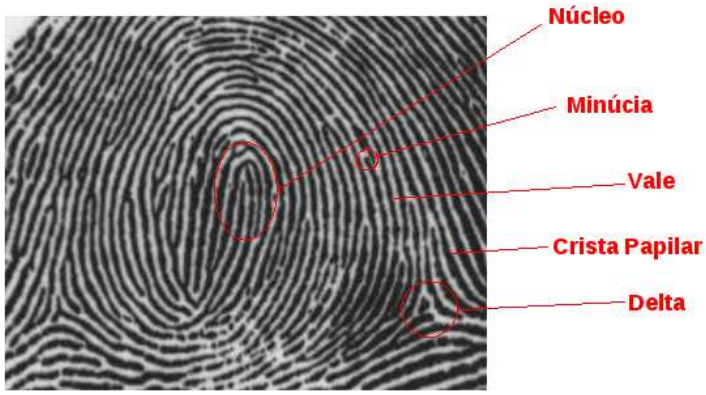

Figura 2.2. Elementos Constitutivos de uma Impressão Digital

2.2.1 Minúcias

As minúcias, também denominadas pontos característicos, são imperfeições que se encontram nas cristas papilares e são responsáveis por definir a unicidade de uma impressão digital. Também são chamadas de detalhes de Galton (MALTONI et al., 2005), em homenagem a Francis Galton que foi o pioneiro a identificar que as minúcias não se modificam ao longo da vida, na grande maioria dos indivíduos.

Essas imperfeições podem ser linhas que terminam abruptamente ou que se bifurcam, e são representadas através dos seguintes aspectos: crista final, bifurcações, ilha, crista curta, espora e cruzamento. A Figura 2.3 apresenta uma ilustração das mesmas. Apesar de apresentar vários aspectos, conforme citados acima, normalmente a classificação das minúcias resume-se em apenas dois tipos: (i) final de linha - ponto onde uma crista termina, e (ii) bifurcação - linha que em um determinado trajeto se divide em duas. Esses dois tipos ocorrem com muito mais frequência e os demais nada mais são do que variações destes.

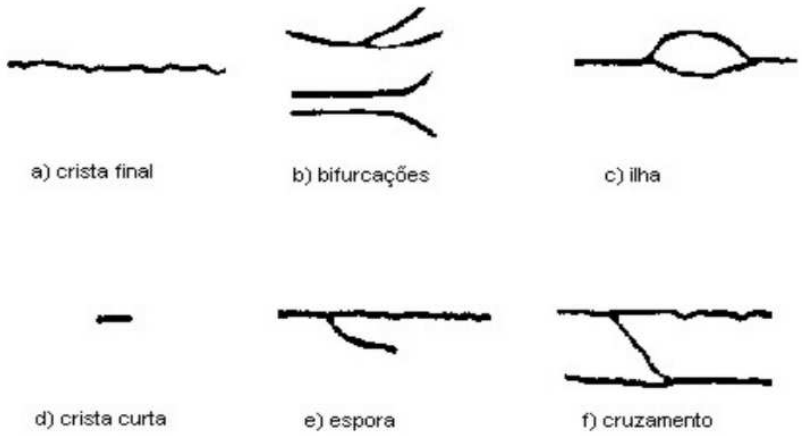

Figura 2.3. Tipos de Minúcias (COSTA, 2001)

Cada dedo tem, em média, de quarenta a cem minúcias, a depender da qualidade da imagem obtida. Para as imagens obtidas em locais de crimes (denominadas impressões latentes), por exemplo, a quantidade de minúcias detectadas normalmente é bastante reduzida. Para localizar uma minúcia, deve-se identificar: suas coordenadas (vertical e horizontal) do sentido da linha a qual ela pertence, o ângulo que esta forma com o eixo horizontal e o seu tipo, como pode ser visualizado na Figura 2.4. Para reconhecimento de uma impressão digital é necessário que sejam encontradas, no mínimo, doze minúcias idênticas (mesmo tipo) e com a mesma localização (COSTA, 2001). Porém, essa quantidade pode variar para mais, de acordo com a determinação do sistema de reconhecimento.

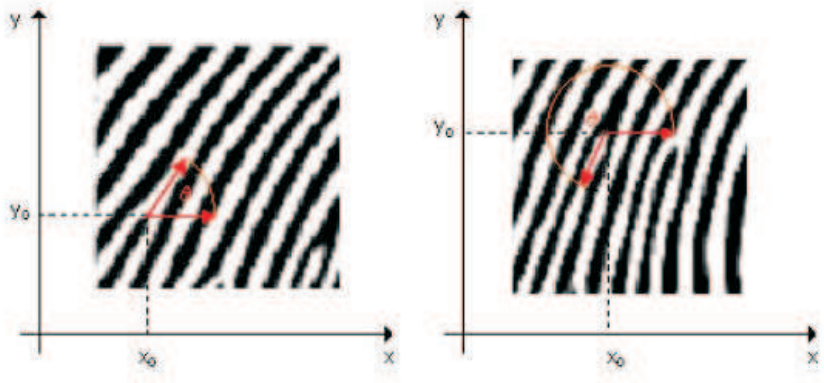

Figura 2.4. Localização de Minúcias (VIOLA, 2006)

2.3 CLASSIFICAÇÃO DE UMA IMPRESSÃO DIGITAL

Todos os dias, grandes quantidades de impressões digitais são coletadas e armazenadas para uma grande variedade de aplicações, como, por exemplo, para registros civis e criminais. Uma identificação automática, baseada em impressões digitais, requer que uma ID seja comparada com grandes números de outras armazenadas em uma base de dados. Para reduzir o tempo de busca e a complexidade computacional, é necessário que as impressões digitais sejam classificadas de maneira exata e consistente para que a impressão digital de entrada seja comparada apenas com um subconjunto, relacionado à classe, em questão, da base de dados. A classificação é uma técnica usada para atribuir a uma impressão digital uma das várias classes já estabelecidas na literatura.

Em 1823, Purkinje propôs as primeiras regras de classificação das impressões digitais, classificando-as em nove categorias de acordo com as configurações das cristas papilares (MALTONI et al., 2005): curva transversal, crista central longitudinal, lista oblíqua, laço oblíquo, verticilo em amêndoa, verticilo espiral, elipse, circulo, e verticilo duplo. O primeiro estudo científico aprofundado sobre classificação de impressões digitais foi feito por Francis Galton em 1888, que dividiu as impressões digitais em três classes principais: arco, presilha e verticilo. Nesse mesmo período, Juan Vucentich, um oficial da policia argentina, desenvolveu um sistema diferente de classificação dividida em arco, presilha externa, presilha interna e verticilo. O sistema de classificação de Vucentich continua sendo usado em muitas cidades da Espanha. Em 1900, Edward Henry refinou a classificação de Galton incluindo algumas novas classes. Essa classificação, também chamada Sistema de Henry, é bastante utilizada até os dias atuais, em todo o mundo, e corresponde as seguintes classes (COSTA, 2001), ilustradas na Figura 2.5:

• Arco Plano - não apresentam delta e as linhas atravessam de um lado para o outro de forma abaulada. As linhas dactilares (linhas pretas em uma impressão tintada em papel) formam-se em um lado e tendem a terminar no outro lado;

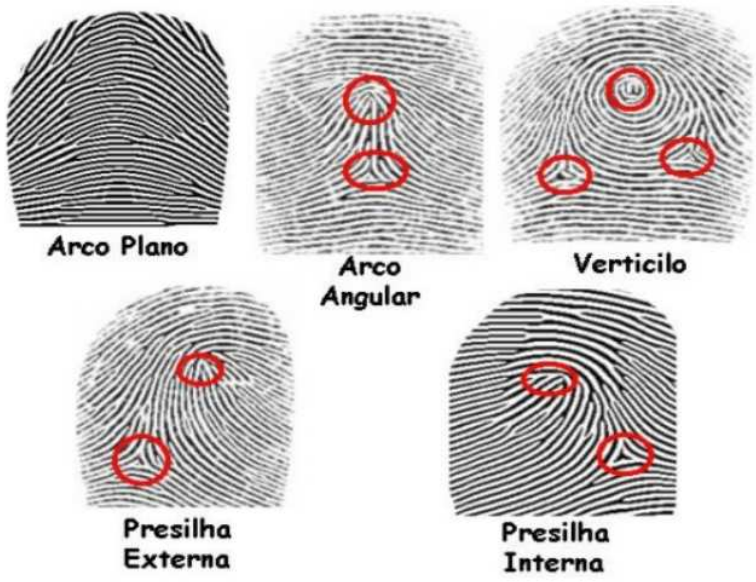

Figura 2.5. Sistemas de Henry (COSTA, 2001)

• Arco Angular - apresentam acentuada elevação das linhas no centro, em forma de tenda. Pode apresentar um delta, mas sem linha ou fragmento de linha, entre o delta e o núcleo;

• Verticilo - apresentam dois deltas, sendo um à direita e outro à esquerda do observador; as linhas nucleares ficam encerradas entre os dois deltas, assumindo configurações variadas;

• Presilha Externa - apresentam um delta à esquerda do observador, as linhas dactilares correm para a direita do observador, ou seja, as linhas formam-se na direita do observador, curvam-se no centro da impressão e tendem a voltar para o mesmo lado;

• Presilha Interna - apresentam um delta à direita do observador, as linhas dactilares correm para a esquerda em forma de laçadas, ou seja, as linhas formam-se à esquerda do observador, curvam-se no centro da impressão e tendem a voltar para o mesmo lado.

Segundo pesquisa feita pelo FBI (Federal Bureau of Investigation) (COSTA, 2001) as ocorrências das classes nas impressões digitais dos indivíduos são distribuídas da seguinte maneira: 65% são presilhas (interna e externa), 30% são verticilos e 5% são arcos. Portanto, as imagens de impressões digitais não estão distribuídas de maneira uniforme nas bases de dados.

2.4 FORMAS DE CAPTURA DE UMA IMPRESSÃO DIGITAL

O método mais conhecido, historicamente, para capturar uma impressão digital é através do uso de tinta preta sobre os dedos, também chamado de método *off-line* ou *ink-technique*. Atualmente, com a evolução da tecnologia tem-se utilizado um digitalizador eletrônico que possui um sensor específico que, ao contato com a superfície do dedo, captura a imagem da impressão digital de forma automática e precisa. Esse método é chamado de *on-line* ou *live-scan* (MALTONI et al., 2005).

A imagem *off-line* é adquirida através do método "tintado em papel"(*ink-paper*), onde a superfície do dedo é mergulhada em uma tinta e pressionada em um papel. Para armazenamento em bases de dados e/ou utilização por sistemas computacionais, a imagem deve ser capturada através de um digitalizador de papel ou fotografada por uma câmera de alta qualidade. É necessário, também, que sejam aplicados algoritmos de segmentação para melhoramento da imagem, remoção de "lixos"como letras, números e borrões que aparecem nas fichas criminais e civis, por exemplo. Um tipo especial das imagens *off-line* é a denominada impressão digital latente. Esse tipo de imagem é adquirida em locais de crime. O óleo natural da pele faz com que a impressão digital seja depositada na superfície em que esta teve contato.

A imagem *live-scan*, por outro lado, é adquirida através do toque da superfície do dedo em um sensor que é capaz de digitalizar a impressão digital através do contato. O dispositivo capaz de capturar a imagem da impressão digital dessa forma é denominado de *scanner* (digitalizador). A parte mais importante do *scanner* de impressão digital é o sensor. É através deste que a imagem da impressão digital é formada. Existem diversos tipos de sensores que se diferem na tecnologia utilizada, que pode ser: sensor óptico, sensor de estado sólido ou sensor de ultra-som. Destes o mais utilizado é o sensor óptico por apresentar melhor qualidade da imagem e maior área de captura. Porém, a utilização dos sensores de estado sólido tem crescido bastante devido ao seu tamanho compacto e pela facilidade de introduzí-lo em *laptops*, telefones celulares, PDA's, entre outros.

Apesar dos primeiros *scanners*, para impressão digital, terem surgido na década de 60 (VIOLA, 2006), ainda hoje a técnica *off-line* é muito utilizada. Isto ocorre por existirem grandes arquivos de dados capturados dessa forma e por causa das impressões latentes que não podem ser capturadas pelos *scanners*. Com a evolução dos *scanners*, os sistemas têm convivido com as duas formas em paralelo.

Os parâmetros principais que caracterizam uma imagem de impressão digital são: resolução, área, número de pixels[1], exatidão da geometria, contraste e distorção da geometria. Para maximizar a compatibilidade entre as imagens das impressões digitais e garantir uma boa qualidade na aquisição das impressões, os Serviços de Informação da Justiça Criminal dos Estados Unidos determinou um conjunto de especificações que regulam a qualidade e formato tantos das impressões digitais quanto dos modos de captura

[1]Pixel é a aglutinação de *Picture* e *Element*, ou seja, elemento de imagem, sendo Pix a abreviatura em inglês para *Picture*. É o menor ponto que forma uma imagem digital, sendo que o conjunto de milhares de pixels formam a imagem inteira.

(*off-line* e *live-scan*). Essas definições serão vistas na Seção 3.1 onde são apresentados os padrões para captura de impressões digitais. A Figura 2.6 mostra a diferença entre uma imagem capturada pelo método *off-line* e *live-scan*.

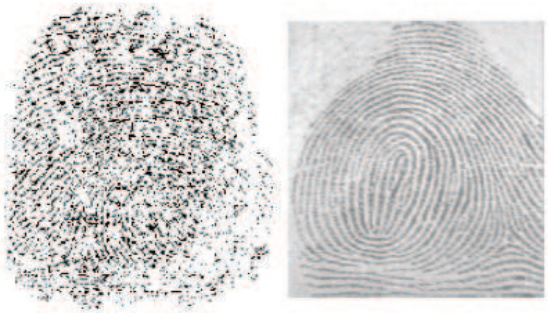

Figura 2.6. Impressão tintada em papel e adquirida por um sensor (COSTA, 2001)

2.4.1 Impressões Digitais Sintéticas

Para testar a eficiência dos sistemas de reconhecimento de impressões digitais, normalmente, é necessária uma grande base de imagens. Nem sempre é possível, nessa fase, obter impressões digitais reais. Isto deve-se ao fato de que os proprietários dessas bases, não as disponibiliza por se tratarem de dados pessoais de indivíduos e, por isso, sigilosos. Para resolver esse problema, são utilizadas, muitas vezes, bases de impressões digitais sintéticas. As impressões sintéticas simulam as características reais da aquisição de impressões de dedos dos indivíduos. Os programas que geram impressões sintéticas, permitem que sejam produzidas impressões simulando os mais diversos tipos de sensores de captura (os sensores foram apresentados na seção 2.4). A Figura 2.7 apresenta uma impressão digital sintética gerada pelo programa *SFINGE* simulando a captura por um sensor capacitivo.

Em (MALTONI et al., 2005) é apresentado o *SFINGE* como exemplo de um aplicativo de geração de impressões sintéticas. O *SFINGE* inicialmente cria uma impressão denominada de mestre. Esta contém um conjunto de parâmetros que representam as características de unicidade e imutabilidade de um dedo sintético. A partir desta impressão digital mestre, é possível gerar outras através da troca de alguns parâmetros que controlam a aparência da impressão digital.

2.5 SISTEMA AUTOMATIZADO DE IDENTIFICAÇÃO PELA IMPRESSÃO DIGITAL (AFIS)

O crescimento do uso de impressões digitais, para identificação de pessoas, associado a enorme quantidade de registros, cujo crescimento é progressivo (principalmente na área civil e criminal), fez surgir a necessidade de um método automatizado. Esse método é

Figura 2.7. Impressão Digital Sintética gerada pelo *SFINGE* (MALTONI et al., 2005)

uma evolução do processo manual, onde é necessária a participação de um perito para identificação. O processo manual é um processo tedioso, custoso e que consome muito tempo, por isso, muitas vezes inviável. Sua inviabilidade é notória na identificação criminal, onde muitas vezes torna-se impossível a não ser que já existam suspeitos. Isto deve-se ao imenso número de registros a serem analisados, por um perito, apenas com o auxílio de lentes de aumento.

Em torno de 1960, o FBI (Escritório Federal de Investigação dos Estados Unidos) e algumas instituições de Segurança Pública da Inglaterra e da França iniciaram o desenvolvimento de sistemas automatizados de identificação através das impressões digitais, denominados AFIS. Hoje essa tecnologia é utilizada por diversas instituições públicas como por exemplo, Secretarias de Seguranças Pública do Brasil, como a do Ceará, Tocantins, São Paulo e mais recentemente na Bahia.

Existem duas maneiras de se projetar um sistema AFIS. Na primeira, denominada um-para-um (1:1), o objetivo é confirmar se a pessoa é quem está dizendo ser. Esse é o caso, por exemplo, de sistemas que usam as impressões digitais como senha. A segunda maneira, denominada um-para-muitos (1:N), visa identificar uma determinada impressão para saber se esta já fora previamente capturada. A entrada de dados para o AFIS é uma impressão digital e sua saída é uma lista de possíveis indivíduos cuja impressão digital possua as mesmas características da impressão digital de entrada (mesmos tipos e localizações de minúcias). Nesse sistema é comparada a impressão digital de entrada com os registros de impressões digitais de um banco de dados.

2.5.1 Etapas de um Sistema AFIS

As etapas gerais de um sistema AFIS envolvem (KAZIENKO, 2003):

Interface com o Usuário
Nesta etapa é feita a aquisição da imagem da impressão digital, através de um *scanner*.

Também pode ser definido o tipo de consulta a ser realizada: se uma verificação ou identificação, visto que um AFIS pode realizar as duas tarefas;

Tratamento da Imagem

Nesta etapa, que pode existir ou não, a imagem passa por um tratamento para remoção de ruídos (letras, números, borrões, entre outros) e melhoramento da qualidade, para que então, possam ser identificados seus pontos característicos numa etapa seguinte;

Classificação da Imagem

Esta etapa consiste em definir a qual classe (classificação de Henry (ver Seção 2.3)) a imagem capturada pertence, e dessa maneira facilitar o processo de busca. Esta etapa também é opcional, porém, de grande utilidade, pois reduz o escopo de busca e consequentemente, diminui o tempo de processamento, já que a busca será feita só nas imagens de mesma classe. Além disso, aumenta a precisão na comparação e reconhecimento. Um excessivo esforço computacional é necessário para extração dos atributos que classificam as imagens, e estes ainda podem ser obtidos com erros ou ruídos. Vários métodos podem ser utilizados para classificar uma impressão digital, dentre os quais se destacam: estrutural, estatístico, sintático, matemático, através de redes neurais e híbidro (combina dois ou mais métodos para efetuar a classificação). Para maiores esclarecimentos sobre esses métodos consultar (MALTONI et al., 2005) e (COSTA, 2001).

Extração de Pontos Característicos

Embora a maioria dos AFIS utilizem as minúcias como método de comparação de impressões digitais, podem ser utilizados também outros métodos como (MALTONI et al., 2005): (i) baseados em correlação - onde duas imagens de impressão digital são sobrepostas e a correlação entre os pixels correspondentes são computadas analisando os diferentes ângulos de rotação em diferentes escalas de distorção; e (ii) baseados na forma das cristas - utilizado em imagens com qualidade ruim e onde a verificação através das minúcias torna-se inviável. Nesses casos, pode-se utilizar informações sobre a quantidade, tipo e posição dos deltas e vales, e relacionamento espacial e atributos geométricos das cristas para classificar a ID. Neste trabalho será apresentado apenas o método que utiliza a comparação de minúcias. Nesta etapa, são armazenadas o tipo de minúcia, sua direção, distância entre as minúcias e as suas localizações. Essas informações são denominadas *templates*, e são usadas no processo de verificação. Antes da extração das minúcias, os sistemas automatizados de verificação de impressões digitais efetuam as seguintes operações:

• Obtenção das Direções: consiste em localizar a direção das cristas para que possa ser identificado, posteriormente, a presença de bifurcações e finais de linhas;

• Binarização ou Limiar: consiste na conversão de imagens em tons de cinza para imagens binárias (preto e branco). Essa operação é denominada de *threshold*. Consiste em verificar os valores de intensidade dos pixels para determinar qual valor será atribuído - 0 para preto ou 255 para branco. A definição de qual valor será atribuído é feita através da comparação numérica dos pixels com um valor que é o *threshold*. Se o valor do pixel analisado for menor que o valor do *threshold*, atribui-se o valor 0, caso contrário, atribui-se o valor 255. Esse procedimento deve ser feito em pequenos blocos da imagem

(geralmente 8x8 ou 10x10), pois o valor do cinza não é o mesmo em diferentes partes da mesma. Devido a isso, o *threshold* é calculado para cada bloco da imagem usando o valor médio de cinza do bloco como o valor de *threshold*. Esse procedimento é também chamado de *threshold* adaptativo;

• Afinamento (*thinning*): utilizada para remover pontos isolados no fundo da imagem que podem ser ruídos devido a sujeira no leitor da impressão digital. Essa operação remove pixels não úteis para imagem sem alterar a estrutura da mesma. Os algoritmos de *thinning* consomem bastante tempo, pois a varredura da imagem é feita linha a linha, examinando qual o *pixel* que pode ou não ser apagado. Esta técnica pode ser executada diversas vezes até que seja detectado que não há mais *pixels* a serem removidos. Em (MALTONI et al., 2005) são apresentados alguns métodos de afinamento;

• Extração: para localizar as minúcias normalmente são utilizados moldes que percorrem a imagem para detectar os pontos característicos. Um dos métodos muito usado encontrado na literatura é o conceito de *Crossing Number*. Este método determina as propriedades de um *pixel* através da contagem do número de transições de preto e branco existentes nos 8 *pixels* vizinhos ao *pixel* analisado.

Verificação

Nesta etapa, o *template* da imagem capturada é comparado aos *templates* armazenados na base de dados. Diferentes impressões digitais do mesmo dedo podem gerar variações de uma mesma classe. Essas variações incluem deslocamento, distorção não-linear, rotação, variação da pressão no dispositivo de captura, condições da pele (seca ou oleosa), entre outros. Devido a isto, impressões digitais do mesmo dedo pode muitas vezes parecer diferentes e impressões de diferentes dedos podem parecer similares. Devido a esses problemas, uma série de fatores devem ser analisados para garantir que a impressão digital é do mesmo dedo, tais como (MALTONI et al., 2005): (i) concordância de configuração padrão global - que significa que duas impressões devem ser do mesmo tipo; (ii) concordância qualitativa - que requer que os detalhes correspondentes às minúcias sejam idênticos; (iii) fator quantitativo - que específica que pelo menos um certo número (no mínimo 12) de detalhes correspondentes às minúcias devem ser encontrados; e, (iv) detalhes correspondentes às minúcias - que devem ser interrelacionadas idênticamente (mesma posição, mesmo ângulo, mesmo tipo). Sendo assim, duas imagens de impressão digital são consideradas coincidentes quando suas minúcias estão localizadas na mesma posição e pertencem ao mesmo tipo. Um outro ponto importante que é considerado pelos sistemas AFIS e estão relacionados à verificação é o desempenho, que pode ser categorizado por duas medidas: taxa de falsa aceitação (FAR) e taxa de falsa rejeição (FRR).

O processo de verificação, num sistema de reconhecimento de impressão digital, é tipicamente a comparação com um valor denominado *threshold* que quantifica a similaridade entre o dado de entrada e a representação do template na base de dados. No método baseado em minúcias é verificado se a quantidade de minúcias correlacionadas é igual ou superior ao valor determinado de *threshold*. Se essa condição for aceita, afirma-se que a impressão digital pode ser do mesmo dedo, caso contrário, afirma-se que é diferente.

Como todo sistema computacional, um sistema AFIS pode apresentar erros devido a variação da qualidade das imagens, bem como seus tamanhos e métodos utilizados para captura. Esses erros são denominados taxa de falsa aceitação e taxa de falsa rejeição. A taxa de falsa aceitação, representada pela sigla FAR (*False Acceptance Rate*), indica que o sistema determinou que duas impressões digitais são iguais, quando na verdade não são. A taxa de falsa rejeição, representado pela sigla FRR (*False Rejection Rate*), indica que o sistema determinou que duas impressões digitais são diferentes quando na verdade são iguais.

Quanto menores forem essas taxas mais preciso se torna o sistema. Como essas taxas são antagônicas, à medida que se reduz uma, aumenta a outra. Por isso, devem ser analisadas de acordo com o objetivo do sistema. No caso de uma identificação criminal, por exemplo, é melhor reduzir a taxa de falsa aceitação, pois com a taxa de falsa rejeição elevada o que vai ocorrer é que o número de suspeitos irá aumentar, nesse caso é melhor que isto ocorra do que deixar de identificar um provável suspeito. A figura 2.8 apresenta um gráfico que ilustra a relação entre as taxas de erro citadas. A linha pontilhada ao centro representa o ponto de equilíbrio entre as taxas.

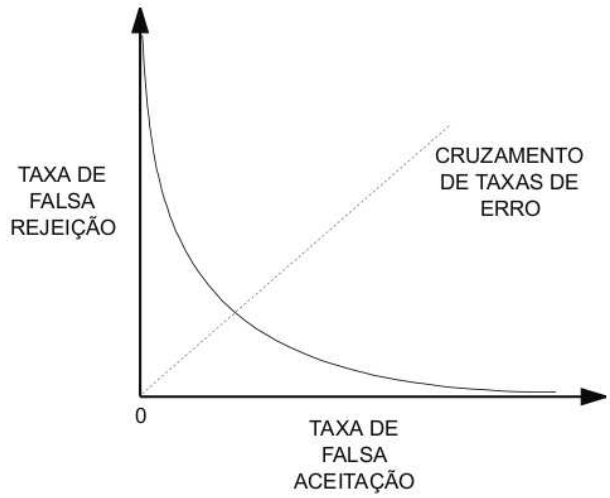

Figura 2.8. Taxas de Erro Associadas a Sistemas Biométricos (KAZIENKO, 2003)

2.5.2 Padrões

Com o crescimento do uso de sistemas AFIS, principalmente no âmbito comercial, tornou-se necessária a criação de padrões para permitir a interoperabilidade entre os diversos sistemas, além de garantir qualidade de software. A criação de padrões foi iniciada por orgãos Americanos, em especial o FBI e o NIST (*National Institute of Standards and Tecnology*) dos EUA.

Os principais padrões existentes são (KAZIENKO, 2003):

• **ANSI/NIST-ITL 1-2000:** a primeira versão surgiu em 1986 resultante da cooperação entre o ANSI (*American National Standards Institute*), NIST através do seu órgão ITL (*Information Technology Laboratory*) e o FBI. No início essa norma especificava apenas o formato de dados para impressões digitais e dados relativos às minúcias. Com o passar do tempo, incorporou atributos relativos a face, cicatriz e tatuagem. Esse padrão organiza os dados em uma estrutura de arquivos, registros, campos, sub-campos e itens de informação. De acordo com esses parâmetros os dados e seus formatos são definidos. Especifica, também, que as imagens da impressão digital devem ser capturadas por um leitor de impressões digitais ou similar operando em resolução de 500 pontos por polegadas. A tolerância permitida é de um por cento desse valor para menos. O valor máximo de resolução não é definido. As minúcias são especificadas através dos seguintes atributos: coordenada x, coordenada y e orientação ou ângulo.

• **FBI/IAFIS EFTS 7.0:** teve início nas décadas de 60 e 70 em paralelo a criação do ANSI/NIST. Detalha requisitos para que outras entidades possam estabelecer comunicação eletrônica com o AFIS do FBI, denominado IAFIS (*Integrated Automated Fingerprint Identification System*). Essa norma, em distinção a anterior, refere-se apenas a troca de informações relativas a impressão digital. Considera as seguintes classes: arco, presilha esquerda, presilha direita e verticilo. Assim, como o ANSI/NIST-ITL, os formatos de imagens JPEG e WSQ são aceitos. Esses são formatos de compressão de imagens com e sem perdas de informações. Os algoritmos de compressão com perda comprimem a imagem gerando perda de qualidade da mesma. Os algoritmos de compressão sem perda mantêm a qualidade da imagem. Tanto o ANSI/NIST-ITL quanto o EFTS detectam situações onde a definição do tipo de impressão digital (classificação) torna-se difícil devido a cicatrizes e amputação. Ambos também utilizam as minúcias como método de individualização da impressão digital. Essas normas também estabelecem um formato para troca de informações de impressões digitais, porém, não existe intercâmbio de *templates*, ou seja, um *template* gerado em um sistema não pode ser comparado com um gerado por outro. As Tabelas 2.1 e 2.2, disponibilizadas em (KAZIENKO, 2003) apresentam as codificações, dos padrões ANSI/NIST-ITL e EFTS, para classificação das imagens e para minúcias.

• INT-I (*Interpol Implementation*): é um documento que tem como objetivo complementar a publicação ANSI/NIST-ITL orientando as organizações policiais internacionais que fazem parte da Interpol (INTERPOL, 2010). Esse padrão aborda a interoperabilidade entre sistemas diferentes e a troca de informações, da área policial, de impressões palmares (impressões da palma da mão) e plantares (impressões dos pés), bem como, impressões digitais. O INT-I foi elaborado pelo órgão denominado IAEG (*Interpol AFIS Experts Working Group*). O IAEG é formado por representantes de oito países, dentre estes o Brasil. O seu objetivo é prover meios para que países e organizações possam adquirir, desenvolver e integrar sistemas AFIS.

Tabela 2.1. Codificação ANSI/NIST e EFTS para Classificação de Padrões de Impressões Digitais (KAZIENKO, 2003)

Padrões de Classificação	ANSI/NIST	EFTS
Arco (tipo não designado)	-	AU
Arco plano	PA	-
Arco tendido	TA	-
Presilha radial	RL	-
Presilha cubital	UL	-
Verticilo plano	PW	-
Presilha centralizada	CP	-
Presilha dupla	DL	-
Verticilo acidental	AW	-
Presilha inclinada à direita	RS	RS
Presilha inclinada à esquerda	LS	LS
Verticilo (tipo não designado)	WN	WU
Cicatriz	SR	SR
Amputação	XX	XX
Desconhecido ou não classificável	UN	UC

2.5.3 NFIS - NIST Fingerprint Image Software

O NFIS é um AFIS, de distribuição gratuita controlada mediante solicitação ao NIST (Instituto Nacional de Padrões e Tecnologias dos Estados Unidos). Foi desenvolvido pelo NIST em parceria com o FBI. Seu código fonte foi escrito na linguagem de programação C e o programa pode ser compilado tanto na plataforma Linux quanto na Win32. Este software encontra-se na sua segunda versão e é composto por sete pacotes responsáveis por (WATSON et al., 2004): manipular imagens (IMGTOOLS e AN2K), segmentar as imagens da ID (NFSEG), medir a qualidade da imagem da ID (NFIQ), identificar a classe a qual a imagem da ID pertence (PCASYS), detectar as minúcias (MINDTCT) e comparar as minúcias (BOZORTH3). Os três últimos foram utilizados nesse projeto e serão detalhados.

PCASYS

Algoritmo que identifica a classe (com base na classificação de Henry (Seção 2.3)) a qual a imagem da impressão digital pertence. As classes identificadas pelo PCASYS são: arco, presilha interna, presilha externa, verticilo e cicatriz. Para identificar a classe o algoritmo segue os seguintes passos:

1) Segmentação - consiste na separação da região da imagem da impressão digi-

Tabela 2.2. Codificação ANSI/NIST e EFTS para Tipos de Minúcias (KAZIENKO, 2003)

Tipo de Minúcia	ANSI/NIST	EFTS
Final de Estria	A	A
Bifurcação de Estria	B	B
Trifurcação (ou cruzamento)	C	-
Tipo Indeterminado	D	C

tal removendo informações desnecessárias, como letras, por exemplo. Este processo é necessário para imagens adquiridas pelo método *offline*;

2) Realce da imagem - realça cristas e vales da imagem segmentada através do uso da transformada de Fourier;

3) Detecção da localização de cristas e vales - utiliza um algoritmo que faz a binarização (conversão da imagem para monocromática) onde as cristas são representadas pela cor preta e os vales pela cor branca;

4) Execução de dois algoritmos - um algoritmo utiliza redes neurais probabilisticas e de multi-camada para identificar a classe, e o outro que identifica se a impressão digital é da classe verticilo ou não. O resultado final é adquirido através da comparação dos resultados desses algorimos. Se o algoritmo responsável por detectar se a impressão digital é verticilo ou não, definir que a mesma é um verticilo, a impressão digital é, então, classificada como verticilo, desconsiderando a classificação pelo outro algoritmo. Caso contrário, vale o resultado do algoritmo de rede neural.

Para que o algoritmo possa efetuar, de maneira precisa, a classificação é necesário que a imagem da impressão digital; (i) esteja na escala de cinza com 8 bits por quadro; (ii) possua largura de no mínimo 512 pixels e altura de no mínimo 480 pixels (essas dimensões podem ser alteradas, para mais, no arquivo pca.h), e (iii) seja escaneada em 500 pixels de profundidade. Esse pacote disponibiliza, também, 2700 imagens de impressão digital que podem ser usadas para testes. A execução do algoritmo pode ser feita pelo modo gráfico, onde são apresentados todos os estágios do processamento da classificação, ou não. A saída é um arquivo com a classe a qual a impressão digital pertence.

MINDTCT

Pode ser definido como o algoritmo mais importante do NFIS, pois é responsável pela detecção das minúcias, que como foi visto na seção 2.2.1 são pontos característicos que definem a unicidade da impressão digital. Esses pontos são utilizados no processo de comparação de impressões digitais. O MINDTCT ao encontrar as minúcias, identifica

sua localização, orientação, tipo e qualidade. Essas informações são armazenadas em um arquivo que é utilizado, posteriormente, no processo de comparação. Os tipos de minúcias detectadas pelo MINDTCT são final de linha e bifurcação. Para a comparação, são armazenadas as coordenadas de localização e a orientação de cada minúcia.

A orientação das minúcias está representada em graus, onde zero representa o ponto direito horizontal e o aumento do ângulo é feito no sentido horário. A orientação de final de linha é determinada pela medição do ângulo entre o eixo horizontal e a linha que inicia no ponto da minúcia e segue em direção ao meio da crista. A orientação da bifurcação é determinada pela medição do ângulo entre o eixo horizontal e a linha que inicia no ponto da minúcia e segue em direção ao meio do vale entre as linhas da bifurcação, conforme demonstrado na figura 2.9.

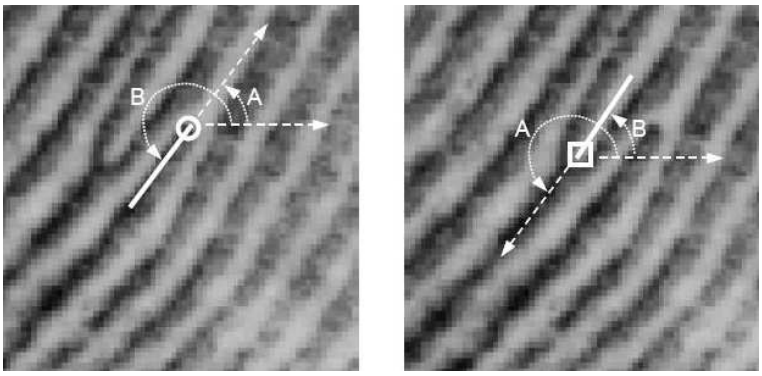

Figura 2.9. Orientação das Minúcias. O "A"representa o ângulo padrão do ANSI/NIST e o "B"representa o ângulo do FBI/IAFIS (WATSON et al., 2004)

Para imagens de boa qualidade são detectadas, em média, cem minúcias em cada dedo das mãos (WATSON et al., 2004). Esse número reduz bastante quando se trata de impressões digitais latentes, onde normalmente captura-se apenas parte do dedo. A tecnologia AFIS funciona de maneira precária para esse tipo de impressão, pois a quantidade de minúcias encontradas é bastante reduzida (menos de doze). Os seguintes passos são executados, através de uma única rotina, para para detecção das minúcias:

1) Arquivo de Entrada da Impressão Digital - as imagens de entrada devem estar escaneadas em 500 dpi e com 256 níveis de cinza. Os formatos de arquivos que são identificados pelo MINDTCT são: ANSI/NIST, WSQ, JPEG *Baseline*, JPEG *Lossless* e IHEAD. O MINDTCT dispõe, também, da opção de realce das imagens de baixo contraste. Quando essa opção é utilizada, é verificado se a imagem tem baixo contraste. Se positivo a imagem é realçada, caso contrário, nenhuma modificação é feita.

2) Geração dos Mapas de Imagem - como a qualidade da imagem da impressão digital pode variar em função do modo em que esta foi capturada, especialmente no caso de impressões latentes, torna-se crítica a identificação de regiões degradadas e que não devem ser analisadas. Para identificar a qualidade das áreas da impressão digital, deve-se

verificar a determinação do fluxo direcional das cristas e identificar as regiões de baixo contraste - com pouco fluxo de cristas e com grandes curvaturas.

Para determinar o fluxo do direcionamento das cristas, a imagem é divida em blocos de 8x8 pixels onde é aplicada a cada bloco a transformada de Fourier para determinar a orientação da sua crista. Para analisar a variação da qualidade da imagem são atribuidos valores. A faixa de valores correspondem de 0 a 4, onde 0 representa o mais baixo nível de qualidade e 4 representa a região de melhor qualidade. Maiores detalhes sobre esses procedimentos podem ser adquiridos em (WATSON et al., 2004).

3) Binarização da Imagem - o algoritmo de detecção de minúcias foi desenvolvido para trabalhar com imagens binárias, onde os pixels pretos representam as cristas e os pixels brancos representam os vales. Para criar a imagem binária, cada pixel da imagem de entrada (em escala de cinza) deve ser analisado para determinar o valor a ser atribuído (preto ou branco). Esse processo é denominado binarização e pode ser observado na Figura 2.10. Uma técnica utilizada para binarizar imagens é a utilização de um *threshold*. *Threshold* é uma palavra, da lingua inglesa, que traduzida significa limiar (limite). Para binarizar uma imagem utilizando *threshold* deve-se determinar um valor de limite para que aos valores abaixo ou nesse limite sejam atribuídos - o valor 0 (preto) e acima desse limite o valor 1 (branco).

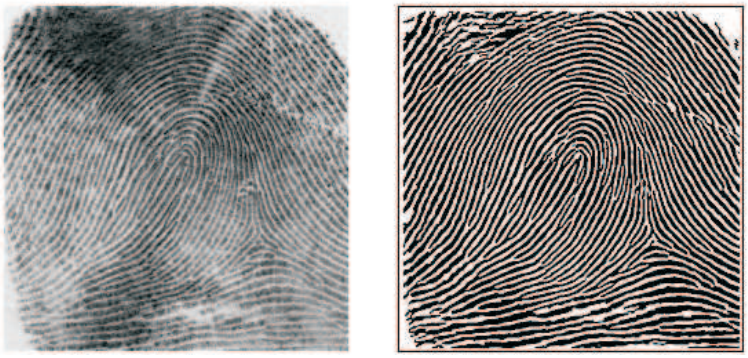

Figura 2.10. Impressão Digital antes e depois da Binarização (WATSON et al., 2004)

4) Detecção da Minúcias - esse passo consiste em percorrer a imagem binária, da impressão digital, identificando a localização de *pixels* através de padrões que definem final de linha ou bifurcação. A varredura da imagem pode ser feita tanto verticalmente quanto horizontalmente, para isto, basta rotacionar os pares de *pixels* em 90º em sentido horário e passar a sequência da esquerda para a direita.

Existem dois padrões para detecção de final de linha e oito padrões para detecção de bifurcação. Esses padrões podem ser observados na figura 2.11. O "*"apresentado na figura indica que o *pixel* identificado pode aparecer uma ou mais vezes. Os atributos *appeearing*(aparecendo) e *disappearing*(desaparecendo), também apresentado na figura,

indicam a direção do surgimento de cada crista ou vale no padrão.

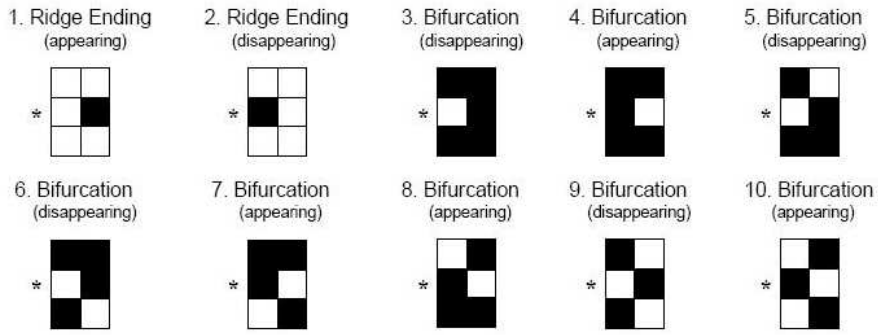

Figura 2.11. Padrões usados para Detecção de Minúcias (WATSON et al., 2004)

5) Remoção das Falsas Minúcias - a detecção de minúcias através do passo 4) não descarta a possibilidade da detecção de falsas minúcias, que podem ser detectadas em imagens de baixa qualidade ou com muitos borrões. Alguns exemplos de falsas minúcias incluem: ilhas, lagos, furo, entre outras. Para removê-las são aplicadas rotinas específicas. Detalhes sobre essas rotinas podem ser encontradas em (WATSON et al., 2004).

6) Contagem de Vizinhos das Cristas - esse processo é utilizado para guardar informações das regiões vizinhas às cristas. Tais como, posições de minúcias, sua direção, tipo, entre outros. Essas informações podem ser utilizadas no processo de comparação.

7) Avaliação da Qualidade das Minúcias - embora exista o processo de remoção de falsas minúcias, ainda assim, podem persistir minúcias que não são verdadeiras. Para detectar essas minúcias falsas é medida a intensidade da qualidade da região da minúcia. Para medir a qualidade das minúcias dois fatores são combinados. O primeiro é baseado na medida da qualidade apresentado no item 2 representada por cinco níveis sendo 0 o pior e 4 o melhor. O segundo fator é baseado na média e desvio padrão dos pixels vizinhos à minúcia, num raio de 11 pixels (tamanho considerado suficientemente grande para conter uma boa quantidade de cristas e vales).

8) Geração do Arquivo de Saída das Minúcias - após executar os passos de 1 a 7, o MINDTCT gera arquivos de saída. O formato dos arquivos de saída depende da imagem de entrada e se baseia se esta estava no formato ANSI/NIST ou não. Os arquivos de saída são arquivos texto que usam delimitadores para separar as informações das minúcias, como suas coordenadas (x,y), ângulo de direção e qualidade da minúcia. Essas informações serão utilizadas pelo algoritmo BOZORTH3 no processo de comparação.

Alguns outros arquivos também são gerados. Estes incluem um arquivo para cada mapa da qualidade apresentado no item 2) e um arquivo onde são listadas todas as minúcias detectadas e seus atributos associados.

BOZORTH3

Algoritmo de comparação de impressões digitais. Seus dados de entrada encontram-se no arquivo com extensão xyt gerado pelo MINDTCT. Para comparação, o BOZORTH3 usa apenas a localização e a orientação das minúcias detectadas na impressão digital. A identificação pode ser feita de um-para-um ou de um-para-muitos, definidos através de comandos específicos. Os seguintes passos são executados para comparação das impressões digitais:

1) Construção das tabelas de comparação intra-minúcias (uma para cada impressão) - o primeiro passo do algoritmo é calcular as medidas relativas de cada minúcia da impressão digital em relação a todas as outras minúcias do mesmo dedo. Essas medidas são, então, guardadas numa tabela denominada de tabela de comparação de minúcias. Essas medidas referem-se à distância entre as minúcias e às medidas dos ângulos entre a linha que liga duas minúcias e a orientação das minúcias. A Figura 2.12 apresenta o cálculo dessas distâncias.

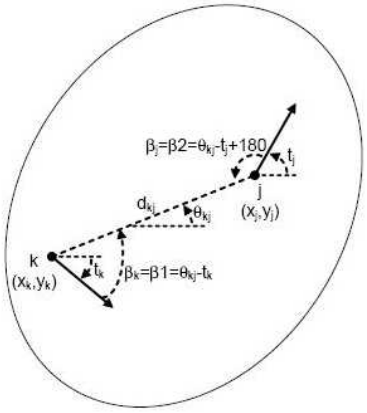

Figura 2.12. Distância intra-minúcias (WATSON et al., 2004)

2) Construção da tabela de compatibilidade das impressões digitais - faz a comparação entre as tabelas do passo anterior a procura por pontos (localizações e orientações) compatíveis. Encontrando-os, estes são armazenados em uma tabela de compatibilidade. A compatibilidade é medida através dos seguintes testes:

- Verificar se as distâncias correspondentes entre as duas tabelas estão dentro de uma determinada tolerância T_d.

- Verificar se os ângulos formados entre as orientações das minúcias e a linha que as liga está dentro de uma tolerância T_β. Isto é feito através de duas funções diferentes.

Se a distância relativa e os ângulos das minúcias estiverem dentro das tolerâncias aceitáveis será gravada a relação entre os dois pares de minúcias numa tabela denominada de tabela de compatibilidade.

3) Fazer o cruzamento das informações das tabelas de compatibilidade - até esse ponto, foi construída a tabela de compatibilidade que consiste em uma lista de ligações (*links*) separados entre duas minúcias que forma um grafo de compatibilidade. Para determinar o quanto duas impressões digitais são iguais, deve-se percorrer o grafo achando o maior caminho de associações de minúcias. A pontuação será o tamanho do caminho mais longo do grafo.

Idealmente, a pontuação deve ser alta se duas impressões digitais forem do mesmo dedo e baixa, caso contrário. Está definido no algoritmo, como regra geral, que uma pontuação com valor superior a 40 indica um bom nível de segurança para garantir que as impressões digitais pertencem ao mesmo dedo. Porém, esse valor pode ser alterado para mais ou para menos em decorrência do tipo de aplicação a ser usada, aferindo diretamente às taxas FAR e FRR, citadas na Seção 2.5.1.

CAPÍTULO 3

GRADE COMPUTACIONAL

O termo grade computacional, também conhecido como *grid computing* ou grade de computadores, surgiu em meados dos anos 90. O objetivo era reproduzir a idéia de um metacomputador que correspondia a um ambiente dinâmico composto por nós processadores podendo ser adicionados ou removidos livremente, e sendo vistos como um único recurso computacional capaz de executar aplicações paralelas em recursos geograficamente dispersos e pertecentes a diversas organizações. Com o decorrer do tempo, essa definição foi estendida para englobar não apenas a integração de processadores, mas também, outros recursos como, fontes de informações e dados, instrumentos científicos, componentes de software, pessoas, entre outros (BARRETO; NAVAUX, 2003).

A promessa era fornecer uma plataforma, bem mais barata, para execução de aplicações distribuídas (conectadas através da Internet), em alternativa aos supercomputadores paralelos, possibilitando, através da junção de recursos dispersos, execuções paralelas numa proporção que seria impossível com a utilização de um único supercomputador (CIRNE; SANTOS, 2005). A sua utilização começou em universidades e centros de pesquisas, porém, atualmente já alcançou o mundo empresarial, envolvendo diversas empresas como IBM, HP, Sun, Microsoft, Oracle, entre outras (CIRNE; SANTOS, 2005).

O conceito inicial de grade computacional proposta por Foster (FOSTER, 2002) faz analogia à rede elétrica. Ou seja, serviços computacionais (ciclos, armazenamento, software, periféricos, entre outros) são disponibilizados sem que o usuário tenha conhecimento de onde estão vindo, precisando apenas estar ligado à grade. Para Foster (FOSTER, 2002) uma grade computacional:

• Coordena recursos que não possuem um controle centralizado – uma grade computacional integra e coordena recursos e usuários que pertencem a diferentes domínios e coordena questões como segurança, grupo de usuários, política de acesso, entre outros, que fazem parte desse ambiente.

• Usa protocolos e interfaces padrão, aberta e de propósito geral – uma grade computacional é constituída para múltiplos propósitos que exigem algumas condições fundamentais como autenticação, autorização, descobrimento de recursos e acesso aos recursos.

• Disponibiliza qualidade de serviço não triviais – uma grade computacional permite que seus recursos constituintes sejam usados de maneira coordenada para garantir qualidade de serviço relacionadas, por exemplo, ao tempo de resposta, taxa de transferência (*throughput*), disponibilidade e segurança.

Seu principal objetivo é compartilhar e agregar recursos e disponibilizá-los como serviços. Como exemplos desses serviços podemos citar: alocação de recursos, gerencia-

mento de processos, comunicação, autenticação e segurança. Os recursos que compõem uma grade computacional são interligados por redes WAN (*Wide Area Network*) de alta velocidade (via cabo ou sem fio (*wireless*)), sendo assim, considerado um ambiente de alto desempenho. As grades computacionais podem ser consideradas como plataformas de computação distribuída. O que as difere das demais plataformas tradicionais (Corba, DCOM, clusters, entre outras) são as seguintes características que lhes são peculiares (CIRNE; SANTOS, 2005):

• Heterogeneidade – os componentes que constituem a infraestrutura normalmente são bastante heterogêneos. Ou seja, a grade computacional permite a comunicação entre componentes (tanto hardware quanto software) de diferentes fabricantes e versões.

• Alta Dispersão Geográfica – as grades computacionais podem atingir escalas globais, reunindo serviços que podem estar localizados em diversas partes do mundo.

• Compartilhamento – uma grade computacional não pode ficar dedicada a uma aplicação exclusivamente por um período de tempo determinado, pois isso pode causar impacto no desenvolvimento de aplicações que executam sobre a infraestrutura da grade.

• Múltiplos Domínios Administrativos – várias instituições diferentes podem compor a grade computacional. Sendo assim, diferentes políticas de acesso e uso dos serviços, podem coexistir, sendo definidas por cada domínio que faça parte da grade.

• Controle Distribuído – não existe uma única entidade que tenha poder sobre toda a grade computacional. Cada instituição pode implementar sua política em seus recursos locais, mas isto não interfere nas políticas determinadas por outras instituições que também compõem a grade.

Na literatura são apresentados dois modelos de arquitetura de grade computacional que são bem aceitos na comunidade científica e são apresentados em (COLVERO; DANTAS; CUNHA, 2005). A Figura 3.1 apresenta os elementos que constituem essas arquiteturas e que serão explicados abaixo.

A primeira arquitetura apresentada na Figura 3.1(A)compreende os seguintes elementos (COLVERO; DANTAS; CUNHA, 2005):

• Aplicação e Serviços – composta por pacotes de software de aplicação, incluindo os conjuntos de ferramentas de desenvolvimento e serviços.

• *Middleware* – camada responsável por fornecer protocolos que permitem que múltiplos elementos heterogêneos possam participar de um ambiente de grade computacional unificado. Como exemplo de heterogeneidade podemos citar: diferentes sistemas operacionais, sistemas de arquivos e protocolos de comunicação entre redes.

• Recursos – camada constituída por um conjunto de recursos que fazem parte da grade computacional, incluindo servidores primários e dispositivos de armazenamento.

• Rede – camada que compõe a base da conectividade para os recursos da grade. Nesta camada estão os *switches*, roteadores e toda infraestrutura das redes de comunicação.

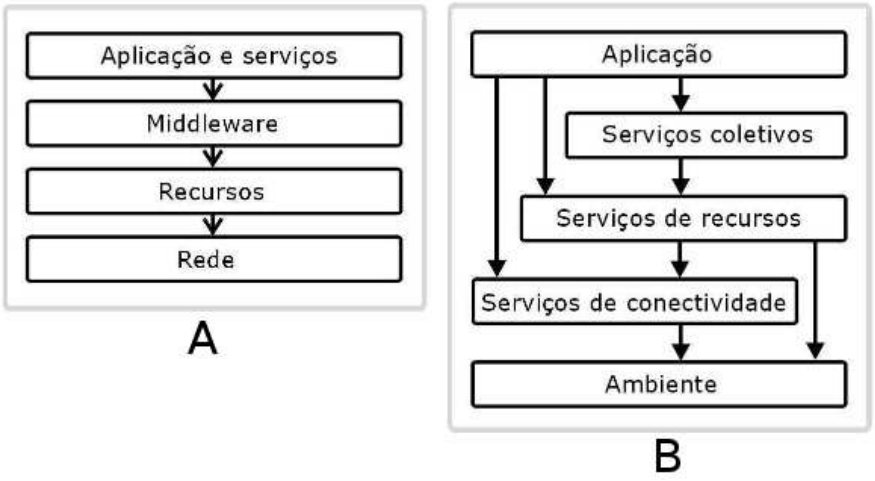

Figura 3.1. Modelos de Arquitetura de Grade Computacional (COLVERO; DANTAS; CUNHA, 2005)

A segunda arquitetura, apresentada na Figura 3.1(B), é formada pelas seguintes camadas (COLVERO; DANTAS; CUNHA, 2005):

• Aplicação – compreende as aplicações do usuário que operam dentro do ambiente da grade computacional, incluindo bibliotecas de funções que utilizam os serviços prestados pelas demais camadas.

• Serviços Coletivos – coordena múltiplos recursos. Fornece protocolos e serviços que não estão associados a um recurso específico e sim a coleções deste. Os serviços englobam: serviços de diretório, monitoração e diagnóstico, replicação de dados, gerenciamento de carga de trabalho e descoberta de recursos, autorização, verificação e colaboração.

• Serviços de Recursos – corresponde à definição dos protocolos e API's que fornecem segurança na negociação, iniciação, monitoramento, controle e contagem dos recursos compartilhados. Utiliza as funcionalidades da camada de serviços de conectividade para negociação segura, monitoramento, controle e contabilização de operações de compartilhamento em recursos individuais. Utiliza, também, as funcionalidades da camada ambiente para acessar e controlar dispositivos locais.

• Serviços de Conectividade – define os protocolos de comunicação e segurança necessários para transações de rede específicas de uma grade computacional, permitindo troca de dados entre os níveis de ambiente e recursos. Fornecem, também, protocolos de autenticação que permitem verificar a identidade de usuários e recursos através da utilização de métodos de autenticação e criptografia.

• Ambiente – esta camada fornece funcionalidades com as quais o compartilhamento de recursos pela grade torna-se possível. Implementa operações locais e específicas para

cada tipo de recurso compartilhado pela grade, fornecendo suporte às funções das camadas superiores.

As grades computacionais são formadas por organizações, denominadas organização virtual (OV) que podem se constituir de universidades, empresas, centros de pesquisa ou, até mesmo, pessoas. As OV's permitem que diferentes grupos de organizações e/ou indivíduos compartilhem recursos de maneira controlada, onde os membros podem colaborar para atingir um objetivo em comum (BARRETO; NAVAUX, 2003). As OV's detêm o controle sobre o compartilhamento dos dados e dos recursos e as condições em que estes poderão ocorrer. Cada organização virtual possui características próprias e necessidades computacionais específicas que variam em função do seu tamanho e objetivo a ser alcançado. As demandas computacionais de dados e/ou processamento que são geradas pelas organizações virtuais são coordenadas pela grade que proporciona segurança (através de controle de acesso não autorizado) e qualidade de serviço na execução de tarefas.

Os serviços providos na OV são agrupados em um sistema integrado, denominado *middleware*. Os *middlewares* foram criados com objetivo de facilitar a alocação e escalonamento dos recursos distribuídos, tornando transparente, para os usuários, questões relacionadas a localização e paralelismo. Através de técnicas de autenticação e autorização (controle de acesso), fica garantido um ambiente seguro. Existem diversos tipos de *middlewares* de grade distribuídos pelo mundo. Dentre os mais conhecidos, pode-se destacar o Globus (GT4, 2007b), Legion (LEGION, 2009), gLite - do projeto Europeu EGEE (GLITE, 2009) e Ourgrid - da UFCG (OURGRID, 2009). Na Seção 3.4 será detalhado o *middleware* Globus Toolkit 4, este foi o *middleware* utilizado neste projeto.

Os sistemas de grade computacional estão divididos em três grandes áreas que se diferem na atividade principal a qual se destinam (SANTOS, 2006):

• Grade Computacional ou de Grade de Processamento (*Computacional Grid*) – corresponde aos sistemas que exploram a capacidade computacional dos recursos distribuídos para propocionar maior capacidade de processamento. Podem ser classificadas como uma extensão dos sistemas distribuídos. Seu objetivo é fornecer computação de alto desempenho.

• Grade de Serviços – abrange os sistemas que provêm serviços executados de forma distribuída que dificilmente seriam disponibilizados por um único recurso. Esses serviços estão subdivididos em: por demanda, colaborativos e multimídia.

• Grade de Dados (*data grid*) – seu foco é no acesso, pesquisa, manipulação e gerência de grandes volumes de dados distribuídos. Uma grade de dados proporciona uma infraestrutura robusta para armazenamento, transferência confiável e alta disponibilidade dos dados distribuídos em diferentes locais, mas que para o usuário final é como se estivessem armazenados em um único local. As grades de dados serão detalhadas na Seção 3.3, por ser o foco principal desse trabalho.

3.1 PADRONIZAÇÃO

Quando a computação em grade surgiu, não existiam padrões para o seu desenvolvimento. Porém, as possibilidades oferecidas por esta tecnologia, tais como, interoperabilidade, diversidade de recursos, dispersão geográfica, entre outros, fez surgir a necessidade de uma padronização. A organização responsável por essa padronização é a OGF - *Open Grid Forum* (OGF, 2008). A OGF foi criada por membros organizacionais, incluindo empresas e academias da área tecnológica e instituições de pesquisas governamentais (OGF, 2008). A OGF é uma organização que resultou da fusão, em 2006, do *Global Grid Forum* (GGF) - comunidade formada por entidades do meio científico e corporativo para criação e padronização de tecnologias para ambientes de grade, e o *Enterprise Grid Alliance* (EGA) - consórcio formado para desenvolver soluções de grade computacional para o mercado corporativo e agilizar a sua adoção. Entre os membros do GGF podemos citar: IBM, Oracle, Intel, Nasa, entre outros.

A principal função do OGF consiste em definir políticas, padrões e boas práticas relacionadas à criação de uma grade computacional, além de permitir, através de fórum, trocas de experiências, conhecimentos e idéias relacionadas à grade computacional. Ela produz o GGF *Document Series* que é uma série de documentos que definem os padrões de funcionamento de uma grade computacional. Tais como, autenticação, transmissão de dados, comunicação, entre outros. Organiza, também, vários eventos, a cada ano, com o intuito de desenvolver especificações e casos de uso relacionadas a grade computacional e,também, compartilhar as melhores práticas.

O modelo OGF foi criado no padrão aberto IETF - *Internet Engineering Task Force* (IETF, 2009), que é uma comunidade internacional, composta por técnicos, agências, fabricantes, fornecedores e pesquisadores, preocupados com a evolução da arquitetura da Internet e seu perfeito funcionamento. Os padrões OGSA (*Open Grid Service Architecture*) e OGSI (*Open Grid Service Infrastructure*) foram criados por este orgão. De acordo com o OGSA, todos os recursos compartilhados na grade são denominados de serviços de grade, que são serviços Web com interfaces para executar tarefas como, descoberta e criação dinâmica de serviços, serviços de notificação, entre outros. Já o OGSI detalha a arquitetura definida pelo OGSA, determinando como o serviço de grade (*Grid Services*) deve se comportar. Para isso, define extensões para o padrão WSDL para que detalhes específicos de um Grid Service possam ser descritos (TEIXEIRA, 2006). A Seção 3.1.1, faz uma abordagem da tecnologia *web services* que serviu como base para criação do padrão OGSA, detalhado na Seção 3.1.2.

3.1.1 Web Services

Web Service ou Serviço Web, em português, surgiu da necessidade de comunicação entre os mais diversos ambientes computacionais, independente de hardware, tecnologia ou localização, em alternativa às soluções já existentes, como Corba, EJB e Java RMI, utilizando a Internet como meio transmissor.

Em (SOTOMAYOR, 2005) são apresentadas as seguintes vantagens do uso de *Web Services* em relação as demais tecnologias:

• São independentes de plataforma e linguagens de programação, desde que crie e aceite mensagens definidas para interface do *Web Service*. Utilizam a linguagem padrão XML para troca de mensagens. Desta maneira, é possível a troca de mensagens entre aplicações desenvolvidas em linguagens de programação diferentes sendo executadas em sistemas operacionais distintos.

• O HTTP é o meio de transmissão de mensagens mais utilizado pelos *Web Services*. Isto é importante para construção de aplicações escaláveis, pois o uso de HTTP permite que os *Web Services* ultrapassem as barreiras dos *proxies* e *firewalls*, pois o tráfego é constituído apenas de dados baseados em XML.

Como desvantagens do uso de *Web Services* (SOTOMAYOR, 2005) destaca:

• Sobrecarga (*Overhead*) – a transmissão de dados em XML pode gerar uma sobrecarga, que é aceitável para alguns tipos de aplicação, porém para aplicações de tempo real críticas, o uso de *Web Services* pode não ser uma boa solução.

• Falta de Versatilidade – *Web Service* não é muito versátil, já que disponibiliza apenas algumas formas básicas de invocação de serviço. Serviços como persistência, notificações, administração da durabilidade, transações, entre outros, não são suportadas. Para tornar o *Web Service* mais versátil, algumas especificações estão sendo criadas, como é o caso do WSRF, explicado na Seção 3.1.2.

Arquitetura de um *Web Service*

Os *Web Services* se baseiam na arquitetura SOA (*Service Oriented Architecture*). Esta arquitetura define que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços. Os seguintes componentes fazem parte da arquitetura SOA: consumidor do serviço (*service consumer*), provedor de serviço (*service provider*) e registro de serviços (*service registry*) (MARQUEZAN; CARISSIMI; NAVAUX, 2006), conforme mostra a Figura 3.2.

O provedor de serviço é o componente que cria o *Web Service* e o disponibiliza para ser utilizado. Para que isso seja possível, é necessário que o serviço esteja descrito em um formato padrão, que seja entendido por qualquer um que queira utilizá-lo. Além disso, o provedor de serviços deve descrever, de forma padronizada, cada serviço criado e a sua disponibilização em um servidor de registros centralizados e públicos. Sendo assim, qualquer instituição que queira utilizá-lo, o encontrará através de mecanismos de busca de *Web Service*.

O consumidor do serviço representa qualquer cliente que irá utilizar o *Web Service* criado e disponibilizado pelo provedor de serviços. O consumidor do serviço pode ser um aplicativo ou um outro *Web Service* que necessite de um serviço disponibilizado. Para que o consumidor seja capaz de ter acesso ao *Web Service* disponível é necessário efetuar

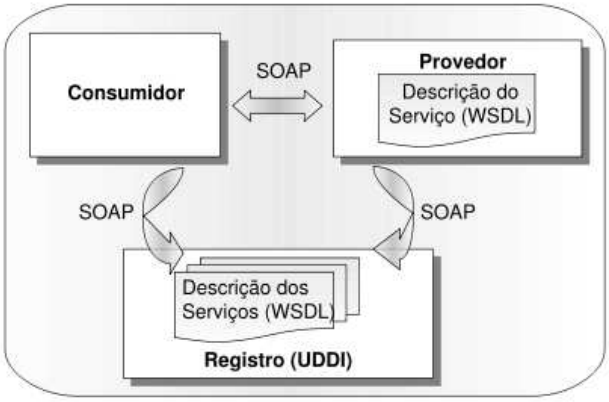

Figura 3.2. Arquitetura *Web Service* (MARQUEZAN; CARISSIMI; NAVAUX, 2006)

uma busca no registro de serviços.

O registro de serviços é o componente central onde tanto o provedor como o consumidor de serviços têm acesso. Os provedores para publicar seus serviços e os consumidores para encontrar os serviços, e assim, consumí-los. O registro de serviços é um repositório baseado na linguagem XML.

Tecnologias Padrão em *Web Services*

Em (BOOTH et al., 2004) um *Web Service* está definido, tecnicamente, como um sistema projetado para permitir comunicação máquina-a-máquina através da Internet. Para isto, utiliza uma interface descrita no formato WSDL, registrado via UDDI. A interação de outros sistemas com *Web Service* é feita através da troca de mensagens SOAP, que normalmente é transmitida via protocolo HTTP no formato XML.

As principais tecnologias padrão utilizadas para criação de *Web Services* serão descritas a seguir.

1. XML - *eXtensible Markup Language*

A XML é uma linguagem de marcação, ou seja, codificada textualmente, definida pela W3C (*World Wide Web Consortium*), para descrever a estruturação dos dados a serem trocados entre aplicativos através da internet. É similar a linguagem HTML, diferindo-se na finalidade a que se propõe. A linguagem HTML foi criada para definir a aparência de páginas Web e a XML foi criada para elaborar documentos estruturados para a Web (COULOURIS; DOLLIMORE; KINDBERG, 2007). Tanto a XML quanto a HTML são derivadas da linguagem SGML (*Standardized Generalized Markup Language*), que é uma metalinguagem através da qual se pode definir linguagens de marcação para documentos.

Os dados em XML são rotulados através do uso de *tags*. As *tags* são *strings* de

marcação usadas para descrever a estrutura lógica dos dados e associar valores aos atributos dessa estrutura lógica. A XML é uma linguagem extensível, pois os usuários podem determinar suas próprias *tags*, porém, os nomes das *tags* devem ser conhecidas por todas aplicações que irão utilizá-las. O uso de texto torna a XML independente de plataformas, além de ser entendível tanto por seres humanos quanto pelo computador. O seu ponto crítico é que a representação textual, no lugar da binária, associada ao uso de *tags*, torna as mensagens muito maiores, exigindo assim um tempo maior de processamento e transmissão, além de mais espaço de armazenamento. Esta tecnologia serve de base a todos os protocolos utilizados nos Web Services.

A Figura 3.3 mostra um exemplo da representação do XML. O exemplo descreve um empréstimo de dois livros de uma biblioteca.

```
<emprestimo>
 <livro id=”1231313”>
   <nome>Brás Cubas</nome>
   <autor>Machado de Assis</autor>
   <editora>Editora Tres</editora>
   <ano>1997</ano>
 </livro>
 <livro id=”2342322”>
   <nome>A origem das espécies</nome>
   <autor>Charles Darwin</autor>
   <editora>Editora Moderna</editora>
   <ano>1995</ano>
 </livro>
</emprestimo>
```

Figura 3.3. Exemplo de uma Representação XML (COULOURIS; DOLLIMORE; KINDBERG, 2007)

2. SOAP - *Simple Object Access Protocol*

Protocolo que define o formato de uma mensagem a ser trocada entre duas aplicações através do uso de uma variedade de protocolos como HTTP, SMTP, TCP e UDP. O conteúdo das mensagens tanto de requisição quanto de resposta são definidas através do uso de XML (COULOURIS; DOLLIMORE; KINDBERG, 2007). Mensagens SOAP são escritas em XML, sendo assim, independente de linguagem e plataforma.

A especificação do protocolo SOAP define (COULOURIS; DOLLIMORE; KINDBERG, 2007):

- Como a XML deve ser utilizada para representar o conteúdo das mensagens;
- Como duas mensagens devem ser definidas para originar um padrão de requisição

e resposta;

• Regras para que os destinatários das mensagens possam processar os elementos XML que estas contêm;

• Como os protocolos HTTP e SMTP devem ser usados para transmitir mensagens SOAP.

Uma mensagem SOAP é transportada dentro de um elemento denominado envelope que define regras específicas para encapsular os dados que serão transferidos entre as máquinas. Pode incluir informações sobre quem deve processar o conteúdo da mensagem, e na ocorrência de falha, pode indicar como codificar as mensagens de erro (TEIXEIRA, 2006). O envelope é composto por uma área denominada cabeçalho, de uso opcional, e uma área de conteúdo, denominado corpo da mensagem. O cabeçalho pode ser utilizado para registrar funções específicas da aplicação no transporte da mensagem, como, controle de transação, autenticação, entre outros. O corpo do envelope armazena a mensagem que será trocada entre as aplicações. As solicitações SOAP podem ser feitas através de três padrões (TEIXEIRA, 2006): (i) GET – para requisição de informações; (ii) POST – para transmitir uma mensagem SOAP; e, (iii) SOAP – semelhante ao POST, com solicitações feitas em XML, permite transportar estruturas e arrays.

As respostas às solicitações são sempre em XML, independente de qual método tenha sido usado para solicitar. Se ocorrer uma falha numa solicitação, a descrição do erro é transmitida no corpo da mensagem de resposta em um elemento denominado falha. Esse elemento além de conter informações sobre o erro, contém detalhes específicos da aplicação. O SOAP define também um padrão chamado WSDL, que descreve os objetos e métodos disponibilizados por uma aplicação, através de páginas XML acessíveis através da Web. Quem publica um serviço deve criar também o WSDL. Para utilizar o serviço deve-se acessar o WSDL para obter informações sobre o mesmo.

3. WSDL - *Web Services Description Language*

Documento escrito em XML para descrever um serviço Web e especificar como acessá-lo, além de apresentar as operações e métodos disponibilizados por determinada aplicação.

A função do WSDL é descrever as interfaces apresentadas e disponibilizar a localização dos seus serviços. É composto dos seguintes elementos (TEIXEIRA, 2006):

• *types* – definição de tipos usados pelo serviço;

• *message* – contém descrição do conjunto de mensagens trocadas (parâmetros de entrada e saída de um serviço);

• *interface* – as definições de interface são necessárias para permitir que os clientes se comuniquem com os serviços;

• *bindings* – contém a descrição do protocolo de rede para invocação;

• *services* – referência para atual localização do serviço.

4. UDDI - *Universal Description Discovery and Integration*

O UDDI é uma especificação técnica cujo objetivo é definir a descrição, descoberta e integração de *Web Services* (MARQUEZAN; CARISSIMI; NAVAUX, 2006). Funciona como um mediador do serviço, permitindo que os clientes requisitantes encontrem um fornecedor do serviço apropriado. Essas informações são disponibilizadas através de um diretório composto por um arquivo XML que descreve negócios e serviços.

O UDDI é, normalmente, comparado a uma lista telefônica composta por (MARQUEZAN; CARISSIMI; NAVAUX, 2006):

• Páginas Brancas (*White Pages*) – contém informações sobre nomes e endereços de negócios, informações para contato, além de outras informações sobre os provedores de serviço;

• Páginas Amarelas (*Yellow Pages*) – contém informações relacionadas a tipos de negócios, organizada por categoria específica e localização geográfica;

• Páginas Verdes (*Green Pages*) – contém informações técnicas sobre o *Web Service*. Essas informações normalmente incluem um ponteiro para uma especificação externa e um endereço para invocar o serviço.

Desta maneira, pode-se concluir que o UDDI é uma interface Web para definir serviços que disponibilizam a descrição e descoberta de negócios, organizações e outros provedores de serviço, permitindo acesso e gerenciamento destes serviços.

3.1.2 OGSA

OGSA acrônimo de *Open Grid Services Architecture*, em português, Arquitetura Aberta de Serviços de Grade, foi construída nos conceitos e tecnologias de grade computacional e *Web Services*. O termo "arquitetura"denota um conjunto bem definido de interfaces básicas da qual pode-se construir sistemas, e o termo "aberta"é usado para demonstrar que a comunicação é extensível, independente de vendedor e comprometido com o processo de padronização (FOSTER et al., 2002). Essa arquitetura define uma semântica de serviços uniformes denominados de serviços de grade (*Grid Services*), além de mecanismos padrão para criação, especificação e descobrimento de instâncias dos serviços de grade, provendo transparência na localização.

A premissa básica da OGSA é que tudo está representado através de um serviço. Recursos computacionais, recursos de armazenamento, rede, programas, banco de dados, entre outros, são considerados como serviços. Mais especificamente, OGSA representa tudo como um serviço de grade, um *Web Service* que obdece a um conjunto de convenções e suporta interfaces padrão para determinados propósitos. Define as semânticas de uma instância do serviço de grade: como este é criado, como é nomeado, como o seu tempo de vida (duração) será determinado, como se comunicar com ele, entre outras. Não determina questões relativas ao modelo de implementação da programação, linguagem de programação, ferramentas de implementação ou ambiente de execução.

Um serviço de grade implementa uma ou mais interfaces comuns para tipos de serviços similares, escondendo, assim, diferenças das suas propriedades e operações, permitindo que estas sejam vistas e/ou manipuladas de maneira comum (BERRY; LUNIEWSKI; ANTONIOLETTI, 2007). Cada interface define um conjunto de operações que são invocadas através da troca de uma sequência definida de mensagens. Os serviços podem ser criados e destruídos dinamicamente, podem ser destruídos explicitamente ou porque se tornou inacessível devido a ocorrência de alguma falha de parada (*crash*) ou de partição de rede.

As seguintes vantagens oferecidas por *Web Services* estimulou a construção da arquitetura OGSA baseada nessa tecnologia (SENGER; STANZANI; RONDINI, 2006):

• Independência de Plataforma – podem ser utilizadas diferentes linguagens de programação tanto para as aplicações cliente quanto para as implementações dos serviços, além de poderem ser executadas em diferentes sistemas operacionais. Isto é possível, através do uso da linguagem XML e da invocação dos serviços utilizando a Internet através de mensagens padronizadas pelo protocolo SOAP.

• Acoplamento Dinâmico – através do acesso à sua interface WSDL, as aplicações cliente podem saber dinamicamente quais operações são suportadas por um determinado serviço. Se surgir uma nova implementação de serviço, sendo necessário alterar sua interface, as aplicações cliente perceberão as modificações de maneira automática, sem que seja necessário, por exemplo, a recompilação do mesmo.

• Ampla Disponibilidade de Ferramentas – os serviços podem ser implementados utilizando diversas tecnologias, como por exemplo, J2EE, .Net, entre outros.

• Facilidade de Transporte na Internet – as mensagens são transportadas por protocolos padrão, como o HTTP, por exemplo.

Como desvantagens, da utilização da tecnologia de serviços Web, pode-se destacar a sobrecarga gerada pelo processamento de documentos XML, e as implementações de segurança, notificação e transações que ainda não são bastante eficientes.

Os principais componentes da arquitetura OGSA são apresentados na Figura 3.4 e detalhadas a seguir (SENGER; STANZANI; RONDINI, 2006):

• Aplicações: nesta camada encontram-se as aplicações que executam na grade computacional.

• Serviços OGSA: define serviços, fundamentais para criação de uma grade computacional, que implementam funcionalidades que envolvem o gerenciamento de tarefas, segurança, serviços de informação e gerenciamento de recursos. A maioria desses serviços utiliza o suporte fornecido pela camada WSRF.

• WSRF (*Web Service Resource Framework*): define uma camada de serviços web adaptada às condições e ao ambiente de grades computacionais. Estes serviços podem ter estado associado a ele (adaptação feita devido à necessidade de guardar estado requerida pelo ambiente de grade e não implementada pelos *web services*). O padrão WSRF possui

Figura 3.4. Arquitetura OGSA (SENGER; STANZANI; RONDINI, 2006)

uma variedade de especificações independentes, porém relacionadas entre si de alguma forma (SENGER; STANZANI; RONDINI, 2006):

- WS-ResourceProperties: propriedades associadas aos recursos como, nome do arquivo, tamanho, entre outros. Define a maneira como as propriedades de um recurso podem ser definidas e acessadas;
- WS-ResourceLifetime: define ciclos de vida dos recursos, ao longo da execução da aplicação, onde os recursos podem ser criados e destruídos a qualquer momento;
- WS-ServiceGroup: define a maneira como os recursos podem ser agrupados e tratados de forma coletiva ou individual;
- WS-BaseFaults: define a maneira como detectar as falhas ocorridas, através de um tipo básico de XML de falha para ser utilizado como sinalizador da ocorrência de falhas na troca de mensagens entre os serviços.

Existem duas especificações que apesar de não fazerem parte do WSRF estão bastante relacionadas (SOTOMAYOR, 2005):

- WS-Notification: permite que o *Web Service* seja configurado como um produtor de notificações. Desta maneira, caso ocorra alguma mudança no *Web Service*, ou mais especificamente, no WS-Resource, a mudança é notificada para todos os assinantes. As notificações a serem propagadas devem ser definidas pelo programador.
- WS-Addressing: provê mecanismos para endereçar *Web Services* de maneira mais versátil. Pode ser usado ainda para endereçar juntos o *Web Service* e o WS-Resource.

• Serviços Web: proporciona a base que envolve, tecnologias, padrões, protocolos e ferramentas, utilizados na implementação da arquitetura OGSA.

3.2 ESCALONAMENTO EM GRADE COMPUTACIONAL

A função de um escalonador é definir, dentre uma variedade disponível, quais os recursos mais adequados para prover um bom desempenho para as aplicações. O que inclui: aumentar a taxa de transferência (*throughput*) do sistema, aumentar a utilização dos recursos, balancear a carga do sistema e minimizar o tempo de resposta (REIS, 2005). Um escalonador determina, para cada aplicação, onde e quando ela será executada, conforme a disponibilidade dos recursos. Para reduzir o tempo de execução em ambientes heterogêneos, como os de grade computacional, são utilizados algoritmos de balanceamento de carga. Esses algoritmos são responsáveis por associar recursos às tarefas garantindo o nivelamento da carga entre os nós que compõem a grade. Na seção 3.2.1 será melhor detalhado o funcionamento do balanceamento de carga.

Embora alguns métodos de escalonamento para sistemas distribuídos possam ser aplicados às grades computacionais, estes não oferecem resultados satisfatórios devido às seguintes particulares de um ambiente de grade computacional (ZHU, 2003):

• Recursos disponíveis heterogêneos – que implica em diferentes capacidades de processamento;

• Autonomia de recursos – os recursos podem ter suas próprias políticas de escalonamento que devem ser priorizadas;

• Recursos não são dedicados – podendo causar diferentes comportamentos e desempenhos;

• Aplicações diversificadas e ampla distribuição – aplicações com usuários e requisitos diversos o que dificulta a construção de um escalonador geral;

• Múltiplos domínios administrativos – redes e computadores pertencem a domínios administrativos diversificados cada um com sua política de utilização própria.

Em uma grade computacional o tipo de escalonamento mais comumente utilizado é o hierárquico. Neste modelo existem dois ou mais níveis de escalonadores (OLIVEIRA, 2006). Os escalonadores do nível mais alto são denominados meta-escalonadores e são responsáveis pelos escalonadores do nível inferior. Cabe ao escalonador de nível mais baixo a responsabilidade de administrar os recursos computacionais. Quem define onde os recursos serão executados é o meta-escalonador. Para isso, é necessário que os escalonadores de nível mais baixo enviem seus estados, periodicamente, ao meta-escalonador. O meta-escalonador deve ser capaz de (TONELLOTTO; WIEDER; YAHYAPOUR, 2006): (i) interagir com os administradores dos recursos locais; (ii) interagir com serviços de grade, tais como, segurança, serviços de informação e execução; e, (iii) definir o escalonamento e retornar o resultado da execução.

A Figura 3.5 representa o funcionamento de um escalonamento num ambiente de grade computacional. Inicialmente, o meta-escalonador recebe requisições dos usuários. Em seguida, seleciona os recursos a serem usados pelas aplicações conforme informações dos serviços a serem executados. Com base nessas informações verifica quais os recursos disponíveis e escalona suas tarefas. O serviço local efetua, então, o escalonamento local e execução da tarefa, retornando o resultado ao meta-escalonador.

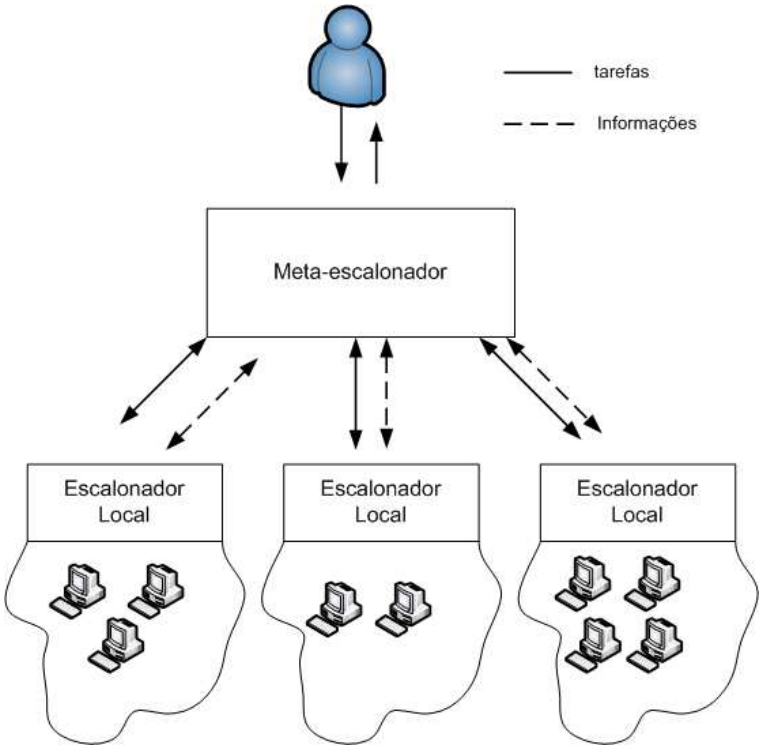

Figura 3.5. Escalonamento em Grade Computacional

Como exemplo de meta-escalonadores de grade computacional, pode-se destacar: MyGrid (CIRNE et al., 2003) – escalonador do OurGrid responsável por escalonar tarefas de aplicações BoT (*Bag of Task*), Condor-G (CONDORG, 2009), Nimrod/G (NIMRODG, 2009) e GridWay (GRIDWAY, 2009). Por ter sido o meta-escalonador utilizado no projeto, o GridWay será detalhado na Seção 3.2.2. Como exemplo de escalonadores locais, pode-se citar: Condor (CONDOR, 2009), LSF (ZHOU, 1992), e Torque (TORQUE, 2009). Por ter sido o escalonador local utilizado no projeto, o Torque será detalhado na Seção 3.2.3.

3.2.1 Balanceamento de Carga

Os algoritmos de balanceamento de carga são responsáveis por associar recursos às tarefas para garantir o nivelamento de carga entre os nós participantes da grade computacional, e assim, minimizar a máxima utilização dos recursos. Para isso, avalia as

condições instantâneas do sistema para a tomada de decisão.

De uma forma geral, o balanceamento de carga pode ser dividido em balanceamento de carga estático e balanceamento de carga dinâmico (YAGOUBI; SLIMANI, 2007). O balanceamento de carga estático é também conhecido como distribuição determinística (YAGOUBI; SLIMANI, 2007). Neste tipo de balanceamento, uma determinada tarefa é atribuída a um recurso fixo. A alocação das tarefas sempre é feita para o mesmo recurso, sem levar em consideração as mudanças que podem ocorrer no ambiente. O balanceamento de carga dinâmico leva em consideração o fato de que os parâmetros do sistema não podem ser conhecidos antecipadamente e portanto, usar um esquema fixo ou estático irá produzir eventualmente resultados insatisfatórios.

No balanceamento de carga dinâmico, a responsabilidade de tomar as decisões pode estar em uma localização centralizada ou ser compartilhada em múltiplas localizações distribuídas. A estratégia centralizada tem a vantagem de ser fácil de implementar, porém, não permite escalabilidade, tolerância a falhas, além de poder se tornar um gargalo de performance. Na estratégia distribuída, o estado dos recursos é distribuído através dos nós, que são responsáveis por administrar seus próprios recursos ou alocar recursos que estão em suas filas para outros nós.

3.2.2 GridWay

O GridWay é um componente, de código aberto, baseado no *Globus Toolkit* para metaescalonamento em grade computacional. Disponibiliza ao usuário final, desenvolvedores de aplicação e administradores de infraestrutrua de grades computacionais uma funcionalidade de escalonamento similar a encontrada em sistemas DRM local (GRIDWAY, 2009). Permite o compartilhamento de recursos computacionais, tais como, clusters, conjunto de computadores (em inglês, *computing farms*), servidores, supercomputadores, entre outros, administrados por diferentes sistemas de administração de recursos distribuídos (DRM) como, por exemplo, Torque (TORQUE, 2009), LSF (ZHOU, 1992) e Condor (CONDOR, 2009).

As principais características do GridWay, incluem (GRIDWAY, 2009):

• Arquitetura flexível e extensível;

• Capacidades avançadas de escalonamento numa grade composta por plataformas de computação distintas;

• Dispositivos de execução para interfaces pre-WS GRAM e WS GRAM;

• Dispositivos de transferência para interface GridFTP e RFT;

• Suporte total para as linguagens C e Java através da API DRMAA;

• Disponibiliza comandos similares ao DRM para: submeter, monitorar, sincronizar e controlar tarefas; monitorar recursos Globus e usuários; e obter informações a respeito da grade;

• Possibilita re-escalonamento dinâmico em situações onde a tarefa inicialmente alocada precisa ser re-escalonada devido a uma necessidade de migração;

• Capacidade de detecção de falhas e recuperação. As falhas detectadas podem incluir cancelamento remoto de tarefa, falhas por parada (*crash*) ou interrupção, e desconexão da rede;

• Possui interoperabilidade entre diferentes infraestruturas de grade como Globus, EGEE (EGEE, 2009), UNICORE (UNICORE, 2009), entre outros.

A arquitetura do GridWay consiste dos seguintes componentes, conforme apresentado na Figura 3.6 (GRIDWAY, 2009):

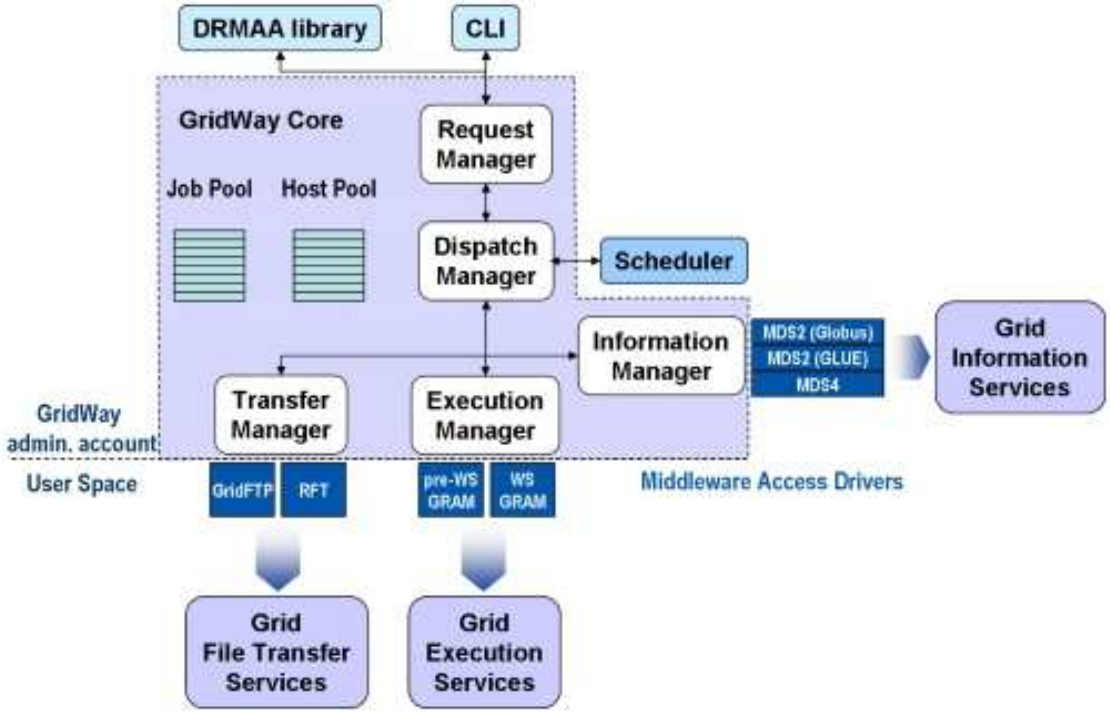

Figura 3.6. Arquitetura do GridWay (GRIDWAY, 2009)

• **Interface do Usuário (*User Interface*)** – representado na Figura 3.6 por CLI (*Command Line Interface*) - interface por linha de comando. Permite a submissão dos comandos inseridos pelo usuário. Esses comandos podem ser submeter, cancelar, monitorar e sincronizar uma tarefa. Inclui a API DRMAA (*Distributed Resource Management Application API*) que utiliza o padrão OGF (OGF, 2008) para dar suporte ao desenvolvimento de aplicações distribuídas através da linguagem C e Java.

• **Middleware de Acesso aos Dispositivos (MAD - *Middleware Access Drivers*)** – responsável por permitir a comunicação entre o ambiente externo à grade com o ambiente interno à grade. Obtém informações do ambiente, como por exemplo, os recursos disponíveis, para disponibilizar para o usuário final. O MAD é acessado por

todos os componentes do núcleo do GridWay para execução de suas tarefas.

• **Núcleo do GridWay (*GridWay Core*)** – responsável pelo gerenciamento da execução das tarefas e provedor de serviços (*Resource Broker*), disponibilizando escalonamento avançado, e capacidade para recuperação de falhas. Os seguintes componentes fazem parte do núcleo do GridWay: (i) administrador de requisições (*Request Manager*) - responsável por administrar as requisições dos clientes; (ii) administrador de submissões (*Dispatch Manager*) – realiza todos os estágios da submissão da tarefa e monitora a execução da tarefa para verificar se foi executada corretamente; (iii) administrador de transferências (*Transfer Manager* – através do seu drive de acesso ao middleware (MADs - *Middleware Access Driver*), é responsável pela transferência de arquivos, configuração remota do diretório de trabalho e organização remota das estações de trabalho; (iv) administrador de execuções (*Execution Manager*) – através do seu drive de acesso ao middleware (MADs - *Middleware Access Driver*), é responsável pela administração e execução das tarefas; e, (v) administrador de informações (*Information Manager*) – através do seu drive de acesso ao middleware (MADs - *Middleware Access Driver*), é responsável por descobrir e monitorar informações sobre as estações.

• **Escalonador (*Scheduler*)** – responsável por tomar as decisões relacionadas ao escalonamento das tarefas nos recursos disponíveis. O escalonamento, no GridWay funciona de maneira adaptativa, onde, no momento da execução são definidas quais as melhores condições de escalonamento. Para isso, são analisadas as informações do ambiente, o histórico de execuções anteriores e as prioridades definidas para os recursos e os usuários.

• **Interface MAD de Administração de Informação (*Information Manager MAD*)** – interface entre o GridWay e os serviços de monitoramento e descoberta disponíveis na infraestrutura de grade computacional.

• **Interface MAD de Administração de Execução (*Execution Manager MAD*)** – interface entre o GridWay e os serviços de administração de tarefas disponíveis nos recursos da grade computacional.

• **Interface MAD de Administração de Transferência (*Transfer Manager MAD*)** – interface entre o GridWay e os serviços de administração de dados disponíveis nos recursos da grade computacional.

De forma resumida e simplificada, pode-se sequenciar o funcionamento do GridWay da seguinte maneira: (i) usuário cria um arquivo com tarefa; (ii) usuário submete a tarefa para o GridWay; (iii) o GridWay seleciona um recurso conforme necessidades da tarefa; (iv) GridWay submete a tarefa ao recurso; (v) o GridWay monitora e reporta o status da tarefa; (vi) o usuário recebe o arquivo de saída da tarefa localmente.

3.2.3 Torque

Torque é um gerenciador de recursos, de código aberto, que prover controle sobre tarefas em lote e recursos de computação distribuídos. Foi criado em 2004, baseado na

versão 2.3.12 do OpenPBS (OPENPBS, 2009), com a colaboração de várias organizações e projetos, como NCSA (NCSA, 2009) e TeraGrid (TERAGRID, 2009). A lista completa de colaboradores encontra-se em (TORQUE, 2009). O Torque incorpora melhorias significativas em relação à tolerância a falhas e escalabilidade, além de apresentar novas extensões, como capacidade para obter grande disponibilidade e suportar altas taxas de transferência. Atualmente, encontra-se na versão 2.4.2 e suas principais características incluem (TORQUE, 2009):

• **Tolerância a Falhas** – implementa verificação dos nós que estão indisponíveis (fora do ar), suporte a várias condições de checagem de falhas, além de incluir opções de recuperação de falhas;

• **Interface de Escalonamento** – disponibiliza interfaces extendidas de consulta e controle que fornecem ao escalonador, respectivamente, informações adicionais e mais apuradas sobre o escalonamento das tarefas, e um maior controle sobre os atributos e o comportamento das tarefas. Além disso, permite a obtenção de dados estatísticos das tarefas já executadas.

• **Escalabilidade** – em relação à escalabilidade apresenta: melhorias no servidor para utilização do modelo de comunicação orientado a messagem (MOM - *Message-Oriented Middleware*); aumento da capacidade para controlar clusters com capacidade acima de 15 TeraFlops e com mais de 2.500 processadores; capacidade para administrar grande volume de tarefas (superior a 2.000 processos); e, capacidade para suportar grande quantidade de mensagens provenientes do servidor.

• **Usabilidade** – disponibiliza logs com informações mais completas e entendíveis pelos usuários, ao invés de apresentar, por exemplo, apenas os códigos do erro e do comando.

3.3 GRADE DE DADOS (DATA GRID)

As grades de dados são próprias para aplicações que necessitam de computação intensiva, além de acesso e análise de grandes quantidades de dados. São usadas para compartilhar, acessar, transportar, processar e administrar grandes coleções de dados distribuídos geograficamente. Combinam tecnologias computacionais de última geração com alta performance da rede e técnicas de administração de armazenamento em larga escala. Projetos como (GENOME, 2007), (MAMMOGRID, 2007), (EUDATAGRID, 2007), entre outros, fazem uso dessa tecnologia. São características próprias das grades de dados: grandes conjuntos de dados, computação de alta vazão (utiliza os recursos livres para aumentar a vazão agregada de tarefas) e sistema de armazenamento heterogêneo. Uma grade de dados deve prover no mínimo duas funcionalidades básicas (VENUGOPAL; BUYYA; RAMAMOHANARAO, 2006): mecanismo de transferência de dados com alta performance e confiável, e mecanismo de administração e descobrimento de réplicas.

Para que o usuário possa obter o máximo de benefícios dessa infraestrutura é necessário que seja possível (VENUGOPAL; BUYYA; RAMAMOHANARAO, 2006): (i)

encontrar dentro do numeroso conjunto de dados, disponível, o conjunto de dados requerido. Para isso, deve ser possível encontrar os recursos apropriados para acessar os dados; (ii) transferir conjuntos de dados de grandes proporções no menor tempo possível; (iii) administrar múltiplas cópias de dados; (iv) selecionar os recursos computacionais apropriados e processar os dados neles; (v) Administrar as permissões de acesso para os dados.

Todas as operações em uma grade de dados são controladas por uma camada de segurança que valida a autenticação das entidades e assegura que apenas as operações autorizadas serão executadas. Outro aspecto das grades de dados é a manutenção das coleções de dados compartilhadas através de diversos domínios administrativos. Essas coleções são mantidas independente do sistema de armazenamento que está por trás, sendo possível incluir ou exluir novos sitios com facilidade. Em (VENUGOPAL; BUYYA; RAMAMOHANARAO, 2006) é apresentada a arquitetura de camadas de uma grade de dados. Essa arquitetura foi criada seguindo as definições apresentadas em (FOSTER; KESSELMAN; TUECKE, 2001) e (BAKER; BUYYA; LAFORENZA, 2002). A Figura 3.7 apresenta essas camadas que são explicitadas a seguir.

• Fabrica – consiste dos recursos computacionais distribuídos (clusters, supercomputadores, entre outros), recursos de armazenamento (RAID, fita, entre outros) e instrumentos (telescópios, entre outros) conectados em uma rede com alta largura de banda. Cada um desses recursos executa programas como sistema operacional, submissão de tarefas e administração de sistemas, além de um sistema de administração de banco de dados relacional.

• Comunicação – consiste de protocolos usados para procurar recursos na camada fábrica e conduzir as transferências de dados entre eles. Esses protocolos são construídos no núcleo dos protocolos de comunicação como TCP/IP e protocolos de autenticação como o PKI (*Public Key Infrastructure*), senhas ou SSL (*Secure Socket Layer*). Esses mecanismos formam parte da infraestrutura de segurança da grade, denominado de GSI (*Grid Security Infrastructure*). Os protocolos de transferência de arquivos como o GRIDFTP, por exemplo, provêm serviços para transferência de dados eficiente entre recursos no *data grid*.

• Serviços de *Data Grid* – prover serviços para administrar e processar dados em um *data grid*. Os serviços do nível do núcleo como replicação, descobrimento de dados e submissão de tarefas provêm acesso transparente para a distribuição de dados e para computação. Os serviços em nível de usuário, tais como, *resource broker*(seleção dos recursos para usuários baseado nos seus requerimentos) e administração de réplicas provêm mecanismos que permitem uma administração eficiente dos recursos escondidos através de comandos intuitivos e API's (interface de programação de aplicativos).

• Aplicações – são serviços específicos que são fornecidos para os usuários através de invocações. São específicos para determinados domínios como energia, biologia, entre outros, cada um com sua interface específica.

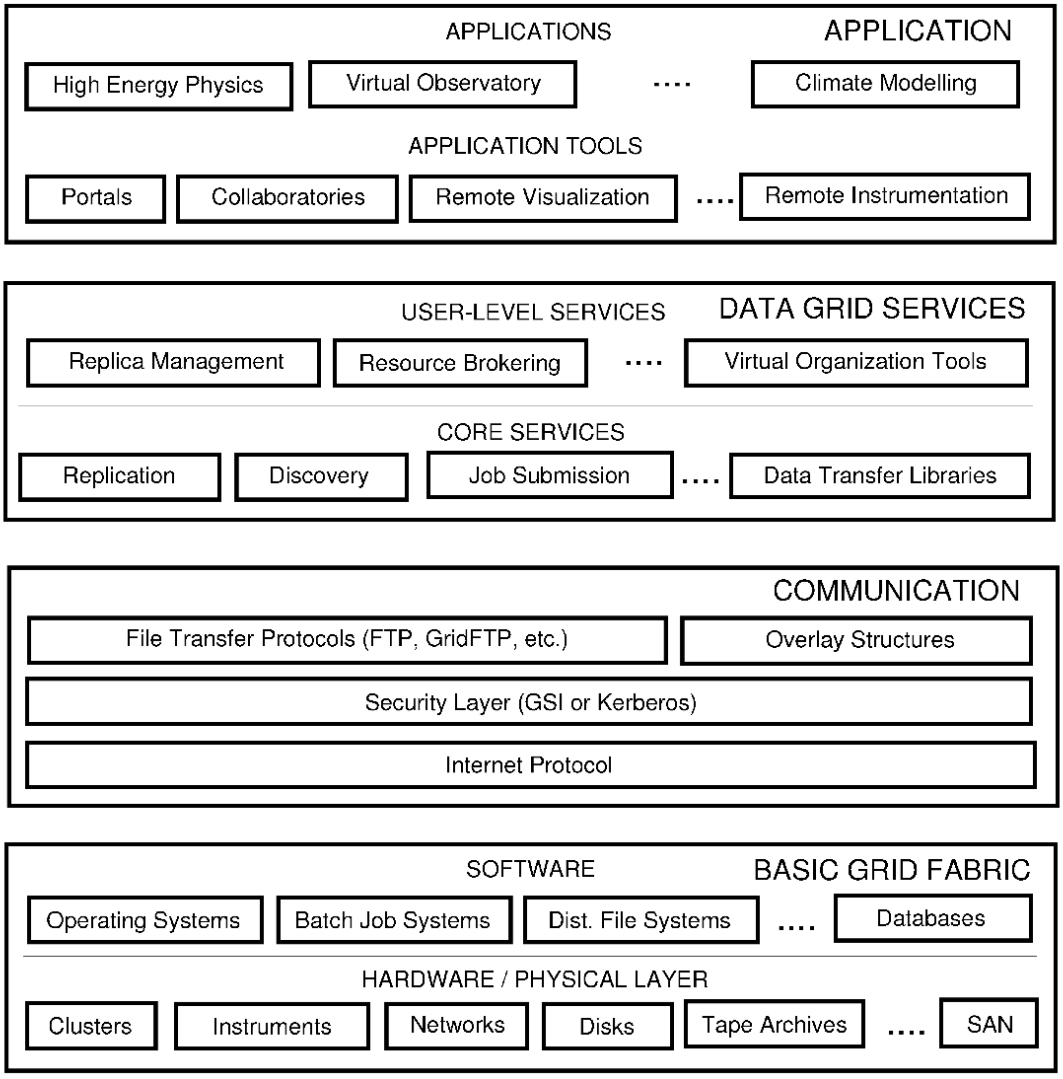

Figura 3.7. Arquitetura em Camadas de uma Grade de Dados (VENUGOPAL; BUYYA; RAMAMOHANARAO, 2006)

Em (VENUGOPAL; BUYYA; RAMAMOHANARAO, 2006) é apresentada uma taxonomia relacionada a uma grade de dados. Nessa taxonomia são apresentados quatro elementos que constituem um ambiente de grade de dados. Esses elementos envolvem organização, transporte de dados, replicação de dados e escalonamento, e serão apresentados a seguir.

i) Organização
Refere-se à maneira como uma grade de dados pode estar organizada em relação aos seguitnes aspectos:

a. Modelo
Determina a maneira como cada fonte de dados é organizada no sistema. Vários fatores determinam a definição do modelo: depende se a fonte de dados é única ou distribuída, do tamanho do dado e do modo de compartilhamento. Os modelos podem ser:

i. Monádico (*Monadic*): É a forma mais genérica de um *data grid*. Neste modelo, todos os dados estão agrupados em um repositório central que responde às requisições dos usuários e provê os dados. Os dados podem ser de várias fontes e estará disponível através de uma interface centralizada (um portal web, por exemplo) que verifica os usuários e checa suas autorizações. Este modelo é ideal para cenários onde a sobrecarga (*overhead*) gerada pela utilização de replicação não compensa para aumentar a eficiência no acesso aos dados, como por exemplo, no caso onde os acessos são locais.

ii. Hierárquico: Esse modelo é usado em *data grids* onde existe uma única fonte de dados, e os dados devem ser distribuídos através de uma rede colaborativa. Uma vantagem desse modelo é que a manutenção da consistência dos dados é muito mais simples.

iii. Federativo: Usado por instituições que desejam compartilhar dados de base de dados já existentes. Uma instituição participante pode requerer dados de quaisquer bases dentro da federação desde que tenha a autenticação apropriada. Cada instituição retém controle sobre sua base de dados local.

iv. Híbrido: Pode combinar os três modelos apresentados acima.

b. Escopo
O escopo de um *data grid* pode ser restrito a um único domínio (intradomínio), ou composto por uma infraestrutura comum a várias áreas científicas (interdomínio).

c. Organização Virtual
Os *data grids* são formados por organizações virtuais (OV's) e, consequentemente, o projeto da OV reflete na organização social do *data grid*, podendo ser: (i) colaborativa: formada por entidades que juntas compartilham recursos e colaboram num objetivo comum. Existe um acordo implicito entre os

participantes para uso dos recursos; ou (ii) regulada: pode ser controlada por uma única organização que determina regras de acesso e compartilhamento de recursos.

d. Fonte de Dados: as fontes de dados de um *data grid* podem ser transientes (a transmissão de dados só é feita em determinados períodos. Ex.: satélite só envia dados em certos períodos do dia) ou estável.

e. Administração: a administração num *data grid* pode ser autonôma (auto organizada e administrada) ou controlada (pela intervenção humana).

ii) Transporte de Dados
O mecanismo de transporte de dados é uma das tecnologias fundamentais que está por trás do *data grid*. Envolve não apenas a transferência de bits através dos recursos, mas também, outros aspectos relacionados ao acesso aos dados como segurança (envolvendo autenticação, autorização e criptografia), mecanismos de tolerância a falhas (muito importante por envolver a transferência de grandes arquivos de dados) e modo de transferência dos dados (com bloqueio ou não, dados comprimidos ou não, entre outros).

iii) Replicação de Dados e Armazenamento
A replicação é um método chave utilizado para garantir escalabilidade, confiabilidade no acesso aos dados e para previnir largura de banda. A replicação é limitada pelo tamanho do armazenamento disponível nos diferentes sitios dentro do *data grid* e da largura de banda entre esses sitios. As estratégias de replicação determinam quando e onde deve ser criada uma réplica do dado. Essas estratégias são guiadas por fatores como demanda de dados, condições da rede, custo e transferência. Os elementos importantes para o mecanismo de replicação são a arquitetura do sistema e a estratégia utilizada para replicar os dados. A arquitetura envolve os seguintes itens:

a. Modelo: determina o modo como os nós estarão organizados e o método da replicação. No modo centralizado, deve existir uma replica *master* que é atualizada e tem a responsibilidade de propagar as atualizações para os demais nós. No modo descentralizado, também denominado peer-to-peer (P2P), as várias cópias devem estar sincronizadas umas com as outras.

b. Topologia: os nós que estão sob um sistema de administração de réplicas podem ser organizados dentro das seguintes topologias: hierárquica (tem estrutura de uma árvore e as atualizações são propagadas por caminhos definidos), *flat* (são encontradas nos sistemas P2P e as atualizações são dependentes dos arranjos entre os pares), e híbridas (envolvem mais de uma topologia).

c. Integração do Armazenamento: a relação da replicação com o armazenamento é muito importante e determina a escalabilidade, robustez, adaptabilidade e aplicabilidade do mecanismo de replicação. Pode ser: • fortemente acoplada – amarrados à arquitetura em que foi implementado. O sistema de replicação controla o sistema de arquivos e mecanismo de I/O do disco local.

As replicações são, normalmente, disparadas através de requisições de leitura e escrita; • intermediária – exerce controle sobre o mecanismo de replicação, mas não sobre o armazenamento dos recursos; • fracamente acoplado – a replicação é iniciada e administrada pelas aplicações e pelos usuários.

d. Protocolos de Transferência: define o tipo de protocolo de transferência a ser utilizado, se aberto (permite que os clientes transfiram dados independente do sistema de administração de réplicas), ou se fechado (restrigem o acesso das réplicas às suas bibliotecas cliente).

e. Metadados: é muito difícil, se não impossível, para os usuários, identificar grandes conjuntos de dados particulares. Por isso, o uso de metadados é fundamental. Os metadados nada mais são do que dados sobre os dados. É a informação que descreve o conjunto de dados e pode consistir dos seguintes atributos: nome, hora da criação, tamanho em disco, hora/data da última atualização, entre outros. Um serviço de metadados pode ser tratado como um serviço de diretório distribuído, como é provido pelo protocolo LDAP (*Lightweight Directory Access Protocol*). Neste protocolo, as informações estão organizadas em uma estrutura de árvore. Ele possui informações sobre os metadados e a localização física dos arquivos distribuídos e define o método no qual um diretório é acessado.

f. Propagação da Atualização: dentro de um *data grid*, um dado é geralmente atualizado em um sitio e as atualizações são propagadas para as demais réplicas. Esse procedimento pode ser feito de forma síncrona (não se encaixa no *data grid*) ou assíncrona, podendo ser: • de maneira epidêmica - um sitio propaga as atualizações aos demais, ou • sob-demanda - os sitios recebem notificações sobre as atualizações e decidem quando atualizar as cópias.

g. Organização do Catálogo: define a maneira como o catálogo de réplicas estará organizado, facilitando assim o seu acesso.

iv) Escalonamento
Estratégias de escalonamento são importantes para aplicações cujas tarefas movimentam grupo de dados. Nesses casos é importante utilizar os nós onde os dados estejam mais próximo ou nos já existem do que tranferí-los para outro local. Os escalonadores têm que avaliar a disponibilidade da largura de banda e a latência de transferência entre um nó computacional no qual uma tarefa está sendo submetida e os recursos de armazenamento que cada dado requer para ser recuperado.

Em (RANGANATHAN; FOSTER, 2002), é proposta uma arquitetura de escalonamento para aplicações de dados intensivos que consiste em três componentes: (i) escalonador externo que decide para qual nó a tarefa deve ser submetida; (ii) o escalonador local, em cada nó, que decide a prioridade das tarefas que chegam dos outros nós; e, (iii) escalonador do conjunto dos dados que determina quais conjuntos de dados serão replicados ou apagados. Ao analisar algoritmos de escalonamento envolvendo esse tipo de arquitetura chegou-se a seguinte conclusão: a pior performance foi onde ocorreram a ausência de replicação. Isto ocorreu, porque alguns nós

que hospedaram os dados foram sobrecarregados. O melhor desempenho foi obtido utilizando essa mesma estratégia, porém, incluindo agendamento de replicação de dados. A maioria das estratégias utilizadas para *data grid* tentam reduzir o tempo de completude mínimo de cada tarefa que consiste da diferença entre o tempo em que a tarefa é submetida para um recurso computacional e o tempo em que esta é completada.

3.4 GLOBUS TOOLKIT 4 (GT4)

O *Globus Toolkit* (GT) é um conjunto de ferramentas e bibliotecas que dão suporte ao desenvolvimento de aplicações em grade computacional. Ele foi desenvolvido no final dos anos 90, pela *Globus Alliance* (ALLIANCE, 2008a), para dar suporte ao desenvolvimento de infraestruturas e aplicações computacionais distribuídas orientadas a serviço (FOSTER, 2005). Os idealizadores desse projeto foram os pesquisadores Ian Foster (*University of Chicago*) e Carl Kesselman (*University of Southern California*) (ALLEMAND, 2006). Os componentes do GT provêm segurança, acesso aos recursos, movimentação e acesso a dados, descobrimento de recursos, monitoramento do ambiente, serviços de comunicação, entre outras funcionalidades. É possível obter suporte e treinamento dos seus componentes através de salas de bate-papo (*chats*), lista de discussões e workshops disponíveis em (GT4, 2007b).

Atualmente, o Globus encontra-se na versão 4 (GT4), versão baseada em Serviços Web (*Web Services*). As versões 1 (GT1) e 2 (GT2) são conhecidas como versões pré *Web Services*. A partir da versão 3 (GT3) os Serviços Web começaram a ser utilizados, se tornando totalmente compatível na versão 4. Porém, nem todos os componentes do GT4 são baseados em *Web Services*, como é o caso o GridFTP. O GT4 possui uma série de melhorias em relação às versões anteriores em termos de robustez, performance, usabilidade, documentação, seguimento de padrões (os principais padrões seguidos são OGSA, OGSI e *Web Services*) e funcionalidades (FOSTER, 2005).

O GT4 disponibiliza (FOSTER, 2005): (i) um conjunto de serviços focados na administração da infraestrutura; (ii) ferramentas para construção de novos *Web Services* nas linguagens C, Java e Python; (iii) infraestrutura de segurança baseadas em padrões; (iv) API's cliente, em diferentes linguagens, e programas de linhas de comando para acesso aos vários serviços disponibilizados; (v) documentação detalhada sobre seus componentes, suas interfaces e como estes podem ser usados para construir aplicações.

Os componentes do GT4 estão agrupados em cinco categorias, conforme mostra a Figura 3.8. A Figura apresenta a distinção entre os componentes que são baseados em *Web Services* (*WS Components*) dos que não são (*Non-WS Components*). Os componentes que estão representados em um quadrado fechado são os que fazem parte do núcleo GT4, os que estão em quadrado tracejado são contribuições ou versões que serão aperfeiçoadas futuramente e, os que estão em um quadrado pontilhado são os que serão trocados em versões futuras (GT4, 2007a; SENGER; STANZANI; RONDINI, 2006). A seguir esses componentes serão explicitados.

1) Segurança (*Security*)

Disponibiliza todo o suporte necessário para implementação de segurança, tais como, verificação das identidades dos usuários e recursos, proteção para troca de mensagens, controle de acesso a recursos e serviços e implementação de políticas de acesso. O componente que provê infraestrutura de segurança é denominado GSI, acrônimo de *Globus Security Infrastructure*. Todos os usuários e serviços da grade são identificados através de certificados. Os certificados GSI seguem o padrão X.509 que é um padrão internacional de certificado, determinado pela IETF - *Internet Engineering Task Force* (IETF, 2009). O GSI é composto pelos seguintes componentes:

• *Credential Management*: disponibiliza duas ferramentas para gerenciamento de credenciais: SimpleCA e Myproxy. A SimpleCA é uma autoridade certificadora usada para geração de certificados para usuários e servidores. O MyProxy é um repositório de credenciais, onde os usuários armazenam um certificado, podendo obtê-lo a qualquer momento através da rede.

• *Pre-WS Authentication and Authorization*: disponibiliza API's e ferramentas para autenticação, autorização e gerenciamento de certificados. A autenticação é feita com o uso de chaves públicas (PKI). Pode utilizar, também, mecanismos de delegação através de certificados proxy (*Proxy Certificate*), que funciona como uma espécie de procuração onde uma entidade assina e entrega a outra entidade a permissão para executar suas tarefas.

• *Authentication and Authorization*: contém classes e bibliotecas Java que permitem autenticação e autorização para os componentes *Web Services* do GT4 e para outros componentes *Web Services* que fazem parte da comunidade da grade. Para isso, usa o padrão X.509 e *GSI Secure Conversation* para autenticação e proteção da mensagem, e um *framework* com vários mecanismos de autorização.

• *Delegation*: através da utilização de chaves públicas e privadas, provê uma interface para delegação de credenciais para permitir acesso aos recursos da grade. Disponibiliza uma única credencial para ser compartilhada através de múltiplas invocações de serviços, por exemplo, pode ser usada para multiplas submissões de tarefas. Provê, também, mecanismo para renovação de credencial.

• *Community Authorization Service* (CAS): responsável pelo gerenciamento das políticas de controle de acesso aos recursos da OV.

2) Gerenciamento de Dados (*Data Management*)

Responsável pela administração de grandes volumes de dados (acesso, movimentação e gerência). Os componentes disponíveis para gerenciamento de dados pertencem, basicamente, a duas categorias: movimentação de dados e replicação de dados. A movimentação de dados abrange as ferramentas GridFTP e RFT. Além disso, o GT4 disponibiliza o OGSA-DAI, para acesso e integração estruturada e semi-estruturada de dados, e o DRS que é o serviço de replicação de dados.

• *Replica Location Service* (RLS): serviço responsável pelo registro e localização de réplicas dos dados distribuídos na grade. Mantém informações sobre os arquivos distribuídos, aumentando, assim, a disponibilidade dos dados, e com isso, diminuindo os pontos de falhas. Para evitar a ocorrência de conflitos entre os nomes dos arquivos, o RLS utiliza um esquema de nomes lógicos e físicos, onde o os arquivos lógicos possuem um identificador único para os arquivos e o nome físico indica a localização dos arquivos nos sistemas de armazenamento físico.

• GridFTP: responsável pela transferência confiável, segura e rápida de grandes quantidade de dados. É uma extensão do protocolo de transferência de dados, FTP. Seu desempenho é considerado em torno de 3 a 5 vezes melhor do que o FTP (GT4, 2007b). Pode ser usado sozinho ou como parte do GT. O GridFTP provê script em linha de comando para o cliente, um servidor altamente extensível, e um conjunto de bibliotecas de desenvolvimento para soluções customizadas.

• *Reliable File Transfer* (RFT): é um serviço Web que provê interfaces para controle e monitoramento de transferências de arquivos usando servidores GridFTP. O controle da transferência do cliente é armazenado em um serviço de grade, sendo assim, ele pode ser administrado através de consultas à interface de serviço de dados, disponível para todos os serviços de grade. A implementação do RFT no GT4 usa mensagens SOAP, como padrão, sob HTTP para submeter e administrar as transferências do GridFTP, bem como, as exclusões de arquivos e diretórios usando o GridFTP. O serviço provê, também, uma interface para controlar vários parâmetros de transferência, tais como, tamanho do *buffer*[1] TCP, fluxos de dados (*streams*) paralelos, entre outros. O usuário cria um recurso RFT através da submissão de uma requisição de transferência ao serviço *RFT Factory Service*. O serviço é, então, criado após o usuário ter sido devidamente autorizado e autenticado.

• OGSA *Data Access and Integration* (OGSA-DAI): disponibiliza um conjunto de ferramentas, em Java, para acesso e integração de recursos de dados, tais como, arquivos e banco de dados relacional e XML que fazem parte da grade. O OGSA-DAI não foi construído e nem é instalado como parte da instalação padrão do GT4, pode ser adquirido como um pacote no diretório de contribuições do GT. Este componente será detalhado na Seção 3.4.1.

• *Data Replication Service* (DRS): é um serviço de administração de dados que foi criado baseado nos componentes RFT e RLS. Sua função é assegurar que um conjunto de arquivos especificados existem num dado local. Seu procedimento inicia através da consulta ao RLS para descobrir onde os arquivos desejados estão armazenados na grade. Depois da localização dos arquivos, o DRS cria uma requisição de transferência que é executada pelo RFT. Quando a transferência é concluída, o DRS registra a nova réplica no RLS. O DRS é implementado como um *Web Service* e satisfaz as especificações do WSRF. Quando uma requisição do DRS é recebida, é criado um WS-Resource que é usado para manter o estado sobre cada arquivo que está sendo replicado, incluindo as

[1] *Buffer* é uma região de memória temporária utilizada para escrita e leitura de dados.

operações que falharam ou que foram realizadas com sucesso.

3) Gerenciamento da Execução (*Execution Management*)

Responsável pela inicialização, monitoramento, gerenciamento da execução das tarefas (*jobs*), escalonamento e/ou coordenação dos *jobs*[2] submetidos aos recursos computacionais (ALLIANCE, 2008b). Para isso, utiliza os seguintes componentes:

• *Pre-WS Grid Resource Allocation and Management*: este componente ainda existe no GT4 apenas por própositos legais (outras versões a utilizam) e será retirado no futuro quando a implementação com *Web Service* já estiver totalmente absorvida pelos usuários.

• *Grid Resources Allocation Manager* (WS-GRAM): responsável por iniciar, monitorar, cancelar e gerenciar a execução de tarefas (*jobs*), através de uma interface de serviços web (WS-GRAM), em computadores remotos. É constituido por vários módulos que podem ser instalados de acordo com os requisitos necessários à aplicação. Os componentes que fazem parte do WS-GRAM são: componentes de protocolo, componentes de software, softwares de segurança, softwares de gerenciamento de *jobs*, softwares de gerenciamento de dados e softwares de gerenciamento de tarefas. A etapa fundamental do GRAM é a criação do recurso *ManagedJob* criado através da invocação do método *ManagedJobFactory*. O cliente que quiser submeter uma tarefa a grade deve chamar esse método. O GRAM controla o tempo de existência do job na grade. Isso pode ser feito de forma explícita, pelo cliente, ou pela configuração de tempo de escalonamento. O administrador do sistema é quem define o tempo máximo para que um *ManagedJob* possa ficar em execução.

• *Workspace Management Service* (WMS): permite que um cliente da grade crie e administre, dinamicamente, um espaço de trabalho (*workspace*) em uma máquina remota. A infraestrutura é composta por um serviço de fábrica que permite que um cliente da grade, que esteja devidamente autorizado, crie contas individuais ou grupo de contas. Exite, também, um serviço de conta que permite que o usuário administre suas contas, como por exemplo, determinar as políticas de acesso ou o tempo de existência. Esses conceitos são representados nos serviços WSRF.

• *Grid TeleControl Protocol* (GTCP): é um serviço de interface para telecontrole. Foi criado a partir da necessidade do projeto NEESgrid (NEESGRID, 2004), que é usado para controlar simulações fisicas e computacionais distribuídas geograficamente. Disponibiliza duas interfaces: serviço de interface WSRF usado pelos clientes para controlar remotamente instrumentos e simulações, e um serviço de interface para facilitar a integração de plataformas de simulação computacional ou física com o servidor GTCP.

• *Community Scheduler Framework*: provê interface e ferramentas para que os usuários da grade possam submeter seus *jobs*, criar reservas e definir diferentes políticas de escalonamento. Além de permitir o acesso a diferentes gerências de recurso, através de uma única interface, como Condor, SGE, entre outros.

[2]Os *Jobs* são tarefas computacionais que executam operações de entrada e saída.

4) Serviços de Informação (*Information Services*)

Responsável por monitorar e descobrir recursos que estão espalhados na grade computacional. Isto é feito através de mecanismos baseados na especificação WS-Notification. Os componentes desse grupo são:

• *Web Service Monitoring and Discovery System* (WebMDS): permite que os usuários vejam o monitoramento da informação através de uma interface Web padrão, sem precisar efetuar nenhum tipo de instalação de software adicional na máquina. Podem ser customizados pelos administradores de páginas Web.

• *Trigger*: efetua a comparação das informações coletadas com determinados valores ou faixas de valores, além de executar ações pré-estabelecidas quando determinadas condições forem atigindas, por exemplo, enviar um email ao administrador quando o tamanho da fila de um determinado recurso atinja seu limite.

• *Index*: componente que permite a agregação e indexação de informações de descoberta e monitoramento de recuros de grade e publicá-los em um único local.

• WebMDS: disponibiliza uma interface web para exibir informações adquiridas por outros serviços. Composto pelos seguintes componentes (NASCIMENTO; MIRANDA, 2006): (i) GRIS (*Grid Resource Information*) - responsável por obter informações da máquina onde o job está sendo executado (sistema operacional utilizado, mémoria disponível, espaço em disco, entre outros); e (ii) GIIS (*Grid Index Information Service*) - *framework* para construção de diretórios customizados, agregando informações de diversos GRIS em um único servidor.

5) Sistemas de Execução Comum (*Common Runtime*)

Conjunto de bibliotecas e ferramentas responsável por prover um conjunto de serviços Web e pré Web independente de plataforma. Dentre os serviços Web providos, podemos citar: XML, SOAP, WSDL, SMTP, entre outros. Permite o desenvolvimento de aplicações para fazerem uso dos recursos fornecidos pelo GT4, utilizando as linguagens C, Java ou Python. Componentes:

• Bibliotecas Comuns C: provê uma camada de abstração para tipos de dados, chamadas do sistema e estrutura de dados usadas através do GT e útil para aplicações que usam o GT.

• *C WS Core*: disponibiliza um conjunto de ferramentas para implementação de serviços e clientes Web através da utilização da linguagem C, podendo fazer ligações com outras linguagens de programação. Os seguintes padrões são suportados: HTTP, SOAP, XML, WSDL, WS-Security, WS-Addressing, WS-ResourceFramework e WS-Notification.

• *Java WS Core*: sua finalidade é a mesma do C WS Core, diferindo-se apenas na linguagem de programação utilizada, que é a linguagem Java. Está dividido em duas partes: o serviço e o recurso.

• *Python WS Core*: provê um conjunto de ferramentas que permitem a construção de componentes Web Services, através da linguagem Python que utilizam o WSRF. É também conhecido como pyGridWare.

• *Globus XIO - eXtensible Input Output*: biblioteca responsável por controlar as conexões do Globus (quando são estabelecidas e encerradas), para isso, provê uma API simples e intuitiva. Seus principais objetivos são: provê uma única API para todos os protocolos de I/O da grade, e minimizar o tempo de desenvolvimento para criação ou prototipação de novos protocolos.

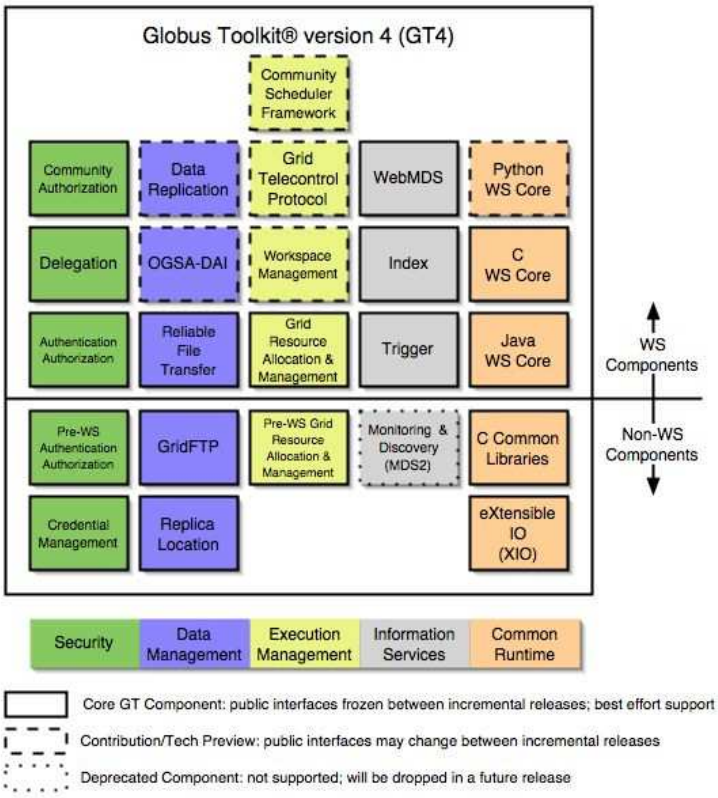

Figura 3.8. Componentes do GT4 (GT4, 2007a; SENGER; STANZANI; RONDINI, 2006)

3.4.1 OGSA-DAI (Open Grid Services Architecture - Data Access)

Foi criado em 2002, como colaboração entre várias instituições acadêmicas do Reino Unido e parceiros industriais com o objetivo de permitir a integração de bases de dados em uma infraestrutura de grade computacional (ANTONIOLETTI et al., 2005). O OGSA-DAI permite o compartilhamento de base de dados relacional, banco de dados XML e arquivo indexado. Atualmente, encontra-se na versão 3, lançada em 2007. Essa versão envolve um redesenho e implementação completa em relação às versões anteriores para seguir padronizações e atender o número de usuários. É um framework extensível,

acessado via *Web Services* que executa um fluxo de trabalho (*workflow*) utilizando recursos de dados heterogêneos para prover acesso, integração, transformação e entrega de dados em uma grade computacional.

As seguintes operações são possíveis ao se utilizar o OGSA-DAI (ANTONIOLETTI et al., 2005):

• Visualizar e consultar Banco de Dados que fazem parte da grade, através de uma interface única, independente de qual SGBD foi utilizado;

• Executar consultas simultâneas a várias bases de dados, permitindo a unificação dos resultados, sem se preocupar com o SGBD utilizado;

• Definir destinos para os resultados das consultas;

• Permitir que os dados obtidos possam ser enviados não apenas para o cliente, mas também, para outros recursos, deixando-os disponíveis para uso pelo recurso quando necessário.

Dentre os vários conceitos relacionados ao OGSA-DAI, vale destacar (ANTONIOLETTI et al., 2007):

• *Activity* – são interfaces para manipulação de recursos de dados disponibilizados na grade. Esses recursos podem ser consultas SQL, lista de arquivos em um diretório, entrega de dados para um servidor FTP, entre outros.

• *Block* – pedaço de dado: número, booelano, string, byte, array de bytes ou tupla, por exemplo.

• *Workflow* – uma ou mais sequências de *activities* conectados. Um *workflow* pode ter múltiplas sequencias de *activities* que podem ser executadas em série ou em paralelo.

• *Request* – é um *workflow*, com identidade única, submetido pelo cliente ao OGSA-DAI.

• *Session* – um contexto que permite múltiplas requisições para compartilhar estado.

• *Activity Framework* – componente de software que cria, inicializa e processa *workflows* e *activities* contidos numa requisição.

• *Engine* – componente de software que enfileira as requisições e as submetem para processamento no *Activity Framework*.

• *OGSA-DAI Core* – componentes de software que provêm as funcionalidades fundamentais do OGSA-DAI, incluindo, *Framework Activity* e *Engine*, bem como, componentes de interação de recursos de dados, persistência e configuração de componentes, e utilidades.

• *Presentation Layer* – um *front-end* para o *OGSA-DAI Core*, através do qual os clientes podem acessar e usar o OGSA-DAI. Pode apresentar o *OGSA-DAI Core* através de *Web Services*.

• *Data Service* – *Web Service* que provê acesso ao *OGSA-DAI Core*.

Os principais componentes do OGSA-DAI estão apresentados na Figura 3.9 e serão detalhados a seguir (OGSA-DAI, 2009):

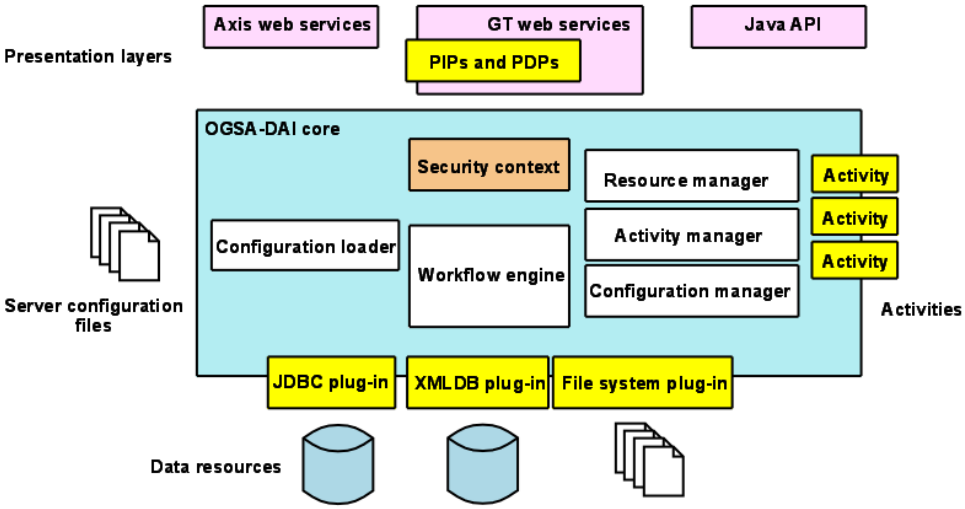

Figura 3.9. Componentes do OGSA-DAI (OGSA-DAI, 2009)

• *Configuration Loader* – guarda os arquivos de configuração do OGSA-DAI. Esses arquivos especificam a configuração do servidor que inclue disponibilidade dos *Activities*, classes da implementação do *Activity*, recursos, *Activities* suportados, classes de implementação dos recursos e todas as bases de dados de usuários e senhas.

• *Resource Manager* – disponibiliza acesso aos recursos disponíveis no servidor. O acesso é fornecido para os *Activities* e para *Presentation Layer*.

• *Activity Manager* – fornece acesso aos *Activities* disponíveis no servidor. É usado pelo *workflow Engine*.

• *Configuration Manager* – detém informação das configurações gerais do servidor de uso por outros componentes.

• *Workflow Engine* – executa os *workflows* dos clientes. Isso inclui criar objetos *Activity* correspondentes a aqueles *Activities* situados no *workflow* do cliente, monitorando a execução do *workflow* e atualizando o estado corrente da execução.

• *Data Resource* – *plug-ins* que são usados para comunicação entre os recursos de dados. Representa o ponto chave da extensibilidade do OGSA-DAI.

• *Activities* – componentes que executam as operações relacionadas aos dados disponibilizados na grade.

• *Security* – possui informações relacionadas à segurança da camada de apresentação,

como por exemplo, as credenciais de um cliente.

• *Presentation Layer* – através do qual o cliente acessa o servidor OGSA-DAI. Pode acessar Web Services através do Apache Axis, Globus Toolkit e uso direto através de API's Java.

3.4.2 CoG - Commodity Grid

O objetivo do CoG é permitir a utilização de tecnologias já conhecidas e utilizadas, denominadas, em inglês, de *commodity* com a tecnologia de grade computacional. Para isso, define interfaces entre as grades computacionais e os *frameworks* já existentes, como Web Services, Corba, DCOM, Java, entre outros (LASZEWSKI1; HATEGAN, 2005). Ou seja, define e implementa uma série de componentes que mapeiam as funcionalidades da grade computacional. Para cada *framework* existe um kit CoG, por exemplo, para a linguagem Java existe o Java CoG kit (COG, 2000). O benefício provido pelo CoG é facilitar ao desenvolvedor explorar os serviços avançados de grid, como gerenciamento de recursos e segurança, enquanto utiliza componentes que já lhes são familiares providos por *frameworks* que já estão habituados a usar, tornando assim, o uso de grid muito mais fácil. Por exemplo, no Globus Toolkit, para gerenciamento da computação remota utiliza-se uma API procedural, ao se utilizar o Java CoG Kit, por exemplo, a mesma funcionalide é provida através de um objeto *job* e eventos Java (LASZEWSKI1; HATEGAN, 2005).

A Figura 3.10 apresenta as funcionalidades e usabilidades através da utilização do Java CoG kit. Através deste, os usuários finais poderão acessar o fluxo de trabalho através de aplicações autônomas, através de estações de trabalho ou portais. Ferramentas de linha de comando permitem que os usuários definam *workflows* facilmente. A programação do *workflow* pode ser feita através de serviços, abstrações e API's. O Java CoG kit é distribuído como parte do Globus Toolkit nas versões 3 e 4.

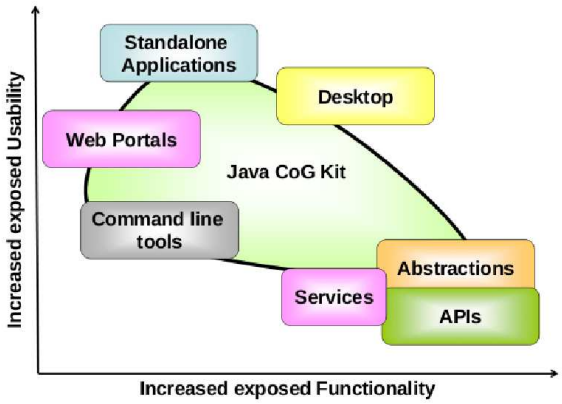

Figura 3.10. Funcionalidades e Usabilidades do Java CoG Kit (LASZEWSKI1; HATEGAN, 2005)

CAPÍTULO 4

ARQUITETURA PARA RECONHECIMENTO DE IMPRESSÃO DIGITAL ATRAVÉS DE UMA GRADE COMPUTACIONAL

Este capítulo apresenta uma arquitetura para integração de bases de dados de impressão digital, distribuídas geograficamente, e baseada nas tecnologias de grade computacional. A definição dos componentes que compõem essa arquitetura levou em consideração as pesquisas realizadas no levantamento do estado da arte de impressão digital e grade computacional. Para isso, deve atender aos objetivos apresentados na Seção 4.1.

4.1 OBJETIVOS DA ARQUITETURA

Os objetivos dessa arquitetura incluem: (i) permitir o compartilhamento de informações de impressões digitais; (ii) através do uso da tecnologia de grade computacional; (iii) para disponibilizar uma grande base de dados virtual; (iv) num ambiente seguro, consistente, multi-plataforma e independente; (v) fundamentado em soluções de código aberto e hardware heterogêneo; (vi) seguindo os padrões definidos pelo OGF (Seção 3.1). A arquitetura deve atender às premissas e restrições, apresentadas na Seção 4.2, que foram consideradas fundamentais para alcançar os objetivos apresentados.

4.2 PREMISSAS E RESTRIÇÕES

Para criação de um sistema automatizado de identficação através da impressão digital, numa grade computacional, é imprescindível que sejam consideradas as seguintes premissas e restrições relacionadas ao projeto.

4.2.1 Premissas

• Acessar bases de dados de diferentes fabricantes – deve-se utilizar ferramentas que permitam a interação de bases de dados diferentes, para que não seja necessário restringir a um único fabricante. Sendo assim, cada instituição envolvida define o SGBD de sua preferência, podendo ser desde SGBD's de código aberto a proprietários de fabricantes.

• Permitir alta escalabilidade – deve ser possível a inclusão de novas bases de dados e recursos, bem como, usuários, sem comprometer o desempenho do sistema.

• Suportar diversidade de componentes físicos – os componentes físicos utilizados

devem ser heterogêneos e fracamente acoplados e devem suportar diferentes tipos de processadores e sistemas operacionais.

• Disponibilidade – consiste na probabilidade de se ter o sistema disponível para execução em um período de tempo. O ambiente de grade computacional possibilita disponibilidade do sistema, pois o seu correto funcionamento é independente da existência de falhas. Na ocorrência de falha o metaescalonador redireciona os trabalhos para um outro recurso livre de falhas e ativo.

• Concorrência – para esse projeto diz respeito à múltiplas requisições (buscas) simultâneas no sistema. A arquitetura deve prover acesso à informações de dados de impressão digital por vários usuários ao mesmo tempo sem que ocorra interferência significativa no tempo de resposta para o usuário.

• Independência – o sistema deve depender pouco ou não depender de licenças proprietárias de softwares, de maneira que, possa ser reutilizado com facilidade, garantindo independência tecnológica ao usuário, além de otimizar o custo da implantação.

• Tolerância a falhas – a arquitetura deve possuir mecanismos para assegurar que as tarefas serão executadas, e se necessário, re-executadas, mesmo na ocorrência de falhas, garantindo, assim, a qualidade no fornecimento de serviços.

4.2.2 Restrições

• Utilizar rede de comunicação – as comunicações devem ser realizadas através do uso de redes de longas distância. Porém, devem ser utilizados mecanismos que restrinjam o acesso apenas às instituições envolvidas.

• Seguir padrões estabelecidos pelo OGF – utilizar os serviços da grade (*Grid Service*), como, autenticação, transmissão de dados, compartilhamento de informações, gerenciamento de tarefas, entre outros, seguindo um conjunto de padrões de interface para estes serviços.

• Interface do Usuário – a arquitetura deve permitir que diferentes interfaces possam acessá-la, de computadores pessoais a dispositivos móveis. A impressão digital deve ser capturada através de sensores específicos que podem estar acoplados ou externos ao hardware utilizado.

• Suportar escalonamento de dados em mais de um nível – a arquitetura deve suportar escalonamento em nível global (metaescalonamento) e local, para que seja possível determinar os recursos mais adequados para prover o melhor desempenho para a aplicação.

• Segurança – devem ser utilizados mecanismos de segurança que garantam a integridade e privacidade, através do uso de criptografia, dos dados trafegados. Além disso, a infraestrutura de segurança utilizada deve ter compatibilidade com as já existentes nas instituições participantes.

• Consistência das informações – deve-se garantir que as informações disponibilizadas

são consistentes. Sendo assim, cada instituição participante é responsável por manter os seus dados, sem que seja possível alteração por outra instituição que compõe a grade. O acesso deve ser apenas para consulta dos dados.

• Integridade dos dados – deve-se garantir que as informações obtidas são de fontes corretas e confiáveis e que em nenhuma circunstância será permitido o acesso não autorizado às informações

• Transparência – o acesso aos recursos deve ser transparente ao usuário. A infraestrutura de grade possibilita transparência para o usuário na disponibilização dos serviços de integração e processamento dos dados.

Levantados os objetivos, premissas e restrições fundamentais para a arquitetura, é possível descrevê-la apresentando suas características, componentes e interação entre os mesmos.

4.3 DESCRIÇÃO DA ARQUITETURA

A arquitetura proposta, foi definida, com base em pesquisas realizadas sobre o estado da arte de grades computacionais e impressão digital. Além de análises de resultados de aplicações semelhantes que utilizam a infraestrutura de grade computacional, tais como (YANG; HAN, 2006) e (MORETTI et al., 2007). (YANG; HAN, 2006) aborda pesquisas de bioinformática que requerem grande poder de computação para obter um melhor desempenho. E mostra como o uso de grades computacionais podem viabilizar a construção de ambientes de supercomputação. (MORETTI et al., 2007) apresenta como estudo de caso, o uso de grades computacionais numa aplicação biométrica para identificação de uma pessoa através de uma imagem em três dimensões. Apresenta a inviabilidade de executar esse tipo de processamento em uma única máquina, apresentando várias técnicas para uso de grades computacionais como solução para armazenar e comparar grande quantidade de dados de forma eficiente. Com base nessas análises, foram identificados os componentes que mais se adequavam ao tipo de aplicação desejada, atendendo aos requisitos e objetivos aos quais a aplicação se propõe. Toda a arquitetura se baseia nos padrões propostos pelo OGF.

Por propor acesso, pesquisa, manipulação e gerência de grandes volumes de dados distribuídos, essa arquitetura é classificada como uma grade de dados (*data grid*). De acordo com a taxonomia apresentada em (VENUGOPAL; BUYYA; RAMAMOHANARAO, 2006), esta arquitetura refere-se a um modelo de *data grid* federativo, por envolver instituições compartilhando base de dados já existentes, onde cada instituição é responsável pelo controle dos seus dados. O escopo envolvido é intradomínio, ou seja, restrito a uma única área – a de Segurança Pública, por exemplo. Além disso, é formada por uma OV colaborativa composta por entidades que juntas compartilham recursos e dados e colaboram num objetivo comum.

A arquitetura propõe que a OV seja composta por um conjunto de *clusters* independentes e distribuídos geograficamente. A definição por esse tipo de estrutura é devido ao

fato do *cluster* permitir um melhor controle dos dados, através de técnicas de tolerância a falhas e replicação, garantindo, assim, maior disponibilidade do serviço. Além disso, permite escalabilidade, onde novos componentes podem ser facilmente acrescentados de acordo com o aumento da carga do sistema. Isso é fundamental para o tipo de aplicação a qual a arquitetura se propõe, que normalmente envolve volumes grandiosos de dados que são gerados e analisados constantemente durante o dia.

O objetivo foi montar uma arquitetura interoperável, modular, formada por componentes que colaboram entre si para prover o serviço, mas que mantêm suas individualidades, permitindo, assim, flexibilidade e escalabilidade sem interferir nos demais componentes. A arquitetura está divida em quatro camadas, conforme apresentado na Figura 4.1, que interagem para formar uma grande base de dados virtual, viabilizando, aos usuários, acesso transparente a qualquer registro que se encontra no ambiente.

Figura 4.1. Componentes da Arquitetura

Apesar da arquitetura apresentada nesse capítulo ser direcionada ao reconhecimento de impressão digital, esta pode ser utilizada para outros tipos de aplicações. Isto deve-se ao fato de que seus componentes são independentes e genéricos sendo necessário apenas a definição das tecnologias específicas para finalidade desejada. Essa mesma arquitetura poderia ser utilizada para uma aplicação de compartilhamento de imagens de exames médicos. Por exemplo, imagens de mamografias para análise e diagnóstico de tratamentos específicos para determinadas doenças, como o câncer. Nesse caso, a camada de aplicação e a de acesso a dados seriam determinados de maneira que atendessem aos requisitos desse tipo de aplicação.

As seções 4.3.1, 4.3.2, 4.3.3 e 4.3.4 detalham as camadas que compõem a arquitetura proposta.

4.3.1 Aplicação

Camada que representa a interface do usuário, através da qual é possível interagir com o ambiente da grade. Representa o sistema de reconhecimento de impressão digital que é executada na grade, através da utilização de alguns pacotes do NFIS (Seção 2.5.3). Os pacotes do NFIS definidos para essa arquitetura são: (i) PCASYS – utilizado para efetuar a classificação da ID capturada, através de um sensor específico, conforme classes de Henry (Seção 2.3). A classe pode ser utilizada como filtro na pesquisa, reduzindo assim, o número de registros a ser pesquisado; (ii) MINDTCT – utilizado para efetuar a extração das minúcias que serão utilizadas no processo da comparação; (iii) BOZORTH3 - utilizado para efetuar a comparação das minúcias a fim de identificar as impressões digitais coincidentes. Esse pacote efetuará a comparação entre o arquivo de minúcias, da ID capturada, gerado pelo MINDTCT, com os arquivos de minúcias previamente armazenados.

A grande evolução da tecnologia de redes sem fio e miniaturização de aparelhos tem proporcionado o crescimento da integração de pequenos e portáteis dispositivos de computação. A portabilidade e a capacidade de conexão com redes nos mais diversos locais é o que torna possível a computação móvel. Dentre os dispositivos para computação móvel, tem destaque os denominados computação de mão ou em inglês, *handheld computing*). O assistente pessoal digital (PDA) é um tipo de *handheld computing*. O PDA é um computador com capacidade de processamento reduzida (em relação ao computador pessoal) através do qual é possível executar uma série de aplicações. Nesse dispositivo é possível encontrar acoplado ao mesmo câmaras, leitores de código de barras, sensores de impressão digital, entre outros acessórios especializados (COULOURIS; DOLLIMORE; KINDBERG, 2007). Permitindo, assim, que os usuários tenham acesso a serviços independente de onde estejam localizados, estando em movimento ou não. A idéia é disponibilizar acesso à informação em qualquer lugar e a qualquer momento (FIGUEIREDO; NAKAMURA, 2003).

A capacidade de executar processamento, trocar informações via rede nos mais diversos locais e ser transportado facilmente pelos usuários, motivou a criação de uma interface para acesso a essa arquitetura, também, através do PDA. Para que de qualquer lugar, que disponibilize uma rede sem fio seja possível efetuar o acesso às informações da grade. Sendo assim, essa arquitetura propõe duas interfaces de usuário, que viabilizam o uso de diferentes plataformas: (i) uma interface para computadores pessoais (*desktops* e *notebooks*), e, (ii) uma interface para dispositivos móveis, mais especificamente para PDA's.

Apesar de interfaces distintas, o procedimento para acesso aos recursos da grade deve ser transparente ao usuário, sem que ele possa perceber a complexidade envolvida na construção de cada ambiente. De forma geral, o funcionamento da aplicação, ilustrado

na Figura 4.2, deve ser a seguinte: 1. Inicialmente, o usuário solicita a captura da impressão digital através de um sensor específico que pode estar acoplado ao hardware ou externo ao mesmo; 2. Após captura da ID o usuário solicita a busca nas bases de dados; 3. A requisição é processada na grade computacional e o resultado é repassado à interface; 4. O resultado da solicitação é apresentado ao usuário através da interface.

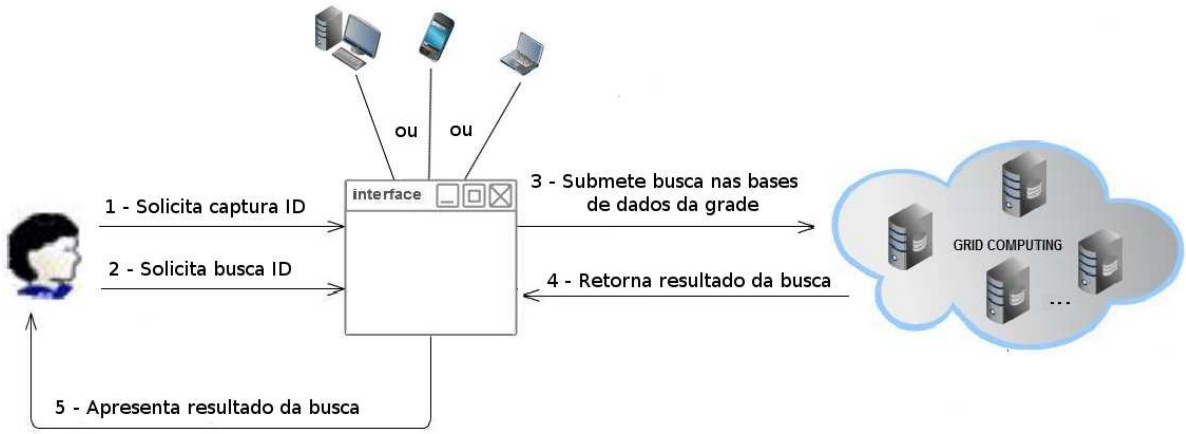

Figura 4.2. Funcionamento Geral da Aplicação

4.3.2 Escalonadores

Essa camada representa o modelo de escalonamento determinado pela arquitetura. Neste caso, o escalonamento das tarefas deverá ser executado em dois níveis, um metaescalonamento responsável pela distribuição das tarefas em ambiente global, e o escalonamento local, responsável pela distribuição das tarefas internamente em cada instituição participante.

Para o metaescalonamento, a ferramenta escolhida foi o GridWay por ter sido desenvolvido seguindo os padrões do Globus pelo consórcio Globus Alliance (ALLIANCE, 2008a). É uma ferramenta que foi desenvolvida especificamente para ser utilizada em ambiente de grade computacional, sendo assim, suas políticas de escalonamento são melhor adaptadas a esse tipo de ambiente. Possui, por exemplo, a capacidade de migração de tarefas em caso de perda de recursos (devido a falhas ou cancelamento de tarefas), o que garante uma maior disponibilidade do serviço. Como escalonador local para o cluster, o Torque (Seção 3.2.3) foi o escolhido por ser compatível com o WS-GRAM do Globus. Esta camada se comunica com os serviços de grade para determinar quais os recursos disponíveis na grade que executarão as tarefas.

4.3.3 Serviços de Grade

Para a escolha do middleware de grade para a comunicação e integração de recursos e dados foram analisados os seguintes *middlewares*: (1) OurGrid (OURGRID, 2009); (2) gLite (GLITE, 2009) e (3) Globus Toolkit 4 (GT4). O OurGrid é um middleware de grade aberta desenvolvido pela UFCG, para tornar mais rápida a execução de aplicações *bag-of-tasks* (aplicações cujas tarefas são independentes, como por exemplo, processamento de imagens). Entretanto, além de não abranger grade de dados, o compartilhamento dos recursos no mesmo está condicionado ao oferecimento de recursos (recebe mais recursos quem oferece mais) não atendendo, assim, a todas as premissas do projeto. O gLite é um *middleware* produzido como parte do projeto EGEE (EGEE, 2009) para fornecer um *framework* para construção de aplicações em grade que necessitam de poder computacional e armazenamento de recursos através da Internet. Os serviços de grade do gLite seguem uma arquitetura orientada a serviços, o que facilita a interoperabilidade entre os serviços de grade existentes. Apesar do gLite atender às premissas do projeto, a sua instalação é restrita a distribuição Scientific do Linux. Sendo assim, sua utilização foi descartada.

Por atender às premissas Descartados o OurGrid e gLite, o *Globus Toolkit* 4 (GT4) foi o *middleware* de grade definido para permitir a comunicação e integração de recursos e dados. Ele possui as ferramentas necessárias para prover segurança, gerenciamento de aplicações e movimentação de dados. Além disso, segue o padrão OGSA que é uma arquitetura de grades fundamentada na tecnologia de serviços web (*web services*), denominada *grid service* (Seção 3.1). Os seguintes componentes do GT4 foram utilizados:

• GSI: utilizado para prover a segurança do sistema em nível de transporte (*transport-level security*) onde toda informação trocada entre cliente e servidor é criptografada. Foi utilizada a entidade certificadora *SimpleCA* como provedor dos certificados digitais para os usuários e servidores. Cada máquina que faz parte da grade computacional possui um certificado digital, assim como, todo usuário precisa ter um certificado próprio (instalado em sua estação) para utilizar os serviços. Além disso, utiliza-se o *framework* de autenticação e autorização para fornecer segurança ao sistema. O mecanismo de autorização utiliza um arquivo de mapeamento de usuários, denominado *grid-mapfile* que correlaciona os usuários da grade (identificados pelos certificados) e usuários locais, respeitando as políticas de segurança locais de cada site. O GSI também possibilita autenticação em nível de serviços. Nesse caso, cada serviço pode ter o seu próprio *grid-mapfile* predominando em relação ao *grid-mapfile* do usuário. Esta abordagem é utilizada para definir quais usuários podem acessar determinadas funcionalidades.

• WS-GRAM: utilizada para fazer todo o gerenciamento relacionado a execução das tarefas, conforme descrito em 3.4. A tarefa executada pelo WS-GRAM é a execução do BOZORTH3 para comparação das minúcias.

• GridFTP: utilizado para efetuar toda a transferência dos dados da aplicação de maneira segura, robusta, rápida e eficiente. Os arquivos de minúcias são submetidos à comparação nos recursos da grade, definidos pelo metaescalonador, através do uso

do GridFTP. Para utilização dessa ferramenta não é necessário efetuar nenhum tipo de configuração adicional sendo distribuída junto com o Globus.

• OGSA-DAI: utilizado para administrar a distribuição das bases de dados. Nessa arquitetura, o OGSA-DAI contacta diretamente o servidor GridFTP para disponibilizar as informações a serem transferidas. Para a integração da base de dados relacional, foi utilizado o *Resource Group*. Este recurso agrupa um conjunto de recursos de dados, independente do SGBD, o que possibilita que uma única instrução SQL seja executada para todos os recursos de dados agrupados.

• CoG: utilizado para facilitar a criação do código Java para submissão das tarefas via WS-GRAM e para transferência de arquivo via GridFTP. As facilidades providas pelo CoG ao desenvolvedor estão detalhadas em 3.4.2.

4.3.4 Acesso a Dados

Essa camada representa as bases de dados que compõem a grade de dados federativa, bem como, o diretório onde estão armazenados os arquivos de minúcias utilizados no processo de comparação das impressões digitais. Nas bases de dados estão armazenadas todas as informações relacionadas às impressões digitais e aos indivíduos detentor da mesma.

Uma vez criado, o arquivo de minúcias não sofre alterações. Sendo assim, determinou-se que o armazenamento dessa informação seria num diretório de dados físico. Esse diretório contém todas as minúcias de todas as bases de dados que compõem a organização virtual. Com isso, elimina-se o tempo de acesso às bases de dados para compor os arquivos de minúcias, imprescindíveis para o funcionamento do BOZORTH3. A transferência das demais informações é feita através do GridFTP.

Para esta arquitetura não existe especificação de SGBD, apenas a modelagem das bases de dados devem ser padronizadas. O OGSA-DAI é o responsável pela integração dos dados disponibilizados, ele possui um conjunto de ferramentas que tornam possível a integração de informações oriundas de bases de dados de diferentes fabricantes e em diferentes localidades. Através do recurso *ResourceGroup*, do OGSA-DAI, é possível submeter um único comando de consulta SQL e obter os dados de diferentes origens que são agrupados de maneira transparente ao usuário.

CAPÍTULO 5

ESTUDO DE CASO: APLICAÇÃO PARA IDENTIFICAÇÃO CRIMINAL ATRAVÉS DO RECONHECIMENTO DA IMPRESSÃO DIGITAL USANDO GRADE COMPUTACIONAL

Este capítulo apresenta um estudo de caso para validação da arquitetura proposta no Capítulo 4. Este estudo de caso envolve a identificação criminal através da impressão digital, tomando como referência o processo atualmente utilizado no Estado da Bahia. Inicialmente será apresentada a descrição do problema que motivou o surgimento desse projeto, como a aplicação foi implementada e por fim, o modelo da distribuição dos dados nas bases de dados.

5.1 DESCRIÇÃO DO PROBLEMA

A identificação criminal no Brasil tem passado por um processo de automatização, onde alguns Estados, como é o caso da Bahia, têm investido na aquisição e implantação de sistemas AFIS para melhor combater a criminalidade e para registros civis, como apresentado em (SSP, 2009). Para que se obtenha informações criminais de todo o Brasil, o Estado deve fazer acesso a base criminal da Polícia Federal, situada em Brasília, onde essas informações estão centralizadas. Levando-se em consideração a quantidade de registros criminais no Estado da Bahia, em torno de 60.000 (dados de (SSP, 2007)), e projetando essa quantidade para os demais Estados (sem considerar os Estados com mais ou menos registros), nota-se que a base da Policia Federal armazena mais de um milhão de registros (dados que crescem de forma exponencial), requerendo, assim, uma máquina com capacidade de processamento muito grande para processar todos esses dados. Além disso, por estarem armazenados em um único local, os dados devem ser transferidos em determinados períodos, o que demonstra que as informações acessadas podem não ser as mais atuais.

A arquitetura proposta na Capítulo 4 mostra-se como alternativa a uma solução centralizada geograficamente, onde é necessário a utilização de computadores de alto desempenho para o processamento dos dados. A solução proposta provê autonomia dos dados entre os Estados Brasileiros, permitindo que sejam compartilhados de forma transparente, evitando investimento em máquinas muito especializadas e garantindo dados consistentes e em tempo real.

Para que a aplicação atenda aos requisitos da arquitetura proposta, é necessário que seja capaz de:

• Manipular Imagem da Impressão Digital: a imagem da impressão digital deve ser capturada através de um sensor específico. Em seguida, deve ser classificada e devem ser extraídas as minúcias, através dos módulos específicos do NFIS.

• Solicitar Busca: de acordo com a classe identificada na impressão capturada e nos filtros de dados definidos pelo usuário (que podem ser utilizados ou não), o sistema efetua a busca nas bases de dados distribuídas. Ao identificar registros relacionados à busca efetuada, o sistema deve efetuar a comparação através da execução do módulo específico do NFIS (BOZORTH3).

• Visualizar Suspeitos: apresenta os dados do(s) indivíduo(s) identificado(s) como suspeito(s). Essas informações são os dados civis do indivíduo.

• Visualizar Crimes: lista as informações sobre os crimes cometidos pelo(s) indivíduo(s) identificado(s) como suspeito(s).

5.2 CENÁRIO DA DISTRIBUIÇÃO DE DADOS

A Figura 5.1 apresenta um exemplo de um cenário da aplicação. Este cenário é composto por três *clusters*, distribuídos geograficamente, formando a OV. Cada cluster possui um nó principal (*HeadNode*) que, por sua vez, contém o serviço de grade, designado *GridService1*. Este, além de classificar e extrair as minúcias da impressão digital capturada, é responsável por distribuir as requisições entre as máquinas que possuem o *GridService2*. Este é responsável por efetuar a comparação das minúcias. Os dados estão armazenados nos SGBDs e a interface do cliente pode ser através de *desktop*, *notebook* ou PDA.

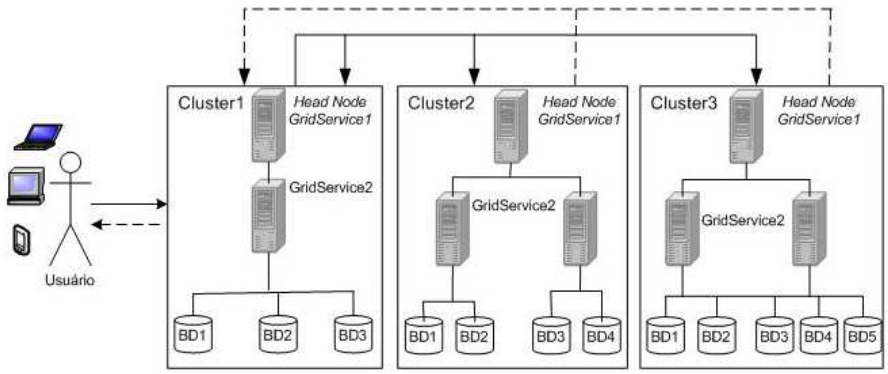

Figura 5.1. Exemplo de um Cenário de Distribuição de Dados

A aplicação visa prover autonomia dos dados a cada Estado Brasileiro, sendo assim, cada Secretaria de Segurança Pública (SSP) deve determinar a quantidade de bases de dados e máquinas que farão parte do *cluster* e que conterão os serviços, levando em consideração a quantidade de registros que estarão armazenados. A aplicação cliente contacta todos os serviços (*GridServices*) de cada Estado, porém isto é realizado de

forma transparente, como se o acesso fosse apenas à sua base local. Para que o acesso aos *GridServices* não seja seqüencial, é criada uma *thread* para contactar cada serviço. O resultado da solicitação (lista de suspeitos ou não) só será apresentada ao cliente quando todas as *threads* finalizarem.

Os seguintes passos são executados para funcionamento da aplicação e estão representados no diagrama de sequência da Figura 5.2:

1. O usuário solicita, através da interface, a captura da ID que é obtida através de um sensor específico.

2. A aplicação retorna sucesso ou falha na captura da ID

2.1 Se falhar, apresenta mensagem ao usuário e retorna ao passo 1.

2.2 Se sucesso, seguir para o passo 3.

3. Usuário solicita busca da ID, passando como parâmetro a ID capturada e filtros (opcionais).

4. A interface cliente, então, envia a imagem capturada para a aplicação que executa na máquina principal do *cluster* do Estado onde está sendo solicitada a busca, denominada *HeadNode*. Em seguida, solicita a sua classificação.

5. É efetuada a classificação da imagem através da execução do módulo PCASYS, do NFIS, retornando sucesso ou falha para a classificação. Se a falha estiver relacionada a um tipo incompatível de arquivo, não reconhecido pelo PCASYS, a aplicação cliente é informada para que seja possível enviar um novo arquivo de impressão digital. Em seguida,é efetuada a extração das minúcias da imagem através da execução do módulo MINDTCT, do NFIS, retornando sucesso ou falha para esse procedimento. Em caso de falha, o procedimento é re-executado, se a falha persistir uma mensagem de erro é retornada à aplicação informando que não será possível efetuar a busca.

6. Se a extração das minúcias retornar sucesso, o *GridService1* contactará os serviços do *GridService2*, espalhados pela grade, responsáveis pela comparação das minúcias.

7. O *GridService2* submete uma consulta (*query*) ao OGSA-DAI para que seja feito o agrupamento dos nomes dos arquivos de minúcias, armazenado em cada base que compõe os *clusters*, de acordo com a classe identificada e com os filtros, se aplicados.

8. A lista com os nomes dos arquivos de minúcias são retornadas para as máquinas que contêm o *GridService2*. Nesse momento, o metaescalanador Gridway é acionado para distribuir a execução do BOZORTH3 entre as máquinas que compõe a grade. Em seguida, o Torque responsabiliza-se por distribuir as tarefas entre as máquinas do *cluster*.

9. Feitas as comparações, a lista de suspeitos é enviada para o *HeadNode* que solicitou a comparação e este retorna à aplicação cliente.

10. A aplicação cliente solicita a visualização dos crimes do suspeito selecionado.

11. O *GridService2* vai buscar, então, na base onde as informações estão armazenadas,

retornando as informações para aplicação cliente.

5.3 IMPLEMENTAÇÃO

Foram desenvolvidos três serviços de grade: um serviço para criação e gerenciamento dos recursos *Web Services*, denominado AfisDCCFactoryService; e dois serviços que provêm as funcionalidades do processo de reconhecimento da impressão digital (classificação e extração das minúcias, e comparação), denominados AfisDCCService1 e AfisDCCService2, respectivamente. Java foi a linguagem utilizada para desenvolvimento desses serviços, por ser uma das linguagens suportadas pelo GT4.

O AfisDCCFactoryService é uma classe que implementa o *Web Service* que é utilizado como interface para integração com os recursos das *Factory* (responsáveis por gerenciar e instanciar os recursos). AfisDCCService1 e AfisDCCService2 são classes que implementam a instância *Web Service* que disponibiliza os principais métodos da aplicação. As operações disponibilizadas pelo *Web Service* às aplicações clientes são:

• *SendFingerResponse sendFinger (SendFinger complexType)*: responsável pelo envio da impressão digital coletada, pelo sensor, através de um vetor de bytes. O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).

• *ClassifierResponse classifier(Classifier complexType)*: permite que a aplicação cliente solicite a classificação da impressão digital enviada pela operação sendFinger(). O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).

• *Minutia_extractorResponse minutia_extractor (Minutia_extractor complexType)*: permite que a aplicação cliente solicite a extração das minúcias da impressão digital enviada pela operação sendFinger(). O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).

• *SendFilterResponse sendFilters (sendFilters complexType)*: permite que a aplicação cliente informe ao recurso algum tipo de filtro a ser aplicado à busca das minúcias. A utilização do filtro é opcional e envolve o tipo do dedo (polegar, indicador, etc.) e o sexo do indivíduo. O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).

• *MatchingResponse matching (Matching complexType)*: responsável por fazer a comparação das minúcias extraídas da impressão digital capturada, com as minúcias coletadas da base de dados, referentes à classe detectada e aplicando os filtros, se definidos. O retorno é uma *string* com o resultado da execução da operação (*SUCESS*, *NO SUCESS* ou mensagem específica do erro).

• *GetSuspectCrimesResponse getSuspectCrimes(GetSuspectCrimes complexType)*: retorna à aplicação cliente as informações relativas aos crimes dos suspeitos identificados.

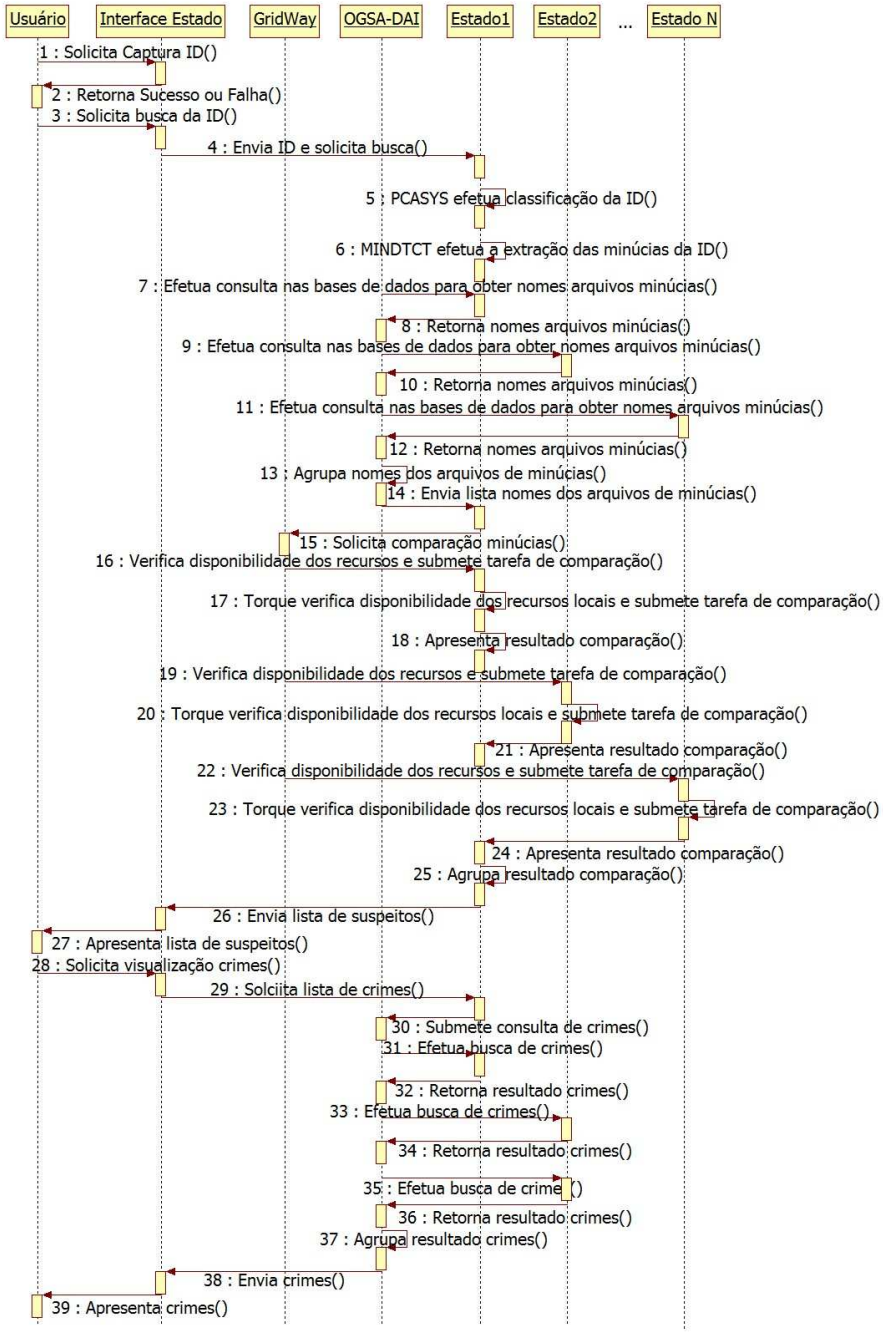

Figura 5.2. Diagrama de Sequencia do Funcionamento da Aplicação

Além das classes supracitadas, foram criadas as seguintes classes, que servem de suporte à aplicação: (i) AfisExecution: utilizada para invocar os pacotes PCASYS, MINDTCT e BOZORTH3 do NFIS, responsáveis, respectivamente, pela classificação da impressão digital, extração das minúcias e comparação das mesmas com as selecionadas das bases de dados; (ii) OgsaDaiClient: utilizada para servir de cliente ao serviço OGSA-DAI, possibilitando que a busca às bases de dados sejam feitas de forma transparente; (iii) SuspectModel, CrimeModel e FilterModel: classes utilizadas para retornar informações a respeito dos suspeitos, crimes e filtros respectivamente.

A Figura 5.3 demonstra o processo de interação entre os serviços da grade além da seqüência de execução, conforme listado abaixo:

1. Inicialmente o cliente solicita a criação do recurso ao AfisDCCFactoryService1;

2. AfisDCCFactoryService1 solicita a criação de um AfisDCCResource1 para o AfisDCCResourceHome1, este então, cria um AfisDCCResource1, adiciona este recurso para a sua lista de recursos e retorna uma chave única que identifica o recurso;

3. O cliente solicita ao AfisDCCService1 que a classificação e extração das minúcias da impressão digital sejam feitas em um determinado recurso;

4. O AfisDCCService1 utiliza o AfisDCCResourceHome1 para encontrar o recurso criado a partir da chave retornada pelo AfisDCCFactoryService1;

5. Ao encontrar o recurso, o AfisDCCService1 executa as operações de classificação e extração das minúcias no AfisDCCResource1;

6. O AfisDCCService1 solicita a criação do recurso ao AfisDCCFactoryService2;

7. AfisDCCFactoryService2 solicita a criação de um AfisDCCResource2 para o AfisDCCResourceHome2, este então, cria um AfisDCCResource2, adiciona este recurso para a sua lista de recursos e retorna uma chave única que identifica o recurso;

8. O AfisDCCService1 solicita ao AfisDCCService2 que a comparação das minúcias agrupadas das bases de dados sejam feitas em um determinado recurso;

9. O AfisDCCService2 utiliza o AfisDCCResourceHome2 para encontrar o recurso criado a partir da chave retornada pelo AfisDCCFactoryService2;

10. Ao encontrar o recurso, o AfisDCCService2 executa a comparação das minúcias no AfisDCCResource2 e retorna os suspeitos para o AfisDCCService1 que entrega ao cliente.

Devido a restrições de recursos e instalações que alguns dispositivos móveis possuem, como é o caso do PDA, foi criado um *Web Service* intermediário, utilizado para efetuar a comunicação com os serviços disponilizados pelo GT4. Para execução através do *Web Service* intermediário, foram criadas duas classes que tratam com a regra de identificação da aplicação. Essas classes são clsCrime e clsSuspeito. Nessas, existem métodos

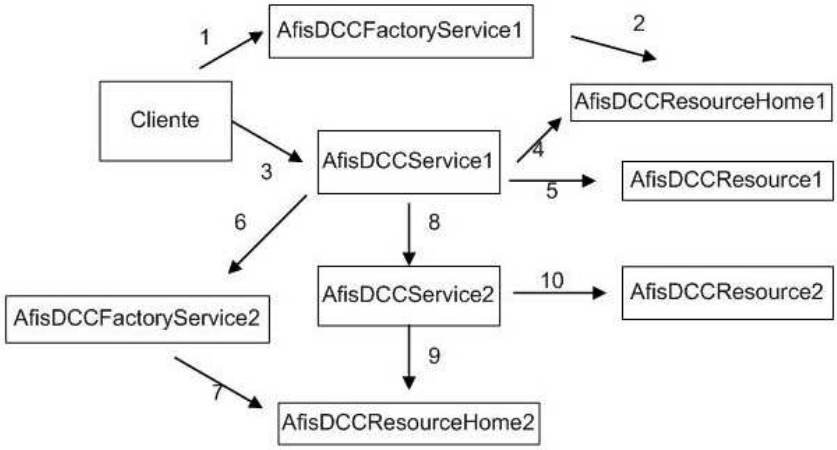

Figura 5.3. Interação entre os Serviços da Grade

para comunicação com o *Web Service* permitindo, assim, a comunicação remota com a aplicação da grade. Este *Web Service* é responsável por executar o cliente que se comunica e consome o AfisDCCService1 e por disponibilizar os serviços: (i) EnviarFiltro: para envio da imagem da impressão digital e filtros selecionados; (ii) BuscarSuspeitos: disponibiliza informações sobre os suspeitos; e (iii) BuscarCrimes: disponibiliza informações dos crimes cometidos pelos suspeitos identificados.

Foram desenvolvidas duas interfaces, uma para computadores pessoais e outra para PDA. As telas da interface PDA podem ser visualizadas nas Figuras 5.4 e 5.5. A Figura mostra a seguinte seqüência de telas: (i) tela de captura da impressão digital que pode ser feita através do sensor embutido no PDA ou através da busca de um arquivo da impressão digital previamente armazenado, (ii) tela de filtro onde podem ser selecionados filtros por dedo e/ou sexo, (iii) tela que lista os suspeitos encontrados com a comparação das minúcias, e (iv) tela que lista os crimes cometidos pelo suspeito selecionado.

A linguagem utilizada no desenvolvimento da aplicação PDA foi C-# (C-Sharp) que é compatível para esse dispositivo móvel. As telas foram desenvolvidas com base nos formulários Windows (*Windows Form*) disponibilizados pelo .NET *Compact Framework* que possui um conjunto específico de ferramentas desenvolvidas ou adaptadas para o desenvolvimento de aplicações para dispositivos móveis. Apesar da aplicação ter sido desenvolvida e testada tendo como foco a execução no modelo do PDA iPAQ HX2790, funcionará em outros modelos que tenham suporte ao .NET *Compact Framework Runtime*.

Para garantir segurança na comunicação entre a interface do PDA, através do *Web Service* intermediário, com os serviços da grade foi utilizado o protocolo SSL (*Secure Socket Layer*). Adicionalmente, foi necessário armazenar temporariamente, em arquivos

Figura 5.4. Interface da Aplicação do PDA - Telas de Captura da ID

Figura 5.5. Interface da Aplicação do PDA - Telas de Resultados

texto, as minúcias coletadas nas bases de dados para manter a compatibilidade com o processo de comparação de minúcias do NFIS. Uma vez que todos os pacotes PCASYS, MINDTCT e BOZORTH3 estão programados em C e assumem o uso de arquivos texto.

5.4 MODELO DA DISTRIBUIÇÃO DOS DADOS

A modelagem de banco de dados para esta aplicação é composta por tabelas que armazenam registros civis do indivíduo, informações sobre os crimes cometidos pelo mesmo, tabelas com as opções de filtro para aplicação e informações relacionadas à impressão digital. Esses dados encontram-se distribuídos entre as instituições participantes da grade. Cada instituição é responsável por administrar e manter os dados armazenados, bem como definir a distribuição dos dados entre os recursos.A Figura 5.6 ilustra a modelagem de dados que deve ser seguida por cada instituição para que seja possível acessar os dados disponibilizados na grade computacional.

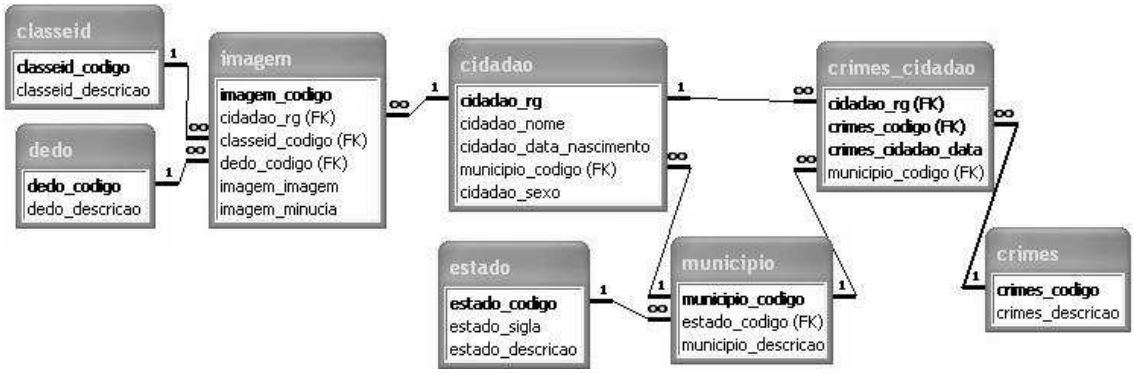

Figura 5.6. Modelagem do Banco de Dados

A tabela ***cidadao*** armazena os dados civis dos indivíduos e está relacionada com as tabelas ***crimes_cidadao*** e ***imagem***. A tabela ***crimes_cidadao*** compõe o relacionamento entre a tabela ***crimes*** e a tabela ***cidadao***, registrando assim, os crimes cometidos pelo ***cidadao***. A tabela ***imagem*** armazena as informações sobre a imagem da impressão digital, sendo registrada a imagem da impressão digital e nomes dos arquivos de minúcias que deve ser formado pela sigla do Estado seguido do numero do registro geral do indivíduo. Essa tabela relaciona-se com as tabelas ***classeid*** e ***dedo***, onde são informados o tipo de dedo e a classe que a impressão digital pertence. A tabela ***municipio*** está relacionada às tabelas ***cidadao*** e ***crimes_cidadao*** para que seja possível armazenar a naturalidade do cidadão e o local onde o crime foi cometido, respectivamente. Os Estados são identificados pela tabela ***estado*** relacionada com a tabela ***municipio***.

Testes preliminares, apresentados no Capítulo 6, demonstraram que o tempo de acesso às bases de dados para busca das minúcias e posterior criação dos seus arquivos era significativo. Sendo assim, foi definido que o procedimento de réplica dos arquivos de minúcias seria feito de forma *offline*, para que desta maneira, seja possível excluir o tempo de trans-

ferência dos dados entre as máquinas. Sendo assim, em cada *headnode* existirão todos os arquivos de minúcias de todos os Estados. Foi determinado esse procedimento ao levar em consideração o tamanho da base de dados a ser utilizada (na ordem de grandeza de 10^6) e o tamanho de um arquivo de minúcias (aproximadamente 4KB). Ao considerar uma rede de 100mbps, gastaria-se 40s só para tranferir os dados para as máquinas, o que poderia inviabilizar a solução proposta. Vale ressaltar que a replicação dos dados de minúcias entre as máquinas não irá infringir a confidencialidade das informações dos indivíduos, visto que os arquivos de minúcias só contêm informações relativas às localizações das mesmas nas impressões digitais. Para ter acesso às informações a quem pertence a impressão digital é necessário acessar as bases de dados fazendo referência ao nome do arquivo de minúcias. O diretório definido para armazenamento dos arquivos de minúcias em cada *head node* foi a pasta "id"criada no diretório "home".

Para que cada Estado tenha o repositório local de minúcias atualizado é feito o sincronismo dos dados a cada uma hora. Determinou-se essa periodicidade para sincronismo dos dados após analisar a estatística dos principais registros criminais (dados dos anos de 2008 e 2009) nos seguintes Estados Brasileiros: BA - aproximadamente 12 registros por hora (SSP-BA, 2010); RJ - aproximadamente 239 registros por hora (SSP-RJ, 2010); SP - aproximadamente 443 registros por hora (SSP-SP, 2010).

CAPÍTULO 6

RESULTADOS DOS EXPERIMENTOS

Este Capítulo descreve os testes realizados e os resultados obtidos para avaliar o protótipo desenvolvido com base na arquitetura proposta. Para isso, foram definidos alguns cenários, para que de forma comparativa fosse possível comprovar a viabilidade da solução apresentada. As seções a seguir apresentam os ambientes utilizados para efetuar os testes, os cenários de testes definidos com seus respectivos resultados e avaliação dos mesmos.

6.1 AMBIENTE DE TESTES

O ambiente de testes envolveu utilização de máquinas diversificadas, com hardwares heterogêneos e infraestruturas de rede diferentes. A aplicação desenvolvida foi considerada como de tempo real brando (*soft real time systems*) onde o tempo de resposta da aplicação (*deadline*) não é crítico, porém desejável. Foi definido que o tempo de resposta da aplicação deve ser de dez segundos, que é o tempo limite determinado em (NIELSEN, 2007) para manter a atenção do usuário focado na aplicação. Caracteiza-se, também, como computação com uso intensivo de dados (*data intensive*) por ser uma aplicação distribuída que manipula grande quantidade de dados remotamente, exigindo assim, um grau elevado de computação e comunicação entre os processos.

A quantidade total de registros utilizados foi de 100.000 (cem mil registros) divididos nas classes: arco, presilha interna, presilha externa e verticilo. As imagens de impressão digital utilizada nos testes foram obtidas de uma base de dados pública disponibilizada pelo Nist (NIST, 2008). Essas imagens foram disponibilizadas junto com os pacotes do NFIS e a quantidade total foi de 2700 (dois mil e setecentas) imagens. Foram reproduzidas cópias dessas imagens para alcançar os cem mil registros.

Nesses testes foi considerada uma base de dados monodactilar, onde por indivíduo foram cadastrados apenas a impressão digital de um único dedo. A repetição dos testes não apresentou uma variância significativa, sendo assim, cada teste foi repetido três vezes e o resultado considerado foi a média dos tempos dessas repetições. Não foram simulados durante os testes nenhum tipo de problema no ambiente, seja falha de hardware, software ou rede. Sendo assim, nenhum método de tolerância a falhas foi testado, considerando o ambiente isento de falhas. As máquinas utilizadas para submissão das tarefas e armazenamento dos dados estão relacionadas na Figura 6.1. Os testes foram realizados utilizando equipamentos geograficamente distribuídos da seguinte maneira:

• Cluster montado no DCC (Departamento de Ciência da Computação) da UFBA composto por quatro máquinas AMD Opteron 2GHz com 2GB de memória RAM, com

processadores de 64 bits com dois núcleos (*Dual Core*) cada, totalizando oito núcleos. O sistema operacional desse ambiente é o Linux com a distribuição Debian;

• Cluster montado no CPGG (Centro de Pesquisa em Geofísica e Geologia) da UFBA, composto por doze máquinas Intel Xeon de 2.5GHz com 16GB de memória RAM, com processadores de 64 bits com oito núcleos (*Quad Dual Core*) cada, totalizando 96 núcleos. O sistema operacional desse ambiente é o Linux com a distribuição Centos;

• Cluster montado na UAB (*Universitat Autònoma de Barcelona*) em Barcelona, composto por oito máquinas Intel Pentium IV de 2.8GHz com 512MB de memória RAM, com processadores de 32 bits, totalizando oito núcleos. O sistema operacional é Linux com a distribuição Fedora.

UFBA – DCC			
Quantidade Máquinas	**Processador**	**Memória RAM**	**Sistema Operacional**
4	AMD Opteron 2GHz Dual Core 64 bits	2GB	Linux (Debian)
UFBA – CPGG			
Quantidade Máquinas	**Processador**	**Memória RAM**	**Sistema Operacional**
12	Intel Xeon 2.5GHz Quad Dual Core 64 bits	16GB	Linux (Centos)
UAB – Barcelona			
Quantidade Máquinas	**Processador**	**Memória RAM**	**Sistema Operacional**
8	Intel Pentium IV 2.8GHz 32 bits	512MB	Linux (Fedora)

Figura 6.1. Máquinas Utilizadas nos Testes

6.2 CENÁRIO DE TESTES E AVALIAÇÃO DOS RESULTADOS

A solução proposta, nesse trabalho, foi adotada depois de testes realizados numa arquitetura onde apenas os dados eram distribuídos e as tarefas (classificação, extração de minúcias e comparação) eram centralizadas em uma única máquina, conforme apresentado em (BARRETTO et al., 2008) e resultados dos testes na Seção 6.2.1. Os testes envolvendo a arquitetura proposta neste trabalho estão detalhados na Seção 6.2.2.

6.2.1 Testes Preliminares

Para execução dos testes nessa fase, foram utilizadas as máquinas do cluster do DCC da UFBA. O número total de registros utilizados foi de 2700 (dois mil e setessentos) distribuídos entre as classes arco, presilha interna, presilha externa e verticilo. Foram efetuados testes envolvendo o armazenamento e busca em uma única máquina e dis-

tribuídos, em igual quantidade de registros, nas máquinas do cluster. Foram utilizados, também, filtros de pesquisa por classe e sexo e foram utilizadas interfaces de usuário através de PDA e *desktop*. A aplicacao desktop foi executada via interconexão Ethernet conectado a 1000Mbps e a aplicação do PDA utilizou conexão *wireless*.

Foram utilizados quatro cenários de testes:

• Cenário 1: os dados estão armazenados em uma única máquina, possuindo um total de 2700 registros. Foram efetuadas buscas através das interfaces PDA e Desktop. Inicialmente foram efetuadas buscas apenas utilizando a interface do PDA e do desktop separadamente. Em seguida, foram submetidas as buscas utilizando as duas interfaces simultaneamente (identificados no cenário 1 da Figura 6.2 por "PDA Compartilhado"e "Desktop Compartilhado", respectivamente). Neste cenário utilizou-se o filtro apenas pela classe da impressão digital.

• Cenário 2: difere-se do cenário 1 apenas pela inclusão do filtro por sexo. Inicialmente foram efetuadas buscas apenas utilizando a interface do PDA e do desktop separadamente, e em seguida com as duas interfaces ao mesmo tempo (identificados no cenário 2 da Figura 6.2 por "PDA Compartilhado"e "Desktop Compartilhado", respectivamente).

• Cenário 3: dados armazenados em quatro máquinas com 675 registros cada. Foram efetuadas buscas através das interfaces PDA e Desktop. Inicialmente foram efetuadas buscas apenas utilizando a interface do PDA e do desktop separadamente, e em seguida com as duas interfaces ao mesmo tempo (identificados no cenário 3 da Figura 6.3 por "PDA Compartilhado"e "Desktop Compartilhado", respectivamente). Neste cenário utilizou-se o filtro apenas pela classe da impressão digital.

• Cenário 4: difere-se do cenário 3 apenas pela inclusão do filtro por sexo. Inicialmente foram efetuadas buscas apenas utilizando a interface do PDA e do desktop separadamente, e em seguida com as duas interfaces ao mesmo tempo (identificados no cenário 4 da Figura 6.3 por "PDA Compartilhado"e "Desktop Compartilhado", respectivamente).

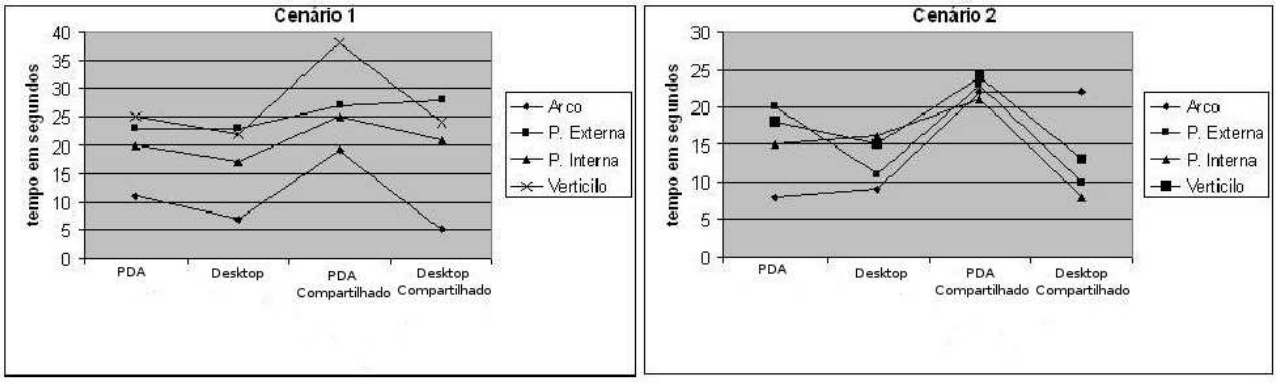

Figura 6.2. Resultados dos Testes dos Cenários 1 e 2

Ao analisar os quatro cenários de testes apresentados foi possível constatar que em

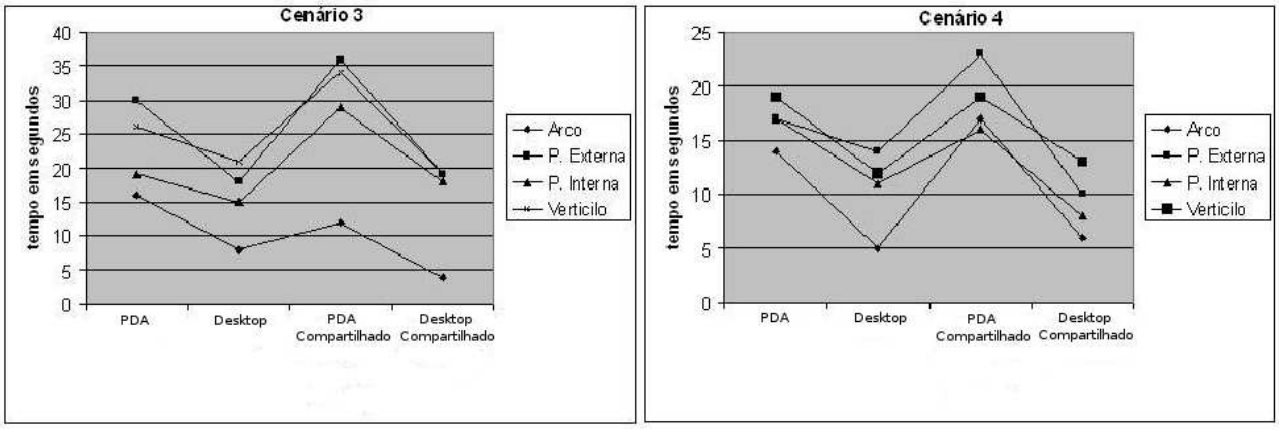

Figura 6.3. Resultados dos Testes dos Cenários 3 e 4

geral: (i) o tempo de resposta da consulta com o acesso feito pelo desktop foi menor do que pelo PDA. Esse resultado já era esperado visto que o PDA precisa fazer acesso ao *Web Service* intermediário; (ii) em todos os cenários o tempo de resposta da consulta foi inferior quando executado somente pelo PDA e pelo desktop em momentos distintos em relação aos dois simultaneamente; (iii) o tempo de resposta da consulta foi menor quando aplicados filtros de classe e sexo em relação a utilização do filtro apenas por classe. Entretanto, observou-se que o tempo de resposta da aplicacao ainda não é satisfatório, visto que o ambiente ao qual se propõe a arquitetura possui base de dados maiores do que as utilizadas. Além disso, estarão distribuídos geograficamente, ao contrário dos testes realizados em uma rede local.

Um dos requisitos críticos da aplicação é a necessidade de agrupar os dados das minúcias em um sistema de arquivo. Para cada registro compatível com a consulta aplicada, um arquivo de minúcias deve ser criado. Este procedimento é necessário visto que o sistema de comparação de minúcias, BOZORTH3, efetua o procedimento da comparação através do acesso a esses arquivos. O tempo gasto na serialização das minúcias para arquivos texto é bastante significativo e o processo de comparação em muitos arquivos, numa única máquina, consome bastante tempo, apresentando, assim, a necessidade de paralelizar esse serviço. O tempo de comparação de minúcias representa a maior parcela da computação, apresentando a necessidade de utilizar a distribuição do processamento do BOZORTH3 para aproveitar a localidade dos dados já distribuídos e diminuir o tempo total de resposta. Além disso, os resultados indicam a necessidade de testes em maior escala para que seja possível observar como a infraestrutura da grade computacional se comporta. Estes testes serão apresentados na Seção 6.2.2.

6.2.2 Testes na Arquitetura Proposta

Os cenários de 1 a 3 envolveram busca de dados centralizados em uma única máquina, busca de dados distribuídos entre as máquinas dos *clusters* e busca de dados distribuídos na grade computacional composta por três CE's (*Computer Element*) representados pelas máquinas principais de cada *cluster*. Os tempos de respostas foram medidos em segundos. Os testes realizados não envolveram a utilização de filtros para que fosse possível obter o melhor tempo de resposta no pior cenário (com mais registros). Para os testes realizados no cluster foram utilizados, para auxiliar no escalonamento global e local, balanceamento de carga estático, onde a distribuição dos dados foi previamente definida de acordo com a quantidade de processadores disponíveis. Nestes cenários não foram utilizadas as interfaces PDA e Desktop, sendo a aplicação executada através de linha de comando. Os cenários utilizados foram os seguintes:

Cenário 1 - Centralizado

Neste cenário os testes foram executados de forma centralizada, onde os 10.000 (dez mil) registros foram armazenados em uma única máquina. Foram efetuados testes nas máquinas principais de cada cluster citado na Seção 6.1. A aplicação foi executada via interconexão Ethernet conectado a 1000Mbps.

Nos testes efetuados foram submetidos de 1 a 12 processos para as máquinas.O número de processos a serem executados foi definido com base no resultado dos testes efetuados, onde notou-se que a partir de oito processos não havia diferença significativa na redução do tempo. Conforme detalhado abaixo. Para medir o tempo total de execução foram considerados os seguintes elementos: (i) tempo de envio do arquivo de minúcias da impressão digital a ser comparada para a máquina onde a tarefa será executada; (ii) tempo de busca na base de dados e carga de dados nos arquivos para comparação; (iii) tempo de envio dos filtros (quando não utilizados esse valor é igual a zero); (iv) tempo do processo de comparação das minúcias; (v) tempo para obtenção das informações dos suspeitos.

O resultado do cenário 1, apresentado na Figura 6.4, demonstra que esse tipo de aplicação apesar de manipular dados não faz uso intenso de instruções de ponto flutuante. Isto pode ser observado no gráfico onde os resultados excutados na máquina da UAB que possui capacidade inferior à do DCC, teve um tempo inferior, por dispor de capacidade para executar duas linhas de execução por processador através da característica *Hyperthread*. Foi possível constatar, também, que ao aumentar o número de processos em execução nas máquinas, só reduz o tempo de resposta até uma determinada quantidade de processos, levando em consideração o desempenho dos núcleos das máquinas. Nesse caso, o melhor desempenho foi da máquina do CPGG que possue 8 núcleos. A partir de 8 processos em execução simultânea, não ocorreu redução do tempo de resposta da aplicação. O mesmo ocorre para as demais máquinas, sendo que o tempo de resposta não teve redução considerável. Isso ocorre porque os processos começam a competir pelo

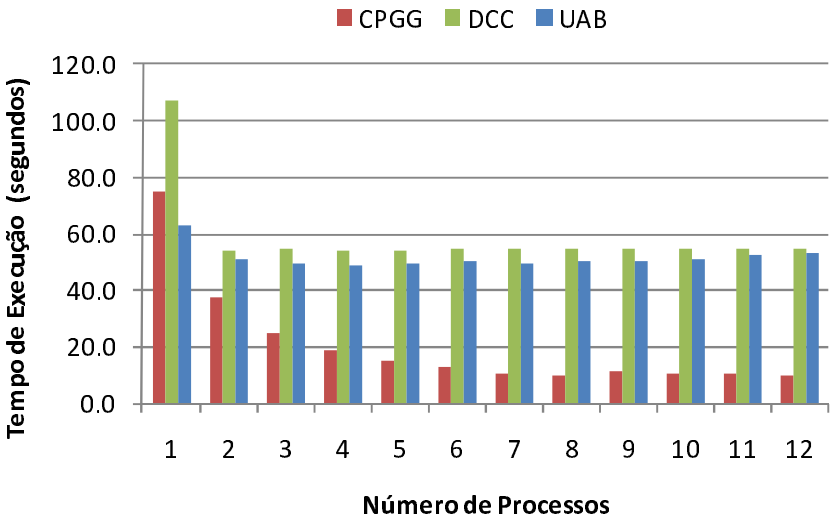

Figura 6.4. Resultado Testes Cenário 1

uso da máquina. Neste caso, a solução ideal seria aumentar a quantidade de núcleos da máquina. Porém, as informações armazenadas têm um crescimento que pode ser diário o que iria requerer que constantemente se investisse em máquinas mais potentes. Uma outra observação importante é que ao executar as tarefas nas máquinas do DCC e da UAB de forma centralizada não se consegue obter o tempo de resposta determinado como ideal para a aplicação (10s).

Cenário 2 - Cluster

Neste cenário os testes foram efetuados nas máquinas de cada cluster. Foi utilizado o balanceamento de carga estático, onde as tarefas foram distribuídas proporcionalmente nas máquinas existentes nos clusters do DCC na UFBA, do CPGG na UFBA e da UAB em Barcelona. Para medir o tempo total de execução foram considerados os mesmos elementos do cenário 1, acrescentando o tempo de escalonamento local, necessário para distribuir as tarefas nas máquinas do cluster.

Para execução no cluster foi determinado que o procedimento de réplica dos registros de minúcias seria feito de forma *offline*, para que desta maneira, fosse possível excluir o tempo de transferência dos dados entre as máquinas, conforme detalhado em 5.4. Os resultados dos testes, deste cenário, podem ser observados nas Figuras 6.5, 6.6 e 6.7

Os resultados referentes aos testes executados nos clusters DCC e UAB apresentados nas figuras 6.5 e 6.6, respectivamente, mostram que não existe grande impacto da influência no acesso aos arquivos de minúcias. Entretanto, no caso dos resultados apresentados no cluster CPGG se constata que a concorrência no acesso aos arquivos de minúcias impede a utilização da total capacidade de processamento do cluster, conseguindo *speed up* máximo de 48.2 em relação ao limite teórico de 96.

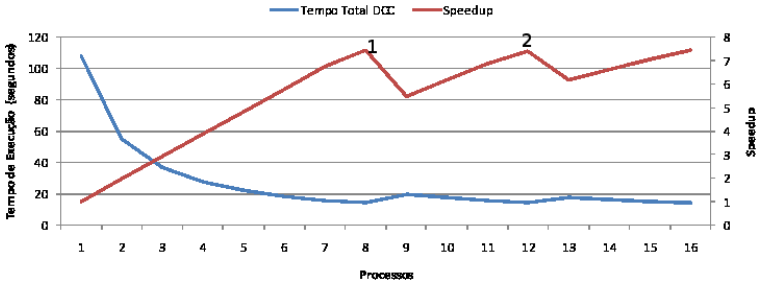

Figura 6.5. Resultado Testes Cenário 2 - DCC-UFBA

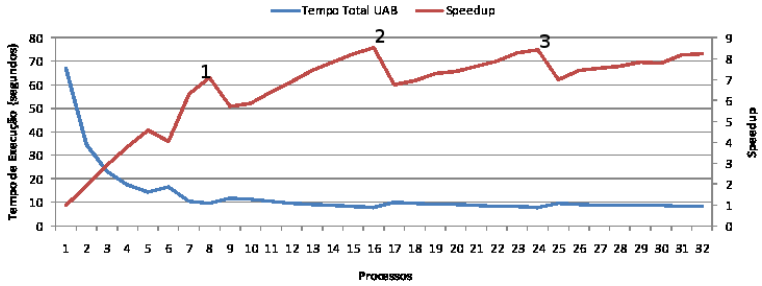

Figura 6.6. Resultado Testes Cenário 2 - UAB

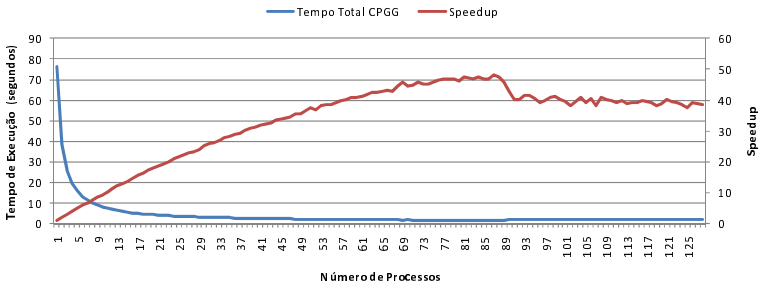

Figura 6.7. Resultado Testes Cenário 2 - CPGG-UFBA

Além disso, pode-se observar que os testes efetuados no cluster do DCC não alcançaram o tempo de resposta determinado como ideal para aplicação. Apesar do cluster da UAB ter a mesma quantidade de núcleos em relação ao do DCC, o tempo de resposta no primeiro foi inferior em relação ao segundo, devido a velocidade de processamento (requerido pela aplicação) superior ao DCC. Com isso, foi possível atingir o *speed up* máximo de 8 e tempo de resposta inferior a 10s.

Os picos representados por números nas Figuras 6.5 e 6.6 provavelmente é decorrente da interferência entre os processos competindo pela CPU. Visto que as máquinas utilizadas nos testes não são dedicadas excluivamente ao uso dessa aplicação, ocorrendo assim, disputa pelo uso local e/ou por outras aplicações. Bem como, o tipo de balanceamento de carga utilizado (estático) que não permite a distribuição de forma equilibrada conforme disponibilidade dos recursos.

Cenário 3 - Grade Computacional

Neste cenário, os testes foram executados utilizando todas as máquinas que compõem a grade computacional (cluster do DCC da UFBA, do CPGG da UFBA e da UAB de Barcelona). O procedimento de réplicas dos registros de minúcias também foi *offline*. Neste cenário, esse procedimento é ainda mais importante, visto que utiliza-se a Internet como meio transmissor e as latências entre as trocas de mensagens é bastante superior a de uma rede local. Segundo (COULOURIS; DOLLIMORE; KINDBERG, 2007), as requisições transmitidas pela Internet são aproximadamente 100 vezes mais lentos do que nas redes locais. Como ficou constatado que seriam necessários 40s para transmitir 100.000 registros numa rede local, seguindo a definição de Coulouris, seriam necessários 4000s (em torno de 1h) para tranferir na Internet.

Para medir o tempo total de execução foram considerados os mesmos elementos do cenário 2, acrescentando o tempo de escalonamento global, necessário para distribuir as tarefas nos nodos da grade. O balanceamento de carga utilizado foi dinâmico utilizando o metaescalonador GridWay. Sendo assim, as tarefas são submetidas cada vez em tamanho menores para que as máquinas com melhor desempenho executem mais tarefas, equilibrando, assim, o tempo de resposta das máquinas com menor desempenho. Dessa maneira, as máquinas com melhor desempenho não ficam esperando a execução das de menor desempenho finalizar para que possa executar novas tarefas.

Foram gerados três tipos de cenários considerando a divisão de carga e localização física dos arquivos de minúcias, e podem ser analisados na Figura 6.8:

• No primeiro cenário dividiu-se a carga igualmente entre os CE's disponíveis na grade. Para esse caso, os arquivos de minúcias foram disponibilizados na rede compartilhada do cluster. Observou-se que o tempo de resposta só foi satisfatório na CE com maior capacidade de processamento.

• No segundo cenário, a divisão de carga foi proporcional a quantidade de núcleos

disponíveis de cada cluster e os arquivos de minúcias estavam localizados na rede compartilhada. O tempo de resposta nesse cenário foi inferior ao primeiro, porém, não foi ainda o ideal. Isto deve-se ao tempo utilizado para acessar os arquivos de minúcias na rede.

• O terceiro cenário considera a divisão proporcional ao número de núcleos e replicação da base em cada disco local do cluster de cada máquina componentes dos CE's. Os resultados obtidos nesse cenário comprovam a viabilidade da arquitetura proposta para esse tipo de aplicação. Pode-se observar que o tempo total de resposta, representado pela barra azul do gráfico da Figura 6.8, foi de 14.1s. Apesar de não corresponder aos 10s definido como o tempo ideal para essa aplicação, o tempo de resposta ficou bem próximo do tempo ideal. Para diminuir esse tempo pode-se aumentar o número de máquinas que irão efetuar o processamento. Além disso, como esses testes não envolveram nenhum tipo de filtro, conclui-se que o tempo de resposta da aplicação reduzirá aproximadamente em 12% ao se utilizar o filtro apenas pela classe, por exemplo.

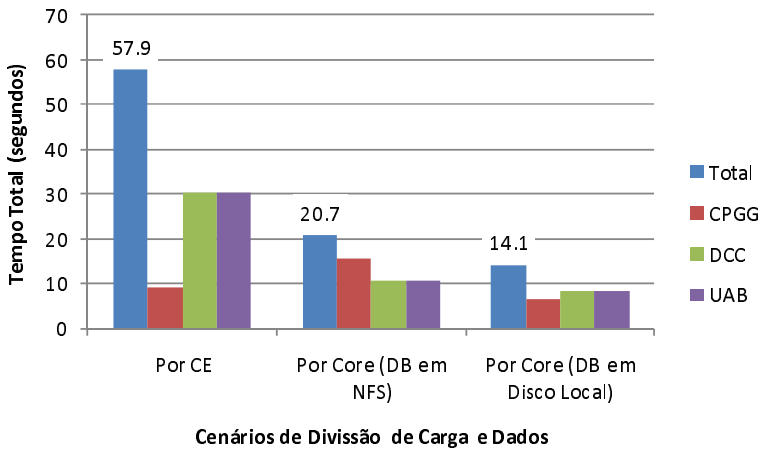

Figura 6.8. Resultado Testes Cenário 3

CAPÍTULO 7

CONCLUSÃO

A identificação automatizada de indivíduos através da impressão digital tem sido cada vez mais utilizada em diversas áreas. Para a área criminal este tipo de tecnologia mostra-se bastante eficiente, visto que o procedimento para identificação em uma base de grandes proporções requer bastante tempo de execução e, em algumas situações, torna-se inviável, caso não se tenha, por exemplo, uma lista de possíveis suspeitos. Esse tipo de aplicação envolve o armazenamento de muitas informações; além da imagem da impressão digital e das informações das minúcias, utilizadas no processo da comparação, é necessário armazenar informações civis do indivíduo, necessárias para sua identificação perante a sociedade, e as informações dos delitos cometidos. Essas informações são responsáveis por gerar grandes quantidades de dados, visto que os registros são feitos diariamente fazendo com que a base de dados tenha crescimento constante, principalmente ao se considerar informações criminais no âmbito nacional. Para se obter informações de criminosos no âmbito nacional é necessário buscar informações nas bases de dados de todos os Estados. Esse procedimento necessita de integração e processamento de dados em larga escala.

Diante desses fatos, acreditamos que a aplicação da tecnologia de grades computacionais no cenário de identificação criminal através da impressão digital, em nível nacional, é uma alternativa inovadora e desafiadora do ponto de vista da pesquisa e desenvolvimento. O uso dessa tecnologia envolve a complexidade de se administrar, integrar, compartilhar e processar informações em larga escala através do compartilhamento de recursos computacionais num ambiente geograficamente distribuído.

Este trabalho apresentou um estudo das tecnologias de impressão digital e de grades computacionais e mostrou como a utilização da infraestrutura de grades computacionais pode auxiliar no desempenho dos sistemas de reconhecimento através da impressão digital, reduzindo o tempo de resposta desse tipo de aplicação, bem como, provendo maior disponibilidade e independência dos dados distribuídos geograficamente. Para isso, foi apresentada uma arquitetura divida em camadas funcionais e independentes que se relacionam para prover o serviço, mas que, ao mesmo tempo, mantêm suas individualidades, o que permite maior flexibilidade e escalabilidade sem impactar nos demais componentes. A integração dos dados é provida pelo uso do OGSA-DAI, que permite que uma única submissão de consulta seja executada para busca de informações armazenadas em bases de dados de diferentes fabricantes e em diferentes localidades.

A criação da arquitetura teve como motivação prática o levantamento do processo de identificação criminal utilizada no Estado da Bahia. Nesse contexto, para se identificar um delito é necessário ter suspeitos; caso contrário, torna-se inviável a identificação, haja vista que o processo de identificação ainda é manual, através do uso de lentes de

aumento por um perito e a quantidade de registros criminais (em papel) era em torno de 60.000 registros (dados de 2004 (SSP, 2007)). Além disso, para se obter informações sobre outros delitos cometidos em outras localidades (mais especificamente, em outros Estados) é necessário fazer acesso à base de dados da Polícia Federal situada em Brasília.

Para validar a arquitetura foi desenvolvida uma aplicação para identificação criminal de indivíduos através da impressão digital. Testes envolvendo cenários de busca centralizada, em *cluster* e na grade foram realizados. Uma análise comparativa dos resultados dos testes efetuados comprovaram que o uso de grades computacionais é uma alternativa viável para esse tipo de aplicação, apresentando uma redução no tempo de resposta em torno de 20% em relação a solução centralizada. Além disso, mostrou-se que o uso de grades possibilita escalabilidade e independência dos dados entre as instituições envolvidas, permitindo o compartilhamento de recursos heterogêneos e com isso uma grande redução de custos para implantação de um sistema nacional de identificação criminal. Os resultados obtidos, considerando a divisão da carga de trabalho entre os recursos disponíveis na grade, apresentaram tempos aceitáveis para uma aplicação interativa, apesar da sobrecarga causada pelos softwares de gerenciamento da grade.

O estudo produzido por essa Dissertação contém informações que indicam como implementar ambientes de grade computacional para integração e processamento de dados distribuídos. A arquitetura desenvolvida em camadas funcionais permite que esse modelo possa ser ajustado e adaptado para atender a diferentes necessidades que não envolvam necessariamente o processo de reconhecimento através da impressão digital.

Durante o desenvolvimento desse trabalho foi publicado o seguinte artigo:

BARRETTO, J. et al. Uma arquitetura para integraçã de impressão digital através de uma grade computacional. 26 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) - VI Workshop on Grid Computing and Applications (WCGA), Rio de Janeiro - RJ, 2008. p. 73-84.

7.1 TRABALHOS FUTUROS

O acesso à aplicação desenvolvida nessa Dissertação pode ser feita através de uma interface desktop ou para dispositivos móveis do tipo PDA permitindo o uso da aplicação em diferentes plataformas. Com o objetivo de abranger qualquer plataforma que tenha acesso a Internet sugerimos a criação de uma interface única através de um portal Web. Para isso, será necessário um estudo mais aprofundado a fim de desenvolver uma SDK (kit para desenvolvimento de software) que seja compatível com diversos tipos de sensores para que a captura da impressão digital, usada como dado de entrada principal para o sistema, seja reconhecida pela aplicação independente do sensor utilizado.

Um dos requisitos para execução dos módulos MINDTCT (detectar as minúcias) e BOZORTH3 (comparar as minúcias) é que as informações estejam agrupadas em arquivos. Devido a essa restrição, determinamos que o armazenamento dos arquivos de minúcias seria local ao invés de ser armazenada no banco de dados, tendo neste apenas as referências

dos nomes dos arquivos. Como trabalho futuro, sugerimos a alteração dos códigos fonte dos pacotes MINDTCT e BOZORTH3 para que a sua execução possa ser feita com dados obtidos de uma base de dados, ao invés da utilização de arquivos. Dessa maneira, elimina-se a necessidade de agrupar as minúcias em arquivos, mantendo-as na base de dados, eliminando o processo de sincronização dos arquivos de minúcias em cada localidade para garantir a consistência dos dados consultados.

Uma das principais dificuldades no desenvolvimento desse projeto foi a obtenção de imagens de impressões digitais para os testes. Foram utilizadas as imagens disponibilizadas com o NFIS (*NIST Fingerprint Image Software*). Todas as imagens utilizadas foram de boa qualidade, inviabilizando, assim, a avaliação dos pacotes do NFIS para impressões digitais de baixa qualidade (aquelas obtidas por scanner, por exemplo) e impressões digitais latentes (colhidas no local do crime, por exemplo). Como trabalhos futuros pretende-se estender os testes para o reconhecimento desse tipo de imagem.

A base de dados montada para os experimentos consistiu de imagens de impressão digital de apenas um dedo, denominada monodactilar. Sugerimos aquisição de uma base de dados composta pelas imagens das impressões digitais dos dez dedos de cada indivíduo - base decdactilar para que novos testes possam ser submetidos e o comportamento do sistema possa ser avaliado nesse contexto que estará mais próximo de uma situação real.

Os testes foram executados em bases de dados com uma quantidade total de 100.000 registros. Sugerimos a execução desses mesmos testes em bases de dados com quantidade superior de registros (por volta de 1.000.000) para melhor avaliar a escalabilidade da aplicação além de estar mais próximo da situação real levando-se em consideração uma base de dados nacional de registros criminais.

Consideramos para os testes a ausência de falhas, não sendo portanto, avaliado o comportamento do sistema na presença de falhas. Como trabalhos futuros pretende-se efetuar um novo conjunto de testes com simulações dos seguintes tipos de falhas: parada de um dos nós da grade, dados perdidos durante a transmissão, além de outros. Desta forma, pretende-se analisar o desempenho da aplicação num ambiente mais próximo do real e por conseguinte, promover melhorias nos tempos de resposta obtidos.

REFERÊNCIAS

ALLEMAND, J. N. C. *Serviço Baseado em Semântica para Descoberta de Recursos em Grade Computacional*. Dissertação (Mestrado) — Universidade de Brasília - Instituto de Ciências Exatas, Brasília, Dezembro 2006.

ALLIANCE. *The Globus Alliance*. 2008. Acesso em janeiro de 2008. Disponível em: <http://www.globus.org/alliance/>.

ALLIANCE. *GT 4.0 Execution Management*. 2008. Acesso em janeiro de 2008. Disponível em: <http://www.globus.org/toolkit/docs/4.0/execution/>.

ANTONIOLETTI, M. et al. *The Design and Implementation of Grid Database Services in OGSA-DAI*. Fevereiro 2005. 357-376 p. Concurrency and Computation: Practice and Experience.

ANTONIOLETTI, M. et al. Ogsa-dai 3.0 - the what's and the why's. In: . [S.l.]: Proceedings of the UK e-Science All Hands Meeting 2007, 2007.

BAKER, M.; BUYYA, R.; LAFORENZA, D. Grids and grid technologies for wide-area distributed computing. *Software Practice and Experience*, John Wiley & Sons, Inc., Nova York, v. 32, n. 15, p. 1437–1466, 2002. ISSN 0038-0644.

BARRETO, M. E.; NAVAUX, P. O. A. Um modelo para a concepção de grades computacionais baseadas em clusters. In: WORKSHOP DO GRUPO DE PROCESSAMENTO PARALELO E DISTRIBUíDO - I WSGPPD. Porto Alegre: Instituto de Informática - UFRGS, 2003. v. 3, p. 163–169.

BARRETTO, J. et al. Uma arquitetura para integração de impressão digital através de uma grade computacional. In: . Rio de Janeiro - RJ: 26º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) - VI Workshop on Grid Computing and Applications (WCGA), 2008.

BERRY, D.; LUNIEWSKI, A.; ANTONIOLETTI, M. *OGSA Data Architecture*. Dezembro 2007. Global Grid Forum. Disponível em: <http://www.ogf.org/documents/GFD.121.pdf>.

BOOTH, D. et al. *Web services Architecture*. Fevereiro 2004. W3C Web Services Architecture Working Group. Disponível em: <http://www.w3.org/TR/ws-arch/wsa.pdf>.

CIRNE, W. et al. Running bag-of-tasks applications on computational grids: The mygrid approach. *Parallel Processing, International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 407, 2003. ISSN 0190-3918.

CIRNE, W.; SANTOS, N. Grids computacionais: Da computação de alto desempenho a serviços sob demanda. In: . [S.l.]: Mini-curso no 23 Simpósio Brasileiro de Redes de Computadores, 2005.

COG. *Java CoG Kit*. 2000. Acesso em outubro de 2009. Disponível em: <http://www.globus.org/cog/java/>.

COLVERO, T. A.; DANTAS, M. A. R.; CUNHA, D. P. da. *Ambientes de Custers e Grids Computacionais: Características, Facilidades e Desafios*. 2005. Anais do I Congresso Sul Catarinense de Computação. Disponível em: <http://www.dcc.unesc.net/sulcomp/05/artigos.htm>.

CONDOR. *Condor High Throughput Computing*. 2009. Acesso em agosto de 2009. Disponível em: <http://www.cs.wisc.edu/condor/>.

CONDORG. *Condor High Throughput Computing*. 2009. Acesso em agosto de 2009. Disponível em: <http://www.cs.wisc.edu/condor/condorg/>.

COSTA, S. M. F. *Classificação e Verificação de Impressões Digitais*. Dissertação (Mestrado) — Universidade de São Paulo - USP, 2001.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. *Sistemas Distribuídos: Conceitos e Projeto*. quarta. Porto Alegre: Bookman, 2007. Tradução: João Tortello.

EGEE. *Enabling Grids for E-sciencE*. 2009. Acesso em novembro de 2009. Disponível em: <http://www.eu-egee.org/>.

EUDATAGRID. *The DataGrid Project*. 2007. Acesso em fevereiro de 2007. Disponível em: <http://eu-datagrid.web.cern.ch/eu%2Ddatagrid/>.

FIGUEIREDO, C. M. S.; NAKAMURA, E. F. *Computação Móvel: Novas Oportunidades e Novos Desafios*. 2003. Revista T&C Amazônia. Disponível em: <https://portal.fucapi.br/tec/imagens/revistas/ed002_016_028.pdf>.

FOSTER, I. What is the grid? -a three point checklist. *GRIDtoday*, v. 1, n. 6, July 2002. Disponível em: <http://www.gridtoday.com/02/0722/100136.html>.

FOSTER, I. Globus toolkit version 4: software for service-oriented systems. In: 3779, S.-V. L. (Ed.). International Conference on Network and Parallel Computing, 2005. v. 3779, p. 2–13. Disponível em: <http://www.globus.org/alliance/events/sc05/GT4.pdf>.

FOSTER, I. et al. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. Junho 2002. Global Grid Forum. Disponível em: <http://www.globus.org/alliance/publications/papers.php>.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. 2001. 200-222 p. International Journal of High Performance Computing Applications. Disponível em: <www.globus.org/alliance/publications/papers/>.

GENOME. *Generic Model Organism Database project,*. 2007. Acesso em fevereiro de 2007. Disponível em: <http://www.gmod.org/Genome_grid>.

GLITE. *Lightweight Middleware for Grid computing*. 2009. Acesso em janeiro de 2009. Disponível em: <http://glite.web.cern.ch/glite/>.

GRIAULE. *GRIAULE Biometrics*. 2008. Acesso em setembro de 2009. Disponível em: <http://www.griaulebiometrics.com/page/pt-br/index>.

GRIAULE. *GRIAULE Biometrics*. 2008. Acesso em agosto de 2009. Disponível em: <http://www.griaulebiometrics.com/page/en-us/node/3121?page=1>.

GRIDWAY. *GridWay Metascheduler*. 2009. Acesso em agosto de 2009. Disponível em: <http://www.gridway.org/doku.php>.

GT4. *Globus toolkit 4 overview*. 2007. Acesso em agosto de 2007. Disponível em: <http://www.globus.org/toolkit/about.html>.

GT4. *Welcome to Globus*. 2007. Acesso em junho de 2007. Disponível em: <http://www.globus.org/>.

IETF. *The Internet Engineering Task Force*. 2009. Acesso em novembro de 2009. Disponível em: <http://www.ietf.org/>.

INTERPOL. *The International Criminal Police Organization*. 2010. Acesso em fevereiro de 2010. Disponível em: <http://www.interpol.int/>.

KAZIENKO, J. F. *Assinatura Digital de Documentos Eletrônicos através da Impressão Digital*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, Fevereiro 2003.

LASZEWSKI1, G. von; HATEGAN, M. Workflow concepts of the java cog kit. *Journal of Grid Computing, Springer Netherlands*, v. 3, n. 3-4, p. 239–259, 2005.

LEGION. *Worldwide Virtual Computer e pluribus unum: one out of many*. 2009. Acesso em janeiro de 2009. Disponível em: <http://legion.virginia.edu/index.html>.

MALTONI, D. et al. *Handbook of Fingerprint Recognition*. Second. [S.l.]: Springer, 2005. (Springer Professional Computing).

MAMMOGRID. *The Mammogrid Project*. 2007. Acesso em fevereiro de 2007. Disponível em: <https://savannah.cern.ch/projects/mammogrid/>.

MARQUEZAN, C. C.; CARISSIMI, A.; NAVAUX, P. O. A. Web services para computação de alto desempenho. In: SOCIEDADE BRASILEIRA DE COMPUTAçãO - SBC. Ouro Preto - MG: VII Workshop em Sistemas Computacionais de Alto Desempenho - WCGA, 2006.

MORETTI, C. et al. Challenges in executing data intensive biometric workloads on a desktop grid. In: . [S.l.]: Proceedings of the Workshop on Large-Scale and Volatile Desktop Grids (PCGrid'07), 2007.

NASCIMENTO, A. C. G.; MIRANDA, M. N. Integração de ambientes heterogêneos de grades computacionais. In: 24° SIMPóSIO BRASILEIRO DE REDES DE COMPUTADORES - SBRC. Curitiba - PR: IV Workshop de Computação em Grid e Aplicações - WCGA, 2006. p. 103–115.

NCSA. *National Center for Supercomputing Applications at the University of Illinois*. 2009. Acesso em novembro de 2009. Disponível em: <http://www.ncsa.illinois.edu>.

NEESGRID. *Building the National Virtual Collaboratory for Earthquake Engineering*. 2004. Acesso em dezembro de 2007. Disponível em: <http://www.neesgrid.org/>.

NIELSEN, J. *Response Times: The Three Important Limits*. 2007. Acesso em novembro de 2009. Disponível em: <http://www.useit.com/papers/responsetime.html>.

NIMRODG. *Nimrod: Tools for Distributed Parametric Modelling*. 2009. Acesso em agosto de 2009. Disponível em: <http://www.csse.monash.edu.au/ davida/nimrod/nimrodg.htm>.

NIST. *National Institute of Standards and Technology*. 2008. Acesso em janeiro de 2008. Disponível em: <http://www.nist.gov>.

OGF. *Open Grid Forum*. 2008. Acesso em dezembro de 2008. Disponível em: <http://www.gridforum.org/>.

OGSA-DAI. *OGSA-DAI component diagram*. 2009. Acesso em novembro de 2009. Disponível em: <http://ogsa-dai.sourceforge.net/documentation/ogsadai3.2/ogsadai3.2-gt/OverviewComponents.html>.

OLIVEIRA, J. A. de. *Um estudo comparativo de cargas de trabalho e políticas de escalonamento para aplicações paralelas em clusters e grids computacionais*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação - ICMC-USP, São Carlos - SP, Setembro 2006.

OPENPBS. *PBS Works Enabling On-Demand Computing*. 2009. Acesso em novembro de 2009. Disponível em: <http://www.pbsgridworks.com>.

OURGRID. *The OurGrid Community*. 2009. Acesso em janeiro de 2009. Disponível em: <http://www.ourgrid.org/>.

RANGANATHAN, K.; FOSTER, I. Decoupling computation and data scheduling in distributed data-intensive applications. *In Proceedings of the 11th IEEE Symposium on High Performance Distributed Computing (HPDC)*, IEEE, Edinburgh, Scotland, 2002.

REIS, V. Q. dos. *Escalonamento em grids computacionais: estudo de caso*. Dissertação (Mestrado) — Instituto de Ciências Matemá?ticas e de Computação - ICMC-USP, São Carlos - SP, Julho 2005.

SANTOS, L. A. S. *Uma Proposta de Escalonamento Descentralizado de Tarefas para Computação em Grade*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul - UFRGS, 2006.

SENGER, H.; STANZANI, S. L.; RONDINI, R. A. *Uma Introdução ao Desenvolvimento de Aplicações Para Grades Computacionais Baseadas no Padrão OGSA*. Outubro 2006. Mini-curso do Workshop em Sistemas de Computacionais de Alto Desempenho - WSCAD.

SOTOMAYOR, B. *The Globus Toolkit 4 Programmer's Tutorial*. 2005. Disponível em: <http://gdp.globus.org/gt4-tutorial/>.

SSP. *Secretaria da Segurança Pública*. 2007. Acesso em dezembro de 2007. Disponível em: <http://www.ssp.ba.gov.br/listanoticias.asp>.

SSP. *Secretaria da Segurança Pública*. 2009. Acesso em outubro de 2009. Disponível em: <http://wv.ww.ssp.ba.gov.br/noticias.asp?cod_Noticia=5473>.

SSP-BA. *Estatística Criminal*. 2010. Acesso em março de 2010. Disponível em: <http://www.ssp.ba.gov.br/estatistica.asp>.

SSP-RJ. *Estatísticas de Segurança Pública - RJ*. 2010. Acesso em março de 2010. Disponível em: <http://www.isp.rj.gov.br/Conteudo.asp?ident=162>.

SSP-SP. *Estatísticas*. 2010. Acesso em março de 2010. Disponível em: <http://www.ssp.sp.gov.br/estatistica/default.aspx>.

TEIXEIRA, F. C. *Um Escalonador para Grades Computacionais Utilizando Modelos Econômicos*. Dissertação (Mestrado) — Universidade Estadual de Campinas - Instituto de Computação, Campinas - SP, Dezembro 2006.

TERAGRID. *The TeraGrid Project*. 2009. Acesso em novembro de 2009. Disponível em: <http://www.teragrid.org>.

TONELLOTTO, N.; WIEDER, P.; YAHYAPOUR, R. *A Proposal for a Generic Grid Scheduling Architecture*. [S.l.], Janeiro 2006.

TORQUE. *TORQUE Resource Manager*. 2009. Acesso em agosto de 2009. Disponível em: <http://www.clusterresources.com/products/torque-resource-manager.php>.

UNICORE. *Uniform Interface to Computing Resources*. 2009. Acesso em novembro de 2009. Disponível em: <http://www.unicore.eu/>.

VENUGOPAL, S.; BUYYA, R.; RAMAMOHANARAO, K. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 38, n. 1, p. 3, 2006. ISSN 0360-0300. Disponível em: <http://doi.acm.org/http://doi.acm.org/10.1145/1132952.1132955>.

VIOLA, F. M. *Estudo sobre Formas de Melhoria na Identificação de Características Relevantes em Imagens de Impressão Digital*. Dissertação (Mestrado) — Universidade Federal Fluminense, Nitéroi - RJ, Agosto 2006.

WATSON, C. I. et al. *User's Guide to NIST Fingerprint Image Software 2 (NFIS2)*. New York, 2004. Conteúdo sujeito a Lei de Controle de Exportação dos EUA.

YAGOUBI, B.; SLIMANI, Y. Load balancing strategy in grid environment. *Journal of Information Technology and Applications*, v. 1, n. 4, p. 285–296, Março 2007.

YANG, C.-T.; HAN, T.-F. G-blast: a grid-based solution of mpiblast on computational grids. In: . Howard Hotel, Taipei, Taiwan: Proceedings of the ICS?2006 International Computer Symposium, 2006. v. 3, p. 1272–1277.

ZEGARRA, J. A. M.; LEITE, N. J.; TORRES, R. da S. Efficient and effective content-based image retrieval framework for fingerprint databases. In: *V Workshop of Theses and Dissertations, XXI Brazilian Symposium on Databases*. [S.l.: s.n.], 2006. p. 77 – 82.

ZHOU, S. Lsf: Load sharing in large-scale heterogeneous distributed systems. In: *Workshop on Cluster Computing*. Tallahassee, FL: [s.n.], 1992.

ZHU, Y. *A survey on grid scheduling systems*. [S.l.], 2003.