REINALDO LOURENSO

SEGMENTAÇÃO DE OBJETOS COMPLEXOS EM UM SISTEMA DE BANCO DE DADOS OBJETO RELACIONAL BASEADO EM *GRID*S

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Elétrica.

São Paulo

2005

OK

REINALDO LOURENSO

SEGMENTAÇÃO DE OBJETOS COMPLEXOS EM UM SISTEMA DE BANCO DE DADOS OBJETO RELACIONAL BASEADO EM *GRID*S

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Elétrica.

Área de Concentração: Sistemas Eletrônicos

Orientador: Professor Doutor
Sergio Takeo Kofuji

São Paulo
2005

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com anuência de seu orientador.

São Paulo, 02 de Janeiro de 2006.

Assinatura do autor

Assinatura do orientador

FICHA CATALOGRÁFICA

Lourenso, Reinaldo
Segmentação de Objetos Complexos em um Sistema de Banco de Dados Objeto Relacional Baseados em *GRIDS* / Reinaldo Lourenso. -- ed.rev. -- São Paulo, 2005.
123 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1. Computação alto desempenho 2. Computação em *Grid.*
3. Banco de Dados Distribuídos.
Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II. t.

À minha família,
pelo incentivo e motivação
na realização deste trabalho.

AGRADECIMENTOS

Agradeço primeiramente ao amigo e orientador Sergio Takeo Kofuji por sempre ter acreditado no meu potencial, ter dado apoio e incentivo na realização deste trabalho e pela paciência.

Agradeço aos professores da banca pela participação na avaliação, pelas críticas que me tornarão um pesquisador melhor e pelas tolerâncias com as imperfeições da minha tese.

Agradeço aos amigos do Laboratório de Sistemas Integráveis (LSI) pela revisão, participação e ajuda no desenvolvimento da tese.

Agradeço aos demais amigos pela oportunidade de conhecê-los.

RESUMO

O principal objetivo desta tese consiste em propor, desenvolver e implementar uma infra-estrutura para gerenciamento de um Banco de Dados baseado em *Grid*.

O armazenamento de objetos complexos como áudio, vídeo, softwares etc., em Sistemas de Banco de Dados, sempre se dá de maneira integral, ou seja, o documento, independente do seu tamanho, não é fragmentado pelo Sistema de Gerência de Banco de Dados (SGBD) ao ser armazenado. Metodologias de modelagem de dados utilizadas também não especificam a fragmentação ou segmentação de um documento complexo quando do seu armazenamento, pois só contemplam a fragmentação das estruturas de armazenamento, no caso relações ou classes, e não os objetos que serão armazenados.

Ao avaliarmos o desempenho de sistemas que armazenam objetos complexos, verificamos que o tamanho dos objetos armazenados influencia consideravelmente o desempenho destes sistemas. Como objetos multimídia, softwares, etc., necessitam de grandes volumes em disco para seu armazenamento, métodos de replicação ou distribuição de cópias tradicionais tornam-se muito dispendiosos e por vezes ineficientes.

Com a infra-estrutura desenvolvida neste trabalho foi possível segmentar e distribuir atributos complexos de linhas de uma tabela, instaladas em Bancos de Dados baseado em *Grid*. Nossa solução melhorou o desempenho do sistema que tinha a necessidade de armazenar documentos de tamanho elevado, acima de um tamanho limite. Também foi testada com sucesso a possível utilização dos códigos LDPC nesta infra-estrutura. Entretanto, não observamos ganhos que justificassem sua utilização em aplicações semelhantes a nossa.

ABSTRACT

This Thesis presents a proposal of an infrastructure to allow the distribution of data in a Database Grid.

The storage of complex objects, such as audio, video and software etc. in Databases is always done in an integral way. This means that the object, regardless of its size, it is not fragmented by the Database Management System (DBMS). Methodologies used for data modeling also do not allow fragmentation or segmentation of complex objects. This happens because only the fragmentation of storage structures such as tables or classes are taken into account, not the embedded objects.

When we evaluate the performance of systems that store complex objects, we can verify that the size of the stored objects has considerable impact. Since multimedia objects or software distribution package require significant disk space for storage, traditional methods for replication or distribution of copies become very costly and many times inefficient.

With the infrastructure developed in this work it was possible to segment and to distribute complex attributes of lines of a table in Database Grids. In this way, our solution improves the performance of the system that had the necessity to store documents of raised size, above of a specified boundary-value. Also the possible use of codes LDPC in this infrastructure was tested successfully. However, does not observe profits that justified its use in same ours applications.

SUMÁRIO

CAPÍTULO 1. INTRODUÇÃO 1

1.1. DESAFIO DO PROJETO 3
1.2. ÁREA DE CONCENTRAÇÃO 6
1.3. RELEVÂNCIA 7
1.4. REVISÃO BIBLIOGRÁFICA 9
1.5. ESTADO DA ARTE 11
1.6. ESTRUTURA DA TESE 13

CAPÍTULO 2. BANCO DE DADOS DISTRIBUÍDOS, *GRIDS*, FRAGMENTAÇÃO DE DADOS E OS CÓDIGOS LDPC 15

2.1. SISTEMA DE GERÊNCIA DE BANCO DE DADOS DISTRIBUÍDOS 16
2.1.1. REQUISITOS FUNCIONAIS DE UM SGBDD 17
2.1.2. VANTAGENS E DESVANTAGENS DE UM SGBDD 18
2.2. GRIDS COMPUTACIONAIS 18
2.2.1. Escalonamento de Aplicações no Grid 22
2.3. GRIDS DE DADOS E BANCO DE DADOS EM GRIDS 23
2.3.1. GRIDS DE DADOS 23
2.3.2. BANCO DE DADOS EM GRIDS 26
2.4. FRAGMENTAÇÃO DE DADOS EM SGBD DISTRIBUÍDOS 31
2.5. OS CÓDIGOS LDPC (LOW DENSITY PARITY CHECK) 35
2.5.1. INTRODUÇÃO - LDPC 35
2.6. CONCLUSÃO 36

CAPÍTULO 3. O PROBLEMA DA FRAGMENTAÇÃO DE DADOS E A PROPOSTA DE DISTRIBUIÇÃO DA BASE DE DADOS EM *GRIDS* 37

3.1. DESCRIÇÃO DO PROBLEMA 39
3.1.1. SEGMENTAÇÃO DE OBJETOS EM UM BANCO DE DADOS OBJETO RELACIONAL BASEADO EM GRID 43
3.2. DETALHAMENTO DO PROBLEMA X PROPOSTAS 43
3.2.1. PROPOSTA I - SEGMENTAÇÃO DO ATRIBUTO COM REPLICAÇÃO DO RESTANTE DA LINHA ... 46
3.2.2. PROPOSTA II – SEGMENTAÇÃO DO ATRIBUTO SEM REPLICAÇÃO DA LINHA E DISTRIBUIÇÃO UTILIZANDO O IBP 47
3.2.3. PROPOSTA III – SEGMENTAÇÃO DO ATRIBUTO SEM REPLICAÇÃO DA LINHA UTILIZANDO APENAS O SGBDD DISPONÍVEL 48
3.3. CONCLUSÃO 49

CAPÍTULO 4. IMPLEMENTAÇÃO, TESTES E RESULTADOS. 50

4.1. Cenário 54
4.2. Modelo Lógico das Tabelas do Protótipo do Sistema 55
4.3. Requisitos do Protótipo do Sistema 57
4.4. Arquitetura de Programas 57
4.4.1. Agentes de Monitoramento 58
4.4.2. Módulo de Entrada 59
4.4.3. Modulo de Recuperação 61
4.5. Fluxo de Processamento 64
4.6. Configuração da Distribuição dos Dados 66
4.7. Testes e Resultados 67
4.7.1. Testes de Validação 68
4.7.2. Testes de Avaliação 71
4.8. Trabalhos Correlatos 85
4.9. Conclusão 88

CAPÍTULO 5. CONCLUSÕES E TRABALHOS FUTUROS 89

5.1. Resultados 90
5.2. Limitações do Trabalho 92
5.3. Trabalhos Futuros 92

ANEXOS 94

Sistema de Banco de Dados Distribuídos 95
As Doze Regras para um SGBDD. 95
Os Códigos LDPC 98
Códigos de Propriedades Assintóticas 101
Determinação da Intensidade Computacional 102

REFERÊNCIAS 104

LISTA DE FIGURAS

FIGURA 1. SEGMENTAÇÃO DOS ATRIBUTOS E SUA DISTRIBUIÇÃO PELOS COMPUTADORES DO *GRID*. 5

FIGURA 2. ARQUITETURA DE BANCO DE DADOS DISTRIBUÍDA. 17

FIGURA 3. - *GRID* COMPUTACIONAL. 19

FIGURA 4. DIVISÃO DOS SEGMENTOS ENTRE OS COMPUTADORES DO *GRID*. 39

FIGURA 5. RELAÇÃO (TABELA) DE PROJETOS. 43

FIGURA 8. RELAÇÃO PROJ COM UM TIPO COMPLEXO DE DADOS. 45

FIGURA 9. PROPOSTA DE REPLICAÇÃO DA LINHA E SEGMENTAÇÃO DO DOCUMENTO CAD. 46

FIGURA 10. PROPOSTA DE BANCO DE DADOS EM *GRID* COM A INFRA-ESTRUTURA IBP [LOURENSO05]. 48

FIGURA 11. ARQUITETURA DO BANCO DE DADOS EM *GRID*. 51

FIGURA 12. MODELO CONCEITUAL/LÓGICO DAS TABELAS DO SISTEMA. 56

FIGURA 13. MODELO DE CLASSES DOS MÓDULOS DO PROTÓTIPO DO SISTEMA. 58

FIGURA 14. JANELA DO MÓDULO DE ENTRADA. 61

FIGURA 15. JANELA DO MÓDULO DE RECUPERAÇÃO. 63

FIGURA 16. FUXO DE INCLUSÃO DE DADOS NO *GRID*. 65

FIGURA 17. – FUXO DE RECUPERAÇÃO DE DADOS NO *GRID*. 66

FIGURA 18. TEMPO TOTAL DE PROCESSAMENTO. 74

FIGURA 19. TEMPO TOTAL DE *DOWNLOAD*. 74

FIGURA 20. DIFERENÇA ENTRE OS TEMPOS TOTAIS. 75

FIGURA 21. DIFERENÇA ENTRE OS TEMPOS DE *DOWNLOAD*. 75

FIGURA 22. TEMPO TOTAL DE PROCESSAMENTO COM DOCUMENTOS CODIFICADOS. 77

FIGURA 23. TEMPO TOTAL DE *DOWNLOAD* COM DOCUMENTOS CODIFICADOS. 77

FIGURA 24. DIFERENÇA ENTRE OS TEMPOS TOTAIS COM DOCUMENTOS CODIFICADOS. 78

FIGURA 25. DIFERENÇA ENTRE OS TEMPOS DE *DOWNLOAD* COM DOCUMENTOS CODIFICADOS. 78

FIGURA 26. TEMPO TOTAL DE PROCESSAMENTO. 80

FIGURA 27. TEMPO TOTAL DE *DOWNLOAD*. 80

FIGURA 28. DIFERENÇA ENTRE OS TEMPOS DE PROCESSAMENTO. 81

FIGURA 29. DIFERENÇA ENTRE OS TEMPOS DE *DOWNLOAD*. 81

FIGURA 30. : (A) EXEMPLO 1 - NÍVEL DE CÓDIGOS SISTEMÁTICOS PARA N = 4, M = 3. (B) EXEMPLO 3 - NÍVEL DE CÓDIGOS SISTEMÁTICOS PARA N = 8, M = 8 [RYAN04] 99

FIGURA 31: (A) EXEMPLO DO CÓDIGO GALLAGER PARA N = 4, M = 3. NOTE QUE OS NÓS DIREITOS DEFINEM RESTRIÇÕES ENTRE OS NÓS ESQUERDOS, E NÃO ARMAZENAM BLOCOS DCODIFICADOS. (B) EXEMPLO CÓDIGOS IRA PARA N = 4, M = 3. OS NÓS ACUMULADORES ESQUERDOS SÃO ARMAZENADOS COMO BLOCOS CODIFICADOS. OS NÓS DIREITOS SÃO USADOS PARA CÁLCULOS [RYAN04]. 101

LISTA DE TABELAS

TABELA-01. CARACTERÍSTICAS TÍPICAS DE DIFERENTES PLATAFORMAS DE EXECUÇÃO [CIRNE02]. ..20
TABELA-02. EXEMPLO DOS DOCUMENTOS UTILIZADOS. ..69
TABELA-03. CONTEÚDO DA TABELA METADOC_TB ..69
TABELA-04. CONTEÚDO DA TABELA DOCS_TB (1) ..70
TABELA-05. CONTEÚDO DA TABELA DOCS_TB (2) ..70
TABELA-06. CONTEÚDO DA TABELA PARTEDOC_TB (1) ..70
TABELA-07. CONTEÚDO DA TABELA PARTEDOC_TB (2) ..71
TABELA-08. DOCUMENTOS UTILIZADOS NOS TESTE DE AVALIAÇÃO ..72
TABELA 09. RESULTADOS OBTIDOS DOS DOCUMENTOS ÚNICOS E SEGMENTADOS ..82
TABELA 10. RESULTADOS OBTIDOS DOS DOCUMENTOS CODIFICADOS E SEGMENTADOS ..83

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface; Interface de Programação de Aplicações;

BLOB - Binary Large Objects;

BD - Banco de Dados;

BDD – Banco de Dados Distribuídos;

Data Warehousing - Ë uma aplicação sobre um banco de dados que contém informações de âmbito estratégico em um nível corporativo;

Data Mart - Similar ao Data Warehouse só que num nível departamental;

Data Mining - Aplicação que utilizam os mais variados tipos de algoritmos para consultas em uma base extensa de dados;

DB - Database;

DDBMS - Distributed Database Manager System;

DSM – Distributed Shared Memory; Memória Compartilhada Distribuída;

HPS – High Performance Storage; Armazenamento de Alto Desempenho;

JOINT - Processo que permite a junção de duas ou mais consultas;

IBP - Internet Backplane Protocol

LAN - Local Area Network;

LDPC - Low-Density Parity-Check;

Query (Querie) - Processo de consulta para uma ou mais tabelas;

Round-Robin - Esquema de controle de escalonamento de acesso a dados ou dispositivos compartilhados;

SBD - Sistema de Banco de Dados;

SGBD – Sistema de Gerência de Banco de Dados;

SBDD - Sistema de Banco de Dados Distribuídos;

SGBDD - Sistema de Gerência de Banco de Dados Distribuídos;

SGBDOO - Sistema de Gerência de Banco de Dados Orientado a Objetos;

SGBDOR - Sistema de Gerência de Banco de Dados Objeto Relacional;

SGBDRO - Sistema de Gerência de Banco de Dados Relacional Objeto;

Stored Procedures – São procedimentos, a maioria padrão, que podem ser executados por qualquer programa, aplicação ou processo que necessite de um procedimento pré-especificado;

Thread – É um fluxo de informações que está no controle de um processo, geralmente utilizado por sistemas operacionais ou SGBD para executar uma tarefa requisitada;

VLDB - Very Large Database;

NOTAÇÃO

Abaixo será descrita a notação utilizada nesta tese:

a) Termos em inglês onde não se encontraram traduções diretas ao português, até o momento da escrita da tese, serão apresentadas em itálico:

Exemplo: *GRID*

b) Termos em inglês traduzidos serão apresentados à direita dos termos em português, nas primeiras três citações, em itálico e entre parênteses:

Exemplo: **convencional** (*commodity*).

c) Quando for definido um novo termo, este será apresentado em negrito:

Exemplo: Na **fragmentação** de dados, uma relação ou objeto poderá ser particionada em diversos fragmentos que serão armazenados em diferentes nós.

d) Citações à bibliografia serão apresentadas no padrão ABNT adotado pelo Serviço de Bibliotecas da Escola Politécnica da USP.

e) Quando algum termo for enfatizado, este será apresentado em sublinhado:

Exemplo: O processo de segmentação não só irá considerar o tamanho do atributo, mas também a quantidade de componentes do Grid.

Capítulo 1. INTRODUÇÃO

A verdadeira sabedoria consiste em saber como aumentar o bem-estar do mundo.

Leonardo da Vinci

Desde a Antigüidade o homem sempre procurou armazenar informações como uma maneira de manter e compartilhar o conhecimento adquirido. Contudo, os métodos para armazenar, compartilhar e, principalmente, para tornar disponíveis essas informações de forma mais rápida, têm se tornado um grande problema.

Atualmente, tem sido possível armazenar vários tipos de informação em meio eletrônico, desde textos, fotos de alta definição, músicas e até filmes inteiros de vários Gigabytes, ou seja, qualquer informação relevante para o usuário.

Se considerarmos que esse cenário tende a se manter, faz-se necessário o desenvolvimento de novas metodologias, técnicas e mecanismos que ajudem no armazenamento e na disponibilização dessas informações, em tempo hábil, tentando com isso solucionar ou, ao menos, minimizar esses problemas.

Esses problemas [BAIÃO00], [MATOSO00] têm sido tratados de inúmeras formas principalmente com o auxílio de sistemas de computadores. A melhoria na facilidade de uso dos programas (*software*), assim como a evolução tecnológica de componentes eletrônicos, ocasionou a disseminação dos computadores na sociedade. Isso tem permitido que técnicas avançadas de processamento e equipamentos (*hardware*) de alto desempenho estejam presentes em sistemas computacionais de pequeno porte.

O objetivo deste trabalho é propor uma infra-estrutura que possibilite identificar e segmentar atributos complexos de instâncias de relações ou de classes de objetos em grandes bases de dados instaladas em *Grids*. Estes *Grids* são compostos por computadores interconectados, por exemplo, por redes de alta velocidade (Internet 2). Além disso, propõe-se que este trabalho seja realizado usando o máximo de softwares "convencionais". Também são considerados outros desafios, como: minimização da replicação de dados, utilização de elementos convencionais e implementação em um sistema portável e escalável.

O conceito **convencional** (*commodity*) atribui ao objeto referenciado a característica de facilidade de aquisição, principalmente pelo problema de custo e também por fatores como usabilidade e disponibilidade no mercado. Por arquitetura convencional entende-se que os equipamentos possuem tecnologias largamente utilizadas, o que facilita sua aquisição. Quando o conceito refere-se a programas (*softwares*), entende-se que os programas podem ser facilmente adquiridos e utilizados pelo usuário, como os programas (*softwares*) de código livre e/ou aberto.

Por grandes bases de dados considera-se uma quantidade enorme de dados que, devido a considerações de desempenho e disponibilidade (disco X processamento), não deve ser armazenada somente em um único computador, mas ser particionada, podendo, cada parte, ser armazenada em um dos servidores componentes do *Grid*. Como exemplos, podemos citar [GENOME05], [SKYSURVEY05]. O particionamento dessas grandes bases de dados e posterior gerenciamento durante o processo de pesquisa e reconstrução das informações, implica em enfrentar talvez a maior dificuldade dos *Grid*s computacionais.

1.1. DESAFIO DO PROJETO

O desafio do trabalho é propor uma solução para os seguintes problemas fundamentais:

I. Identificar e segmentar atributos complexos de instâncias de **relações** ou de **classes de objetos** que possuam tamanhos elevados em segmentos menores, conforme a disponibilidade de servidores de Banco de Dados associados ao *Grid*;
II. Gerenciar os segmentos durante um processo de pesquisa e recomposição, utilizando somente tecnologia convencional;
III. Mapear as instâncias disponíveis de Banco de Dados associados ao *Grid*, no momento em que um determinado atributo, de tamanho elevado, necessite ser segmentado ou recuperado;

IV. Gerar os metadados de cada um dos segmentos que possibilite, de maneira efetiva, pesquisar, recuperar e recompor a informação original;
V. Fornecer acesso único e transparente aos usuários para a aplicação que estiver sendo executada no *Grid*.

A Figura 1 apresenta a seqüência proposta para a segmentação e distribuição dos segmentos pelos servidores do *Grid*. A quantidade de segmentos é determinada pela quantidade de servidores disponíveis no momento da segmentação. Cada servidor só receberá um único segmento.

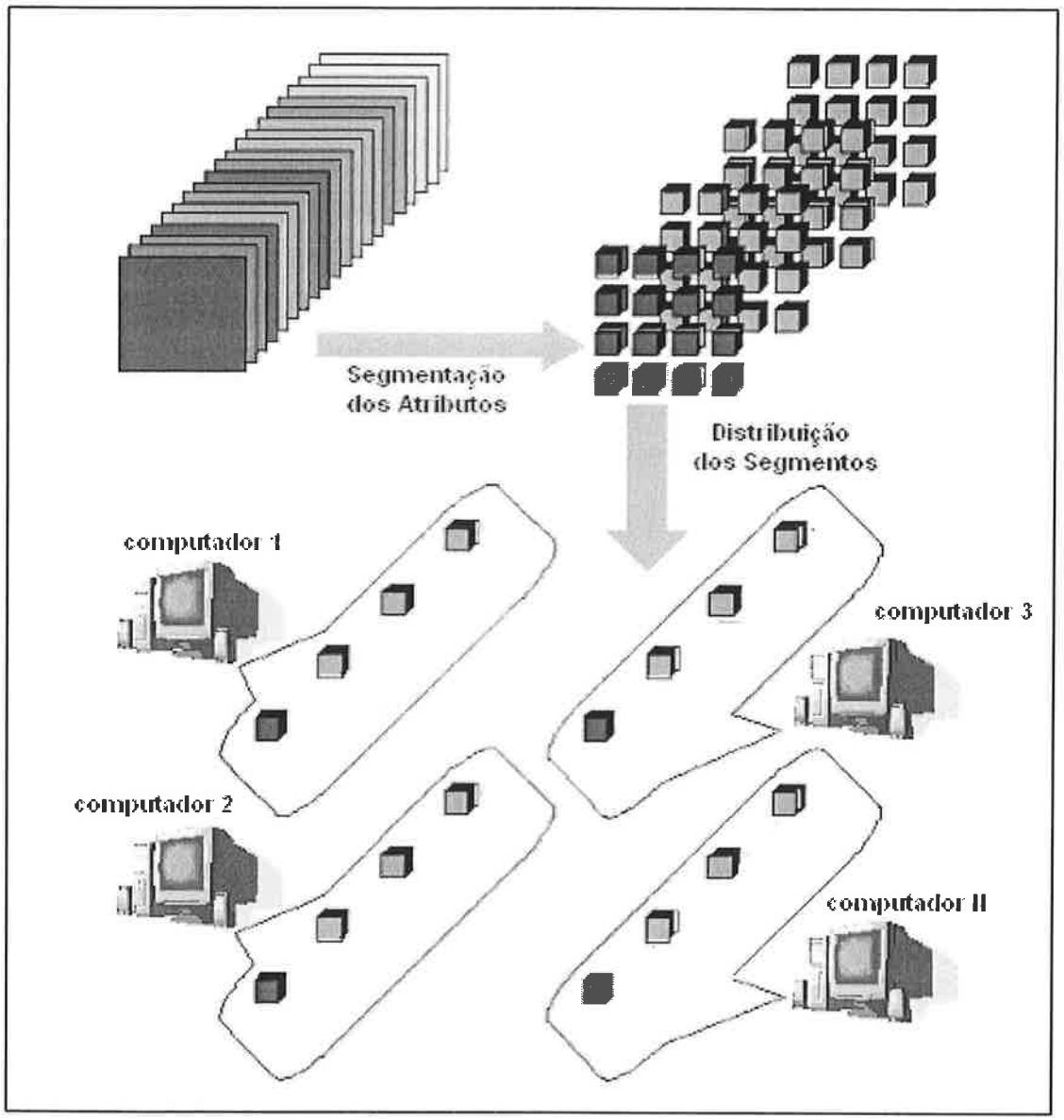

FIGURA 1. SEGMENTAÇÃO DOS ATRIBUTOS E SUA DISTRIBUIÇÃO PELOS COMPUTADORES DO *GRID*.

Não serão consideradas as ferramentas adequadas para visualização, edição ou execução dos atributos; só serão consideradas para este trabalho a segmentação/recomposição e a distribuição/recuperação, bem como a avaliação dos códigos LDPC na codificação dos atributos a serem segmentados em um processo do Sistema de Banco de Dados.

1.2. ÁREA DE CONCENTRAÇÃO

Este trabalho se relaciona com os seguintes tópicos: Computação de Alto Desempenho, *Grid* Computacional, *Grid*s de Dados, Database Grids, Aplicações em *Grid*, Armazenamento de Alto Desempenho e Gerenciamento de Dados, Computação de Dados Intensiva e Banco de Dados Volumosos.

Problemas cada vez maiores e mais complexos fazem com que a quantidade de dados gerados, por vezes, exceda a capacidade de processamento e armazenamento dos sistemas. *Clusters* e *Grid*s são opções para armazenamento, processamento e disponibilização de grandes quantidades de dados. Além disso, a demanda por processamento é quase sempre orientada à disponibilização e análise interativa de tempo real, o que requer equipamentos específicos, não-convencionais e custosos.

Neste sentido, **Computação de Alto Desempenho** pode ser vista como uma linha de pesquisa complexa que procura solucionar problemas que necessitam de grande poder computacional. Esta área envolve várias outras, tais como: Computação Paralela, Processamento Paralelo Massivo, Computação Distribuída, Super Computação. Mais recentemente, a tecnologia de *Grid*s computacionais tem recebido atenção especial dos grupos de pesquisa e está sendo utilizada em grande variedade de aplicações [MOORE04], [LIU03].

Na convergência das áreas anteriores pode-se considerar **Armazenamento de Alto Desempenho e Gerenciamento de Dados** (*High Performance Storage and Data Management*) como a que trata do estudo de técnicas de alto desempenho para armazenamento de dados que demandam grande poder computacional. Para isso, a integração de equipamentos de última geração – como entrada/saída paralela, apoio a redes de alta velocidade, estações de trabalho, entre outros elementos – é essencial para alcançar eficiência de todo sistema.

1.3. RELEVÂNCIA

Como já mencionado, com a tendência de crescimento da utilização de armazenamento tipos de dados complexos em meio eletrônico, criam-se problemas de armazenamento, tratamento e disponibilização dessas informações.

Notou-se que em muitos trabalhos [BAIÃO00] [OZSU02], inclusive atuais, procurou-se resolver o problema de armazenamento desses tipos de dados em SGBD Orientados a Objetos, pois, até a pouco tempo, os SGBD Relacional convencionais careciam de maior poder semântico de representação e manipulação dos dados que possuem estruturas mais sofisticadas de representação.

Essa tendência deveu-se ao sucesso das linguagens de programação orientada a objetos que conseguiram em parte resolver seus problemas, encontrando nos mecanismos de herança, encapsulamento, entre outras, respostas aos seus problemas de tipos de dados complexos e também de reutilização de código. No entanto, estas aplicações se ressentiam de capacidades inerentes aos bancos de dados como persistência, consultas *ad-hoc*, controle de concorrência e transações, entre outras. Assim, para atender por completo às novas necessidades das aplicações, a direção está na confluência das metodologias de orientação a objetos e banco de dados, dando origem aos Sistemas de Gerência de Bases de Dados Orientados a Objetos (SGBDOO) e aos Sistemas de Gerência de Bases de Dados Relacionais-Objetos (SGBDRO).

Contudo, mais recentemente, apesar da melhoria nas técnicas de modelagem de dados para os SGBDOO, o recuo na utilização desses tipos de sistemas e até mesmo a sua troca para SGBDRO, deve-se à grande dificuldade de implementação de projetos de Banco de Dados para estes sistemas e também devido ao desempenho inicialmente esperado não ter sido atingido.

Em projetos correntes de *Grids*, como apresentado em [MARIA], Banco de Dados e Sistemas de Bancos de Dados são tipicamente ferramentas úteis para

gerenciar metadados, recursos de dados e *workflows*, etc. A presença de Banco de Dados é reduzida em aplicações científicas que rodam em *Grid*s. Aplicações típicas de uso intensivo de dados científicos em *Grid* ainda usam *Flat Files* para processar e armazenar dados e não podem se beneficiar do poder que os Sistemas de Banco de Dados oferecem.

Esses projetos desprezam o ponto forte dos Sistemas de Banco de Dados como a sua habilidade de indexar, pesquisar e unir eficientemente grandes quantidades de dados – freqüentemente em paralelo. Pode ser ineficiente mover grandes quantidades de dados para a consulta, quando você pode mover a consulta para os dados e executar a consulta em paralelo. Por esta razão, pode ser útil um *Grid* apoiar diferentes Sistemas de Gerência de Banco de Dados até que aplicações *SQL* possam ser disponibilizadas tão facilmente quanto às aplicações tradicionais em *Grid* codificadas em C ou Fortran.

Portanto, a integração de Banco de Dados e *Grid*s é promissora para aplicações de Banco de Dados de negócios e científicos, bem como gerenciar metadados. Esforços significativos têm sido feitos na definição de requisitos, protocolos e na implementação de interface para acessar Banco de Dados em ambientes de *Grid*s.

Outro ponto relevante é a alocação de dados, que envolve encontrar a distribuição "ótima" dos dados através de nós do *Grid* de forma a minimizar o custo das aplicações sobre esses dados. A alocação é um aspecto crítico num Sistema de Banco de Dados Distribuídos (SBDD), já que uma alocação ineficiente dos dados pode levar a um aumento considerável do custo de acesso ao SBDD. Em geral, em um projeto de distribuição da base de dados, a fase de alocação é realizada após a fase de fragmentação, a qual determina a distribuição dos dados em fragmentos. Um dos principais objetivos que devem ser alcançados na fase de alocação é o aumento da proximidade entre os fragmentos e os nós que os utilizam, minimizando o custo de comunicação.

O que determina a qualidade da solução para o problema de alocação de dados é a eficiência do projeto de alocação. Em outras palavras, a alocação dos dados nos nós deve minimizar, tanto quanto possível, o custo das consultas executadas. Contudo, isso é válido apenas quando trabalhamos com dados estruturados cujos tipos são conhecidos na fase do projeto. Em soluções que trabalham com tipos de dados não-estruturados ou semi-estruturados, a solução da fragmentação se torna ineficiente porque podem ser encontrados conteúdos com tipos e tamanhos diferentes em fragmentos situados em locais diferentes.

Hoje, muitos sistemas altamente distribuídos ou *peer-to-peer* [PLANCK04] estão utilizando códigos para a codificação de arquivos ao invés da utilização de replicação, com a finalidade de aumentar o desempenho e eficiência desses sistemas. Códigos LDPC apresentam-se como uma alternativa para códigos padrão, tais como *Reed-Solomon*, trazendo melhoria no desempenho da decodificação de dados obtidos do ambiente.

Neste momento, é importante salientar que será apresentada uma solução para a integração de um Banco de Dados em *Grid*s com segmentação de atributos complexos (não-estruturados).

1.4. Revisão Bibliográfica

A área de Banco de Dados para aplicações não convencionais tem sido uma das áreas da computação com maior efervescência nos últimos anos. Além dos interesses comerciais específicos, há uma grande demanda por certas áreas de pesquisa e de atividade econômica por soluções para seus problemas de armazenamento e gerência de dados ou objetos. Dentre as áreas mais importantes destacamos a de Sistemas de Informação Geográfica, a de Mineração de Dados e Projeto Assistido por Computador (CAD), entre outras.

Todas essas áreas de aplicação compartilham necessidades comuns de possuírem estruturas de representação mais sofisticadas do que as oferecidas pelas tabelas do modelo relacional tradicional. Operações específicas sobre estruturas de representação complexa também são necessárias para uma gerência eficiente desses dados. A solução para os novos desafios de Banco de Dados aponta para as tecnologias que possibilitem também a manipulação de objetos complexos não-estruturados.

Nieh; Levoy [NIEH92] propõe um algoritmo para o Stanford Dash (*Directory Architecture for Shared Memory*) Multiprocessor, uma máquina MIMD (*Multiple Instruction Multiple Data*; Múltiplas Instruções Múltiplos Dados) de memória compartilhada escalável com 48 processadores. Neste trabalho foram apresentadas as primeiras discussões e dificuldades do assunto, como particionamento de dados e balanceamento de tarefas.

Montani; Perego; Scopigno [MONTANI92] também apresenta uma solução para outra arquitetura MIMD, um sistema nCUBE 2 com 128 nós de processamento, cada um com 4 Mbytes de memória local. Várias estratégias de particionamento da base de dados foram descritas e uma solução híbrida foi adotada.

Baião, Mattoso [BAIÃO98], [BAIÃO00], [MATTOSO98], [MATTOSO00] apresentam uma proposta de implementação de algoritmos para a fase de fragmentação do projeto de Banco de Dados Distribuídos Orientado a Objetos. O desempenho de aplicações em Sistemas de Gerência de Banco de Dados Orientado a Objetos é fortemente afetado pelo projeto de Distribuição, que reduz o acesso a dados irrelevantes pela aplicação e a permutação de dados entre sítios. Isto se deve a dificuldades inerentes nas fases de projeto e implementação de um Banco de Dados Orientado a Objetos.

Smoot, Flaster [SMOOT95] apresentam uma técnica para replicação de arquivos de dados em sistemas distribuídos. O intuito desta técnica é aumentar a disponibilidade de dados, otimizando o espaço em disco. O método empregado é

conhecido como Técnica de Codificação *Reed-Salomon*, e possibilita que cada sítio controle uma pequena quantidade de um arquivo de dados, e ainda possa reconstruir o arquivo inteiro de dados, mesmo que somente um subconjunto de nós esteja ativo. Este foi um dos primeiros trabalhos abordando a segmentação de um arquivo em um ambiente distribuído estudado na elaboração deste trabalho de tese.

Em Lourenso [LOURENSO01] foi apresentada e implementada uma proposta de replicação e distribuição de objetos sob demanda num Banco de Dados baseado em *cluster*, o que propiciava melhor balanceamento de carga no *cluster* nos momentos de alta demanda por certos objetos.

1.5. Estado da Arte

Como já mencionamos, a quantidade de dados gerada pela humanidade vem crescendo de forma exponencial, principalmente no caso de documentos multimídia - documentos de áudio e vídeo, *Computer-Aided Design* (CAD) ou *softwares*. Essa tendência deve a se manter, pelo menos, a curto e médio prazo, como observa Walfredo Cirne no seu trabalho *Grid*s Computacionais: Arquitetura, Tecnologia e Aplicações [CIRNE02].

Vários pesquisadores [PLANK04] [BAIÃO00] têm estudado algorítmos, métodos e técnicas para distribuição de dados para ambientes distribuídos utilizando fragmentação, replicação e segmentação de dados, com a finalidade de melhorar o desempenho desses sistemas pela utilização de processamento paralelo e distribuído quando da requisição desses dados.

Walfredo, [CIRNE02] mostra que uma solução natural para tratar grandes volumes de dados é a utilização de processamento paralelo e distribuído. Além disso, as soluções adotadas são fracamente acopladas, o que as torna adequadas para execução em *Grid*s computacionais.

*Grid*s computacionais, em particular, *Grid*s de Dados são uma promessa de manipulação e disponibilização de grandes volumes de dados em escala global. Em sistemas emergentes de *Grid*s, o gerenciamento de dados é primordial. Os *Grid*s modernos tais como *Grid Physics Network* [LIU03] estão reconhecendo a importância dos SGBD para o gerenciamento de dados.

Reagan W. Moore [MOORE04] apresenta uma tendência de integração entre a tecnologia de *Grid*s e a tecnologia de biblioteca digital. Contudo, ele afirma que ainda é necessário um maior aprimoramento da tecnologia de *Grid*s para que esta possa fornecer as capacidades necessárias para uma biblioteca digital.

M. T. Özsu [OZSU02] apresenta um Banco de Dados específico para vídeo, onde cada vídeo é segmentado em *clips* com intervalo de tempos iguais. Apesar de esse trabalho tratar com fragmentos de um mesmo vídeo, ele define previamente o tamanho que cada segmento terá. A finalidade é possibilitar a reconstrução do vídeo por meio de uma consulta baseada nos atributos semelhantes contidos em cada *clip* desse vídeo. O trabalho considera que o vídeo já foi segmentado e distribuído previamente e trata principalmente da sua reconstrução.

James S. Plank [PLANK04] mostra que a proliferação dos sistemas de armazenamento *peer-to-peer* e a necessidade de se garantir a execução de um código eficiente de particionamento, em vez da utilização de replicação, como um fator crucial de desempenho e eficiência. Ele adota os códigos LDPC como uma alternativa para padrões de códigos que permitam partição, tais como os códigos *Reed-Solomon*, trazendo melhoria no desempenho de decodificação de grandes quantidades de dados.

Fernanda Baião [BAIÃO03] apresenta um estudo de alocação de dados em Banco de Dados Distribuídos, onde aborda os problemas relacionados à distribuição "ótima" dos dados através de nós da rede, de forma a minimizar o custo das aplicações sobre esses dados.

Jim Gray [GRAY04] apresenta problemas, questões técnicas, e o processo de migração de uma base de dados muito grande do projeto SDSS (*Sloan Digital Sky Survey*) de dados astronômicos de um engenho comercial de Banco de Dados Orientado a Objetos, para uma tecnologia de Banco de Dados Relacional, devido ao BDOO não ter atingido os objetivos de desempenho esperado.

Susana Malaika [MALAIKA03] apresenta uma introdução para *Grid* e descreve como o Banco de Dados poderá ser usado neste novo ambiente, baseando-se no Fórum Global sobre *Grid* (GGF) que esta produzindo especificações técnicas para disponibilizar ambos, Banco de Dados Relacional e Repositórios XML, para serem alocados, acessados e replicados dentro do ambiente de *Grid*.

María A., Jim Gray [MARÍA04] apresentam os benefícios de se combinar sistema de Banco de Dados e a tecnologia de *Grid*s para aplicações que tenham acesso intensivo aos dados. O trabalho compara uma aplicação em *Grid* baseada em arquivo MaxBCG existente, com uma implementação SQL equivalente, destacando a diferença entre consultas realizadas entre as bases de dados.

As principais diferenças deste trabalho em relação aos trabalhos apresentados anteriormente consistem na utilização de tecnologia convencional na segmentação de atributos complexos (não-estruturados) e na sua distribuição em um Banco de Dados Objeto Relacional baseado em *Grid*. Foi realizada também a avaliação da possível utilização dos códigos LDPC para este tipo de aplicação.

1.6. Estrutura da Tese

Neste capítulo foi apresentada uma introdução ao problema alvo deste trabalho, mostrando a utilização da área de Banco de Dados, *Grid*s computacionais e algoritmos de fragmentação de dados, bem como os principais desafios dessas áreas. No capítulo 2 serão apresentados os conceitos básicos de um Banco de Dados Distribuídos, *Grid*s Computacionais e *Grids de Dados*, bem como as definições para

Banco de Dados em *Grid*, fragmentação de dados para ambientes distribuídos, e do código LDPC que será utilizado para segmentação de documentos neste trabalho, conceitos estes, necessários para o entendimento do escopo do trabalho.

O problema tratado e um conjunto de soluções para ele será apresentado no capítulo 3. O capítulo 4 descreverá a implementação destas soluções, os ambientes de testes utilizados, seus resultados e fará uma comparação com outras propostas. O capítulo 5 descreverá as limitações do trabalho e as conclusões obtidas. Por fim, serão apresentados os anexos e as referências que contribuíram para este trabalho.

Capítulo 2. BANCO DE DADOS DISTRIBUÍDOS, *GRIDS*, FRAGMENTAÇÃO DE DADOS E OS CÓDIGOS LDPC

"Não basta saber o que se vai dizer; é preciso também saber como se vai dizer."

Aristóteles

Este capítulo descreve alguns tópicos necessários para a compreensão do trabalho são eles: Sistemas de Banco de Dados, *Grids*, Fragmentação de Dados e uma breve descrição dos códigos LDPC.

2.1. Sistema de Gerência de Banco de Dados Distribuídos

Um Banco de Dados é um conjunto de dados organizados, compartilhados e integrados, que seguem um conjunto de regras de formação e conteúdo. Normalmente este conjunto de dados se encontra centralizado, ou seja, os dados estão armazenados em apenas um local.

No entanto, uma organização pode precisar distribuir seus dados. De acordo com [CASANOVA85], a idéia de se ter um SGBDD é muito atrativa, sob vários pontos de vista. Sob o ponto de vista administrativo, por exemplo, permite que setores geograficamente dispersos, de uma organização, mantenham controle de seus próprios dados, mesmo oferecendo um compartilhamento global no uso destes por outros setores da organização.

Segundo Kort [KORTH99], um Banco de Dados Distribuídos consiste em uma coleção de nós, e cada um destes nós mantém um Banco de Dados localmente. Cada nó pode executar consultas locais, assim como executar consultas que acessam dados que estão localizados em outros nós da rede.

Para se conseguir esta distribuição de dados, devem ser consideradas algumas questões, como o custo de instalação, o custo da comunicação (cabos ou linha telefônica, por exemplo), a segurança dos dados, assim como a disponibilidade (acesso aos dados mesmo com a falha de alguns nós).

Apesar dos dados serem distribuídos fisicamente, eles devem ser vistos de forma única para o usuário, permitindo o acesso aos dados de maneira transparente.

A primeira vez em que se utilizou o conceito de Sistema de Gerência de Banco de Dados Distribuídos (SGBDD) foi no final dos anos 70, por um grupo pioneiro da IBM, no projeto de pesquisa R* STAR. [DATE00]. Após um período de 10 anos de amadurecimento da tecnologia, a IBM publicou seu padrão DRDA (*Distributed Relational Database Architecture*), como uma proposta para integração de conjuntos de dados por meio de versões diferentes de DB2, que rodam em diferentes plataformas e sistemas operacionais como o AIX, OS/2, OS/400, VM e MVS [DATE00], A Figura 2 exemplifica uma arquitetura de Banco de Dados verdadeiramente distribuída [NAVATHE04].

2.1.1. REQUISITOS FUNCIONAIS DE UM SGBDD

C. J. Date [DATE00], propôs 12 regras que um SGBDD completo deveria seguir. Essas regras não representam requisitos absolutos; foram propostas somente para esclarecer as discussões sobre Sistemas de Bancos de Dados Distribuídos. Estas regras se tornaram largamente aceitas como um conjunto de definições de trabalho e como critérios de um Banco de Dados Distribuídos; no anexo-1 são apresentadas estas regras que servirão como parâmetro comparativo quando definirmos *Grids de Dados*.

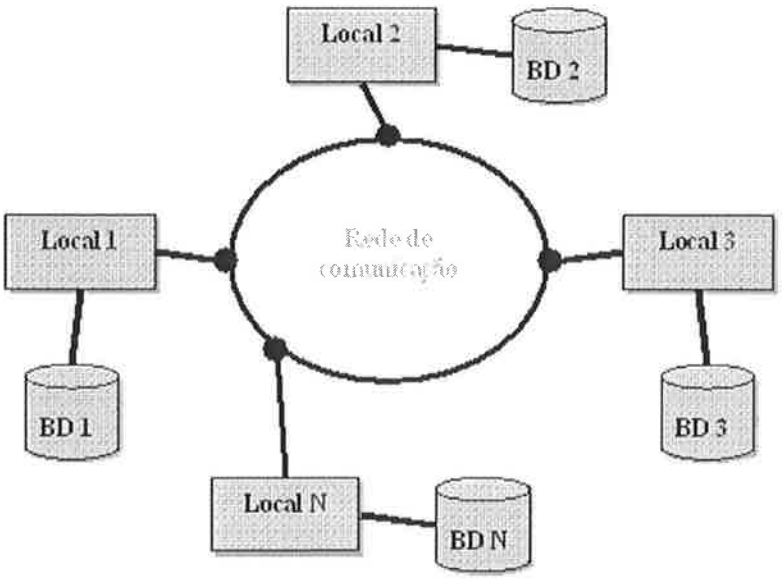

FIGURA 2. ARQUITETURA DE BANCO DE DADOS DISTRIBUÍDA.

2.1.2. VANTAGENS E DESVANTAGENS DE UM SGBDD

Há várias razões para que se construa um Sistema de Banco de Dados Distribuídos, incluindo compartilhamento de dados, confiabilidade e disponibilidade, aumento da velocidade de processamento de consultas. No entanto, junto com essas vantagens vêm várias desvantagens, incluindo o custo de desenvolvimento de software, grande potencial de erros, aumento da sobrecarga de processamento.

Vantagens:

- Compartilhamento de dados e controle distribuído;
- Confiabilidade e disponibilidade;
- Aumento da velocidade de processamento de consultas;
- Crescimento incremental.

Desvantagens:

A principal desvantagem de sistemas de Banco de Dados Distribuídos é a complexidade adicional requerida para assegurar a adequada coordenação entre os nós. Esta complexidade adicional toma a forma de:

- Custo do desenvolvimento de software;
- Grande potencial de erros e questões de impasse;
- Aumento da sobrecarga de processamento.

2.2. *GRIDS* COMPUTACIONAIS

Com a grande melhoria no desempenho das redes de comunicação nos últimos anos, tornou-se possível a utilização de computadores independentes conectados em redes como plataforma para execução de aplicações paralelas. A essas plataformas deu-se o nome de *Grids* Computacionais - uma alusão aos *Grids* Elétricos que disponibilizam energia elétrica sob demanda e escondem do usuário

detalhes como a origem da energia e a complexidade da malha de transmissão e distribuição [FOSTER01].

O alvo da tecnologia *Grid* é fornecer recurso computacional compartilhado, sem a necessidade dos usuários conhecerem os detalhes dos componentes computacionais, onde eles estão localizados, ou como são gerenciados ou atualizados [FOSTER01].

A infra-estrutura do *Grid* conecta hardware, armazenamento e software através de uma rede de alta velocidade, fornecendo acesso flexível e seguro para aplicações e dados.

A Figura 3 exemplifica um possível *Grid*, composto por computadores de vários tipos conectados via Internet. Um desses computadores pode estar realizando alguma tarefa específica, enquanto outro computador dispõe de grande capacidade para armazenamento de dados.

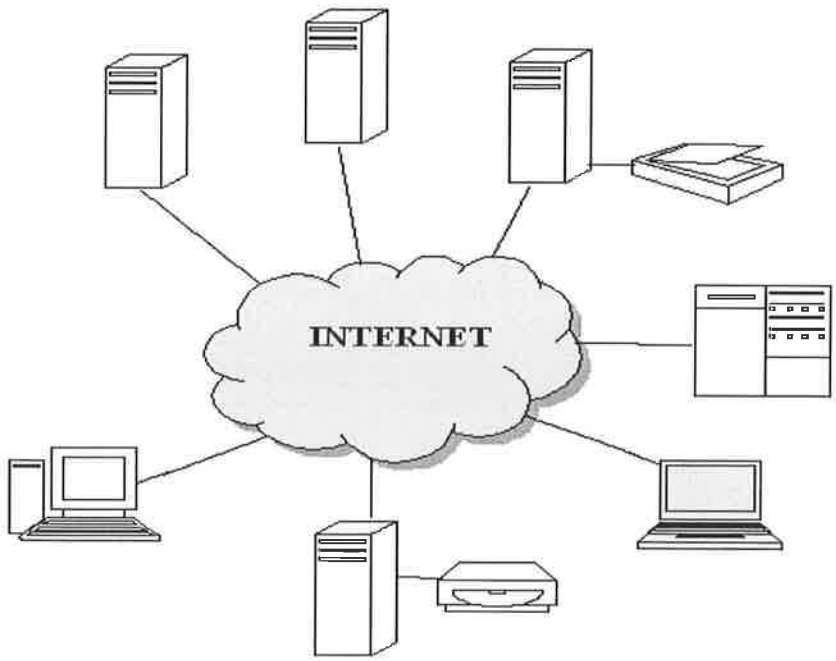

FIGURA 3. - *GRID* COMPUTACIONAL.

Diferentemente dos *Clusters*, *Grids* são mais distribuídos, diversos e complexos, e por vezes podem possuir *Clusters* como um dos seus componentes. Alguns aspectos que podem evidenciar estas características são [CIRNE02]:

- Heterogeneidade dos componentes do *Grid*;
- Alta dispersão geográfica;
- Compartilhamento ou diversidade de aplicações;
- Múltiplos domínios administrativos.

A Tabela 1 sumariza as características das plataformas de execução de aplicações paralelas. Contudo, esta tabela só descreve o comportamento "médio" ou "típico" dos diferentes tipos de plataformas de execução. Certas plataformas podem apresentar características arquiteturais adicionais que impactam na performance das aplicações paralelas que nela executam, não sendo o enfoque deste trabalho discuti-las.

Tabela-01. CARACTERÍSTICAS TÍPICAS DE DIFERENTES PLATAFORMAS DE EXECUÇÃO [CIRNE02].

	SMPs	MPPs	NOWs	Grids
Conectividade	excelente	muito boa	boa	média/ruim
Heterogeneidade	nula	baixa	média	alta
Compartilhado	não	não	sim	sim
Imagem	única	comum	comum	múltipla
Escala	10	1.000	1.000	100.000

Grids computacionais possuem controle distribuído, prevalecendo a autonomia local em detrimento da global.

Notamos que estes aspectos combinam com a definição para Sistemas de Banco de Dados Distribuídos proposta por C. J. Date [DATE00], apresentado no item anterior, onde uma única aplicação deve ser capaz de operar de modo

transparente sobre dados dispersos em uma variedade de Bancos de Dados, gerenciados por vários SGBDs diferentes, em execução em uma variedade de máquinas diferentes, admitindo uma variedade de sistemas operacionais e conectados entre si por uma variedade de redes de comunicações.

Conforme Roure [ROURE03] existem três principais questões que tem que ser confrontadas quando tratamos de *Grid*s:

- **Heterogeneidade**: um *Grid* envolve múltiplos recursos que são naturalmente heterogêneos e podem se espalhar por numerosos domínios administrativos através de uma extensão global;
- **Escalabilidade**: um *Grid* cresce de poucos recursos para milhões. Isto levanta o problema da potencial degradação de desempenho conforme o crescimento do tamanho do *Grid*. Conseqüentemente, aplicações que necessitam de grande número de recursos separados geograficamente devem ser projetados para serem tolerantes às latências de acesso e explorar os recursos acessados localmente. Além do aumento da escala também envolver um aumento no número de limites organizacionais, com ênfase na heterogeneidade e a necessidade de autenticação de endereços e questões de confiabilidade. Aplicações de grande escala podem ainda resultar da composição de outras aplicações, que aumentam a complexidade do sistema.
- **Adaptabilidade**: em um *Grid*, se um recurso falhar é uma regra não uma exceção. De fato, com alguns poucos recursos, em um *Grid* a probabilidade de alguns recursos falharem é naturalmente alta.

A gerência de recursos ou aplicações deve ser feita de maneira dinâmica para que se possa extrair o máximo de desempenho dos serviços e recursos disponíveis. Apesar de alguns padrões, tanto para *Grid*s como para Sistemas de Banco de Dados em *Grid*s ainda estarem em fase de definição pelo *Global Grid Fórum* (GGF) [MALAIKA03] [MOORE04] podemos, contudo, utilizar SGBD para ambientes em *Grid*s, visto que os aspectos e requisitos de ambos os sistemas são semelhantes,

conforme mostrado na seção anterior. Assim, um dos objetivos deste trabalho é realizar um estudo detalhado dos pontos positivos e negativos do emprego de SGBD em *Grid*, principalmente devido aos requisitos do *Grid* em relação ao tempo de resposta e as suas diferenças em relação a um Sistema de Banco de Dados Distribuídos.

2.2.1. Escalonamento de Aplicações no *Grid*

Em um Sistema de Banco de Dados Distribuídos o escalonamento de uma transação é realizado pelo servidor que disparou a transação, sendo este responsável pelo processamento e o sincronismo entre os outros servidores durante o tempo de vida da transação. Por outro lado, em *Grids* o escalonamento é realizado por um elemento do *Grid* que tem essa função pré-determinada. Esse elemento, escalonadores de recursos, controla os recursos do sistema (i.e., não se acessa os recursos do *Grid* sem a autorização do escalonador). Uma característica importante dos escalonadores de recurso é que eles recebem solicitações de vários usuários e, portanto, têm que arbitrar entre estes vários usuários o uso dos recursos que controlam.

Devido à grande escala, ampla distribuição e existência de múltiplos domínios administrativos, é praticamente impossível construir um escalonador de recursos para *Grids*. [CIRNE02]. Portanto, para utilizar recursos controlados por vários escalonadores de recurso distintos, alguém tem que:

(i) Escolher quais recursos serão utilizados na execução da aplicação;

(ii) Estabelecer quais tarefas cada um destes recursos realizará, e:

(iii) Submeter solicitações aos escalonadores de recurso apropriados para que estas tarefas sejam executadas.

Esta é a tarefa do escalonador de aplicação. Escalonadores de aplicação não controlam os recursos que usam. Eles obtêm acesso a tais recursos submetendo solicitações para os escalonadores que controlam os recursos.

Logo, em um *Grid*, as decisões de escalonamento são divididas em duas camadas, com parte da responsabilidade pelo escalonamento sendo transferida dos escalonadores de recurso para o nível de aplicação.

2.3. *GRIDS* DE DADOS E BANCO DE DADOS EM *GRIDS*

Como a tendência de crescimento exponencial no armazenamento de informações devem se manter, pelo menos, a curto e médio prazo, a solução natural para tratar desses volumes crescentes de dados é a utilização de processamento paralelo e distribuído. Conforme Walfredo Cirne [CIRNE02], as soluções adotadas para esse problema costumam ser fracamente acopladas, o que as tornam adequadas para execução em *Grid*s computacionais.

Uma das principais funções de um SGBD é o armazenamento, manipulação e disponibilização de grandes volumes de dados. *Datas Grids também prometem* a manipulação e disponibilização de grandes volumes de dados em escala global. Portanto, a integração entre essas duas tecnologias é desejável [LIU03], [MOORE04].

2.3.1. *GRIDS* DE DADOS

Schoreder e Moore [SCHOREDER04] afirmam que *Grid*s são infra-estruturas emergentes para coordenar acesso entre organizações autônomas compartilhando recursos e dados heterogêneos. *Grid*s de Dados estão sendo construídos ao redor do mundo como a próxima geração de sistemas de manipulação de dados para compartilhamento, publicação e preservação de dados colocados em sistemas de armazenamento localizados em múltiplos domínios administrativos. Um *Grid* de Dados fornece nomes de locais lógicos para usuários, entidades digitais e recursos de armazenamento para criar identidades persistentes para controle de acesso, disponibilizando descoberta e gerenciando latência em grandes áreas.

O maior desafio em projetos de sistemas de gerenciamento de dados está em integrar um conjunto de múltiplos requisitos impostos por comunidades de usuários. Como já mencionamos, a quantidade de dados está crescendo exponencialmente em número de entidades digitais e nos tamanhos dos arquivos. As fontes de dados estão distribuídas em múltiplas localidades, com dados gerados em múltiplos domínios administrativos e em locais acessíveis por uma grande rede de dimensões globais. A necessidade de encontrar e unir esses dados torna-se muito importante, a fim de disponibilizá-los em coleções para serem pesquisadas. A sua preservação também é importante, onde a reunião das necessidades de retenção de dados legais e a preservação do capital intelectual das organizações são fundamentais.

Na prática, seis tipos de sistemas de manipulação de dados são encontrados [SCHOREDER04]:

1. Sistemas de coleta de dados;
2. Ambiente de criação de coleções de dados;
3. Ambiente de dados compartilhados baseado em *Grid*s;
4. Bibliotecas digitais para publicação de dados;
5. Arquivos persistentes para preservação de dados;
6. *Pipelines* para processamento de dados.

Novamente, citando Schoreder [SCHOREDER04], o alvo de um sistema de gerenciamento de dados genérico é construir uma infra-estrutura de software que possa unir as necessidades das comunidades virtuais. *Grid*s de Dados fornecem uma abstração de gerenciamento de dados genérica necessária para gerenciar dados distribuídos, onde todos os sistemas possam ser construídos sobre um software comum.

O gerenciamento de dados em recursos distribuídos necessita de quatro convenções básicas, são elas:

1. Gerenciamento dos recursos para acessar sistemas de armazenamento através de domínios administrativos distintos. Isto é usado para implementar virtualização de dados armazenados;
2. Distinção dos nomes de usuários para identificação de pessoas através de domínios administrativos. Isto é usado para implementar conexão simples no ambiente de segurança;
3. Distinção dos nomes de arquivos para identificação de arquivos através de domínios administrativos. Isto é usado para implementar virtualização de dados;
4. Definição do contexto dos atributos para gerência do estado da informação gerada por processo remoto. Isto é usado para implementar bibliotecas digitais e *Grid*s de Dados Federados.

Tópicos relativos a dados no ambientes de *Grid*s podem ser classificados dentro das seguintes categorias conforme o GGF (*Global Grid Forun*) [MALAIKA03]:

- **Virtualização** de dados no *Grid*: significa que aplicações têm a habilidade de Descobrimento, Acesso e Atualização de dados independente do formato dos dados e de onde eles estejam localizados no *Grid*;
- **Gerenciamento** de dados no *Grid*: significa a certeza da disponibilidade dos dados para aplicações com características de desempenho apropriado no ambientes de *Grid*. Assim, recursos de dados podem tomar parte na qualidade total de serviço em mecanismos de provisionamento de sistemas dos *Grid*s escaláveis;
- **Integração** de Dados dentro de uma infra-estrutura de *Grid*: significa ter a certeza de que o sistema de gerenciamento de dados atende aos mecanismos do *Grid*. Os exemplos incluem: assegurar que os sistemas de recursos de dados continuem fornecendo informação de contexto necessária, emprego apropriado de rastreamento, monitoramentos, detecção de erros e informações de contabilização, tornando-se uma parte integral do *Grid*.

Um *Grid* de Dados deve fornecer mecanismos de virtualização para recursos, usuários, arquivos e metadados. Cada mecanismo de virtualização implementa localização e infra-estrutura independente do espaço de nomes (*name space*) que fornece identificadores independentes. Os identificadores independentes para dados são organizados como uma coleção hierárquica e chamado de espaço de nome lógico (*logical name space*). Na prática, os espaços de nomes lógicos são implementados em catálogos de metadados separados para cada *Grid* de dados. Dentro de um *Grid* de Dados acesso para dados, gerenciamento de dados e manipulação de dados é feito via comandos aplicados no espaço de nome lógico.

2.3.2. BANCO DE DADOS EM *GRIDS*

Conforme Atkinson [ATKINSON 03], um serviço de Banco de Dados pode ser definido como qualquer serviço que suporta uma interface a esse Banco de Dados. Serviços de interface são abstratos e não descrevem como suas capacidades são tratadas. Assim, serviços de Banco de Dados específicos podem fornecer acesso para Banco de Dados Relacional, Banco de Dados Objeto e repositórios XML ou Sistema de Armazenamento Especialista (i.e. para armazenar matrizes ou seqüência de DNA).

Podemos considerar para este trabalho que o principal serviço será um Serviço de Banco de Dados para *Grid* (*Grid Database Service*). Este serviço fornece capacidade para consultas, atualizações e envolve um Banco de Dados. Contudo interfaces são descritas também para:

- **Entrega de Dados (*Data Delivery*)**: fornecer facilidades para transmissão de dados estruturados. Tal serviço difere dos mecanismos de transferência de arquivos em que a estrutura interna dos dados pode ser levada em conta dentro do serviço;

- **Transações** (***Transactions***): fornecem facilidades para coordenação de coleções de operações, por exemplo, como uma visão para controlar interfaces entre usuários concorrentes;
- **Metadados de Banco de Dados** (***Database Metadata***): fornecem acesso para as informações sobre as capacidades de um serviço de Banco de Dados e seus dados.

O fornecimento de um serviço de Banco de Dados para *Grid*, dentro do contexto da OGSA (*Open Grid Service Architecture*), [MALAIKA03] envolve:

- Descrição de interfaces que ampliam as suas partes já fornecidas da OGSA;
- Delimitação dos serviços que importam suas interfaces;
- Descrição de como são obtidos os serviços dentro da infra-estrutura fornecida pelo OGSA.

Em [MALAIKA03], a **DAIS-WG** (*Data Access and Integration Services Working Group*) tem o objetivo de produzir uma especificação que descreva uma interface baseada em serviço para acessar e integrar dados em Banco de Dados Relacional e Repositórios XML existentes. Características essas incluídas e que são muito úteis em aplicações em *Grid*s, e que podem ser implementadas fora do Sistema de Banco de Dados, sem modificá-lo. Não obstante, os requerimentos que têm sido coletados das comunidades científicas, em princípio estas características incluem:

i. Nomear resultados para subseqüente uso;
ii. Obter resultados em múltiplos formatos;
iii. Quebrar grandes quantidades de dados;
iv. Entregar os resultados de maneira assíncrona;
v. Entregar os resultados para uma terceira parte.

As especificações da **DAIS-WG** envolvem gerência de recursos tentando aderir aos princípios e infra-estrutura descritos na **OGSI-WG** (*Open Grid Services*

Infrastructure Working Group) [MALAIKA03]. As especificações submetidas ao GGF (*Global Grid Forun*) incluem as seguintes construções no modelo:

- **Gerência de Recursos de Dados**: representa o Sistema de Gerência de Banco de Dados;
- **Recursos de Dados**: representa um Banco de Dados (tabelas para um Banco de Dados Relacional ou coleções para um repositório XML);
- **Seção de Acesso aos Dados**: representa o relacionamento entre um cliente e um recurso de dados;
- **Requisição de Dados**: contém as requisições SQL, *XPath* ou *XQuery* para serem executadas;
- **Conjunto de Dados**: representam os formatos de saída dos resultados.

A **DAIS-WG** opera em termos de tópicos, para disponibilizar grupos pequenos ou individuais nas seções de especificação e para explorar áreas relacionadas. Estes tópicos incluem:

- **Transformação**: que disponibiliza a transformação de resultados;
- **Procedimentos Armazenados**: que descrevem como parâmetros e conjunto de resultados podem ser manipulados;
- **Segurança**: que descrevem como o Banco de Dados e o *Grid* devem interagir;
- **Transações**: que descrevem como o Banco de Dados e as transações interagem no ambiente *Grid*;
- **Metadados**: descreve os estilos de entrega e de acessos a dados no *Grid*.

2.3.2.1. CARACTERÍSTICAS DE UM SISTEMA DE GERÊNCIA DE BANCO DE DADOS EM *GRIDS*

Conforme [MALAIKA03], **DAIS-WG** especifica que a Gerência de Recursos de Dados tem um serviço que corresponde a qualquer execução de gerência de recursos, por exemplo:

- **Gerência de Recursos Externos**: nome do produto, versão e nome do vendedor;
- **Propriedades Físicas**: construção física inclui espaço usado, *buffers*;
- **Capacidades do Software**: capacidades dos softwares incluem versão do software, atualizações e nível de *patch*;
- **Características Instaladas**: descrição das características instaladas; inclui melhoria no produto, características e funcionalidades extras.

A **DAIS-WG** ainda especifica que os Recursos de Dados podem ser de dois tipos:

- **Recursos de Dados Relacional**: onde o conteúdo é armazenado em tabelas relacionais. Resultados de uma requisição SQL são retornados em um conjunto de linhas de uma tabela;
- **Recursos de Dados XML**: onde o conteúdo é armazenado em coleções XML. Resultados de uma requisição *XQuery* são retornados como uma seqüência de itens.

Para representar o aspecto estrutural de Recursos de Dados Relacional como serviços de dados, descrição do Banco de Dados, tabelas, colunas, tipos de dados, restrições, gatilhos, procedimentos armazenados etc. Esses aspectos também são necessários para repositórios em XML.

Os padrões SQL não fornecem tais descrições em XML. Contudo, os próximos dois padrões podem fornecer tais representações [MALAIKA03]:

- **Esquema de informação SQL** -- fornece as descrições sobre 60 visões relacionais de informações do catálogo do banco de dados. Estas visões são compreensíveis e incluem visões para representação de: Colunas, Integridade Referencial, Rotinas, Características do SQL, Linguagem SQL e Tabelas;
- **SQL/XML** – fornece um cabeçalho XML padrão para dados relacionais. Para ver os esquemas. XML definidos pelo SQL.

2.3.2.2. CONTEÚDO DO BANCO DE DADOS EM *GRIDS*

Para representar um conteúdo de um Banco de Dados Relacional, no contexto de tabelas relacionais o **ISO SQL/XML** fornece as especificações que devem conter um esquema de Banco de Dados para poder ser usado.

Não existe notação para representar documentos XML no contexto de coleções que são armazenadas. Existem algumas prováveis direções na área de dados e computação em *Grid*:

- **Arquivos e Banco de Dados**: Requerimentos estão emergindo para tratar arquivos de dados e dados em Banco de Dados como um todo. Algumas das características e interfaces que são relevantes para dados no geral e podem ser descritos Recursos de Dados que representam arquivos e sistemas de arquivos;
- **Sistemas de Arquivos e Banco de Dados Adaptados**: As atividades de padronização até aqui têm focado em disponibilizar banco de dados, sistemas de arquivos e interfaces em sistemas de suporte que são adaptados para o ambiente *Grid* são prováveis para serem padronizados;
- **Grupos no GGF área de Dados**: Tecnologias que são comuns para todos os tipos de dados incluem metadados, anotações, federações, descobrimento, movimento, etc.

Essas últimas especificações justificam a utilização, neste projeto, de um Banco de Dados não definido especificamente, mas adaptado para a sua utilização em ambientes de *Grid*.

2.4. Fragmentação de Dados em SGBD Distribuídos

Um ponto importante a ser considerado é como fornecer condições para que o processamento em paralelo e distribuído dos dados seja possível e eficaz. Ao considerarmos a utilização de um SGBD em ambiente *Grid*, temos que considerar também o projeto de distribuição de dados. Para isso, consideremos uma relação (tabela) **r** que será armazenada em um banco de dados.

Há diversas questões envolvidas no armazenamento em um Banco de Dados Distribuídos, incluindo: Replicação de Dados, Fragmentação de Dados e Replicação e Fragmentação de Dados.

Ao adotarmos uma solução, temos que ter em mente a autonomia local de cada sítio exigida por ambos os ambientes, distribuídos e nos *Grid*s e, portanto, temos que adotar uma solução que não viole estas autonomias, como já mencionado anteriormente.

Na **Replicação de Dados**, o sistema mantém várias réplicas idênticas de uma relação. Cada réplica é armazenada em um nó diferente, resultando em replicação de dados. Se a relação **r** é replicada, uma cópia sua é armazenada em dois ou mais nós. Em casos mais extremos, há a replicação total, em que uma cópia é armazenada em todos os nós do sistema.

Em [LOURENSO01] verificamos que a replicação de objetos sob demanda, além de aumentar a disponibilidade do objeto realmente melhora o balanceamento de carga. Contudo, um dos problemas verificados no caso dos objetos possuírem um tamanho muito elevado (por trabalharmos com imagens de alta definição com

MBytes de tamanho), foi a demora na replicação do objeto. Além disso, o espaço necessário para seu armazenamento e de suas cópias acabava tornando-se muito custosa para o sistema.

Na **Fragmentação de Dados**, uma relação (tabela) poderá ser particionada em diversos fragmentos que serão armazenados em diferentes nós.

Se uma relação **r** é fragmentada, **r** é dividida em um número de fragmentos **r1, r2,..., rn**, esses fragmentos contêm informações suficientes para reconstruir a relação **r** original. Essa reconstrução pode ser feita através da aplicação de uma operação de união ou um tipo especial de junção dos vários fragmentos.

Existem dois esquemas básicos para se fragmentar uma relação **r**:

- Fragmentação Horizontal;
- Fragmentação Vertical.

A **Fragmentação Horizontal** divide a relação pela associação de cada tupla (linha) de **r** a um ou mais fragmentos. A relação **r** é particionada em um número de subconjuntos, **r1, r2,..., rn**. Cada tupla da relação **r** deve pertencer a um dos fragmentos, para que a relação original possa ser reconstruída se necessário. Um fragmento pode ser definido como uma seleção sobre a relação global **r**. Isto é, um predicado **Pi** é usado para construir o fragmento ri da seguinte forma:

$$r_i = \sigma_P(r)$$

A reconstrução de uma relação **r** pode ser obtida tomando a união de todos os fragmentos, isto é:

$$r = r_1 \cup r_2 \cup \ldots \cup r_n$$

A **Fragmentação Vertical** divide a relação **r** pela decomposição de determinadas colunas em diferentes fragmentos. Cada fragmento **ri** de **r** é definido por:

$$r_i = \Pi_R(r)$$

A relação **r** pode ser reconstruída a partir da junção natural dos fragmentos:

$$r = r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

Geralmente, a fragmentação vertical é completada pela adição de um atributo especial, chamado de identificado da tupla (*tuple-id)*. Um identificador da tupla é um endereço físico ou lógico de uma tupla, que será utilizado para fazer a junção natural dessa tupla, possibilitando a reconstrução da relação original.

Temos ainda a utilização da **Fragmentação Mista** onde cada fragmento é obtido como resultado da aplicação dos métodos para Fragmentação Horizontal e Vertical na relação **r**, ou em um fragmento de **r** que foi previamente obtido.

Podem-se ainda combinar as técnicas de **Fragmentação** com a técnica de **Replicação** de dados e aplicá-las sucessivamente na mesma relação. A relação é particionada em vários fragmentos. O sistema mantém várias cópias idênticas de cada fragmento. Isto é, um fragmento pode ser replicado, as réplicas do fragmento podem ser fragmentadas, e assim por diante.

As soluções utilizando fragmentação são freqüentemente adotadas como um meio para se conseguir um melhor desempenho do Sistema de Banco de Dados Distribuídos pela redução de acesso a disco necessário para executar uma aplicação, minimização do número de acessos a objetos irrelevantes, e redução da transferência de dados entre sites. Normalmente, tais metodologias são utilizadas como auxílio ao projetista de Banco de Dados na distribuição dos dados pelo ambiente distribuído [BAIÃO00].

Em [BAIÃO04], os autores apresentam um estudo de alocação de dados em Banco de Dados Distribuídos. Eles abordam os problemas decorrentes desta alocação

de dados que envolvem encontrar a distribuição "ótima" dos dados através de nós da rede, de forma a minimizar o custo das aplicações sobre esses dados. Neste estudo, destaca-se a criticidade desta alocação num Sistema de Banco de Dados Distribuídos (SBDD), já que uma alocação ineficiente dos dados pode levar a um aumento considerável do custo de acesso ao SBDD. Em geral, em um projeto de distribuição da base de dados, a fase de alocação é realizada após a fase de fragmentação, a qual determina a distribuição dos dados em fragmentos.

O principal aspecto que determina a qualidade da solução encontrada para o problema de alocação de dados é a eficiência do projeto de alocação, ou seja, a alocação dos dados nos nós deve minimizar, tanto quanto possível, o custo das consultas executadas [BAIÃO04].

O processo de tomada de decisão presente no Problema de Alocação de Dados (PAD) em um SBDD, de uma forma geral, pode ser definido como: escolher dentre todos os possíveis modelos de alocação de fragmentos nos nós de uma rede, (ou seja, entre todas as possíveis distribuições dos fragmentos), aquela que minimize o custo de acesso aos dados alocados, considerando um conjunto de restrições do problema, (como por exemplo, a capacidade dos nós onde os dados são alocados). Este problema inclui a decisão de replicação dos fragmentos e seu tipo é classificado na literatura como Problema de Otimização Combinatória [BAIÃO04].

Em soluções como a utilizada em [BAIÃO04], temos uma combinação de fragmentação e replicação dos fragmentos. Essa solução pode ser aplicada sucessivamente na mesma relação ou classe. A relação ou classe é particionada em vários fragmentos. O sistema mantém várias cópias idênticas de cada fragmento. Isto é, um fragmento pode ser replicado; as réplicas do fragmento podem ser fragmentadas, e assim por diante.

2.5. Os códigos LDPC (*Low Density Parity Check*)

A seguir será apresentada uma pequena introdução dos códigos LDPC que foram testados no nosso trabalho.

2.5.1. Introdução - LDPC

Conforme descrito em [PLANK04] [RYAN04], Checagem de Paridade de Baixa Densidade (LDPC) representam um conjunto de classes de códigos de blocos de códigos lineares que fornecem melhor capacidade de desempenho na transmissão de uma grande coleção de dados e canais de armazenamento enquanto simultaneamente admite decodificadores implementados.

Conforme Ryan [RYAN04] os códigos LDPC foram inicialmente propostos por Gallager, em 1960, em uma tese de doutorado e foram pouco considerados nos 35 anos seguintes. Uma notável exceção é o importante trabalho de Tanner em 1981 onde ele generalizou os códigos LDPC e introduziu uma representação gráfica desses códigos, chamados de Grafos Tammer. Os estudos dos códigos LDPC foram retomados no meio de 1990 com o trabalho de MacKay, Luby, e outros que noticiaram, aparentemente com independência do trabalho de Gallager, as vantagens dos blocos de código linear possuindo matrizes de checagem de paridade esparsas (baixa densidade).

Os códigos Tornado são instâncias de classes dos códigos chamados códigos LDPC, que têm uma longa história desde os anos 60. Contudo, as pesquisas sobre estes códigos foram retomadas a partir da metade dos anos 90. Em 1998 essa pesquisa tomou dois caminhos – Pesquisas acadêmicas e Industriais resultando em muitas publicações e patentes [PLANK04].

No caso dos códigos LDPC, esses são baseados em grafos, com a finalidade de definir uma base de códigos para operações de paridade. Todas as pesquisas sobre

códigos LDPC têm sido direcionadas para a definição de códigos que aumentem a "capacidade do canal de comunicação" de maneira assintoticamente. Em outras palavras, eles definem códigos onde o fator determinante é a redução da sobrecarga de processamento. Outras propostas são baseadas em códigos com grafos regulares, tendo cada nó uma cardinalidade constante de entradas e saídas. Portanto, os melhores códigos são baseados em grafos irregulares gerados aleatoriamente. Uma classe de grafos irregulares é definida baseada em distribuição de probabilidade da cardinalidade de nós e, então, propriedades são atribuídas sobre as características deste conjunto de classes.

Novamente citando Ryan, existem três tipos distintos de códigos LDPC descritos na literatura acadêmica. Todos são baseados em grafos bipartidos gerados aleatoriamente a partir de uma distribuição de probabilidade. O detalhamento desses códigos está no anexo 2.

2.6. CONCLUSÃO

Neste capítulo foram apresentados os conceitos estudados e considerados no desenvolvimento do projeto. Muitos desses conceitos, de alguma forma, envolvem ou se relacionam com o sistema desenvolvido. A técnica de segmentação de dados utilizada em *Grid*s mostrou-se difícil de ser utilizada devido aos inúmeros fatores que estão envolvidos e relacionados entre si. Entretanto, o sistema foi implementado de forma a receber, futuramente, melhorias.

Com base neste capítulo será apresentado o problema e como ele foi abordado, bem como a modelagem do sistema proposto no capítulo 3. Nele, serão encontradas as decisões tomadas para a implementação de um processo de segmentação de atributos complexos em um Sistema de Banco de Dados baseado em *Grid*s de dados.

Capítulo 3. O Problema da Fragmentação de Dados e a Proposta de Distribuição da Base de Dados em *Grid*s

"Tudo que uma pessoa pode imaginar, outras podem tornar real."

Julio Verne

Atualmente uma das áreas promissoras de sistemas computacionais para inúmeras aplicações é o *Grid* Computacional (*Computational Grids*). Basicamente ele consiste de uma plataforma heterogênea de computadores geograficamente dispersos, interligados por uma rede de comunicação de alta velocidade (Internet 2), onde os usuários fazem acesso ao ambiente através de uma interface única, visando à obtenção de recursos compartilhados.

Este trabalho implementa um ambiente de gerenciamento de Banco de Dados Objeto Relacional baseado em *Grid*s de computadores para sistemas de Armazenamento de Alto Desempenho. Dois fatores são importantes em relação à base de dados: distribuição e compartilhamento. A **distribuição** determina quais e quantos dados serão mantidos por um determinado computador. Já o **compartilhamento** consiste na forma como os dados remotos serão acessados pelos computadores e pode ser feito pela migração ou replicação da base de dados, ou parte dela. Como o tamanho das bases de dados costuma ser grande, a replicação de dados será evitada e conseqüentemente apenas a migração de dados será utilizada como solução para o compartilhamento.

Os processos de segmentação e/ou recomposição dos dados são executados nos computadores do usuário local do servidor que estiver participando do *Grid*. Os segmentos gerados serão então distribuídos ou resgatados, dependendo do processo executado (Figura 4). Não serão criadas cópias dos segmentos gerados e também cada servidor do *Grid* só receberá um segmento de um mesmo documento evitando com isso a manutenção de replicas ou que um servidor armazene mais segmentos de um mesmo documento.

Algumas características esperadas são apresentadas a seguir:

a) Evitar qualquer tipo de migração de segmento desnecessária;
b) Evitar replicação dos segmentos, assim não existe uso desnecessário de espaço de armazenamento.

A segmentação de um dado documento, bem como a distribuição do mesmo ou de seus segmentos, está associada à quantidade de servidores que estiverem disponíveis no momento em que o documento estiver sendo inserido. Os processos de identificação, segmentação e distribuição são totalmente independentes, não permitindo ao usuário ter ciência de como e onde o documento está sendo armazenado.

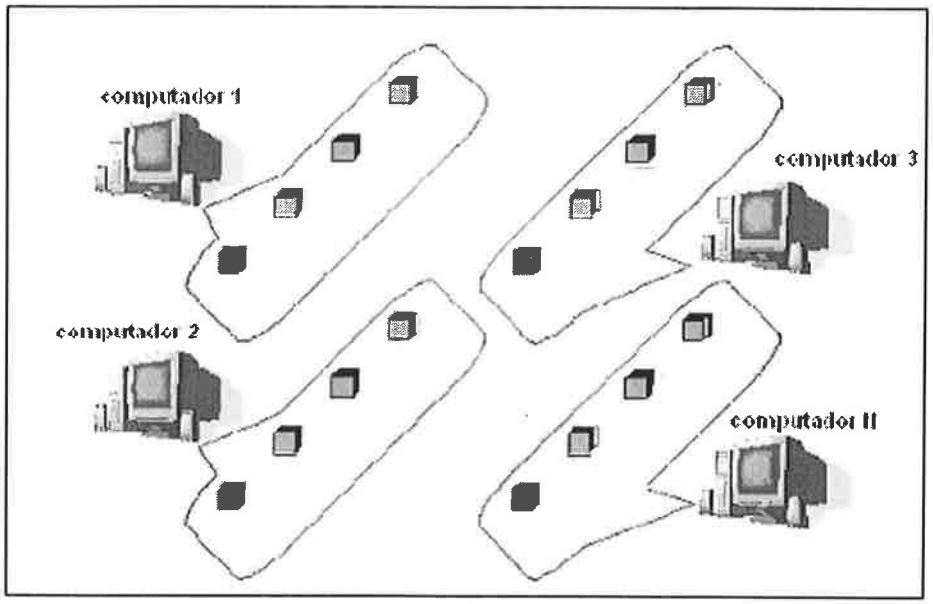

FIGURA 4. DIVISÃO DOS SEGMENTOS ENTRE OS COMPUTADORES DO *GRID*.

3.1. DESCRIÇÃO DO PROBLEMA

Notamos que, na maioria dos trabalhos, metodologias e mesmo na literatura, só são considerados os fragmentos de uma relação (tabela) de Banco de Dados Relacional ou Objeto Relacional [BAIÃO04] [OZSU99], e classe de objetos de Banco de Dados Orientado a Objetos [BAIÃO00] [MATTOSO98] [BAIÃO98]. No caso de uma determinada instância de uma relação ou classe de objetos possuir um atributo complexo (não-estruturado ou semi-estruturado), cujo conteúdo possa diferir em tamanho e/ou tipo de uma instância para outra, esse atributo será enviado junto com o fragmento ao qual está associado. No caso de uma consulta ser realizada,

recuperando todos os fragmentos e seus atributos, o servidor que estiver de posse desse atributo se sobrecarregará tornando o esquema empregado de fragmentação ineficiente. Nesse caso, não estamos considerando que o documento, ou melhor, o objeto tenha sido previamente segmentado pelo usuário.

Como já mencionado na seção 1.5, Ozsu apresenta o projeto de um Banco de Dados Orientado a Objetos específico para vídeo, onde o vídeo é segmentado previamente em *clips* com intervalo de tempos iguais. Similarmente a segmentação, o tamanho de cada segmento do vídeo também é definido previamente. A finalidade dessa proposta é tornar disponível consultas baseadas em características semelhantes contidas em cada *clip* do vídeo para a sua reconstrução.

Tomando este cenário como ponto de partida, verificamos que propostas tradicionais são inviáveis para certos tipos de aplicações não convencionais e para Banco de Dados relacionais tradicionais [OZSU99], contudo, estas aplicações estão se tornando cada vez mais comuns [LOURENSO00].

Uma proposta para estes problemas é a utilização de um Banco de Dados que permita gerenciamento de objetos conforme abordado por Ozsu [OZSU99]. Apesar da abordagem de Ozsu ser direcionada para o modelo de Banco de Dados Orientado a Objetos, como vemos também em seu outro trabalho [OZSU02], pode-se adotar o Modelo de Dados Objeto-Relacional [KORT99]. O modelo Objeto Relacional fornece um sistema rico, incluindo orientação a objeto com linguagem de consultas relacional, tal como SQL conforme o tipo de dados. Tal extensão tende a preservar o acesso declarativo a dados, enquanto estende o poder da modelagem.

Sistemas de Banco de Dados Objeto Relacional fornecem um caminho conveniente de migração do modelo de Banco de Dados Relacional para quem deseja utilizar modelo Orientado a Objetos [DATE00]. Contudo, para o nosso trabalho não será necessário adotarmos um SGBD que incorpore todo o padrão SQL/3, ou seja, o SGBD deve incorporar a parte do padrão que define o armazenamento de BLOBs (*Binary Large Objects*).

Para solucionar o problema de segmentação de um atributo atômico, contido em uma relação ou instância de classe, estamos propondo a extensão do método de fragmentação de dados vertical, a fim de possibilitar a segmentação de um único atributo em pedaços de tamanhos definidos pela simples divisão do seu tamanho pela quantidade de servidores disponíveis.

Vamos também avaliar a técnica de codificação baseada nos códigos LDPC, já utilizada para arquivos de dados [PLANK04]. Esta técnica permite que arquivos sejam quebrados em segmentos e, quando esses arquivos forem recompostos, será possível checar sua integridade no momento da sua decodificação.

Os códigos da família LDPC possibilitam que, após um determinado arquivo de dados ser segmentado, cada sítio do sistema controle uma pequena quantidade deste arquivo. A finalidade desse método é a de possibilitar o aumento da disponibilidade de um arquivo sem, no entanto, aumentar também a utilização do espaço em disco. Essa técnica, porém, pode ocasionar um aumento na carga de processamento local devido ao método utilizado, que codifica e decodifica o arquivo durante o armazenamento e recuperação, respectivamente. Contudo, esse processamento pode ser considerado relativamente baixo se utilizarmos uma arquitetura multiprocessada. Um outro ponto, nessa técnica, a ser considerado é a necessidade de recuperar todo arquivo para só então verificar se ele é realmente o arquivo desejado.

Na nossa proposta, segmentaremos documentos de tipos variados, que ultrapassem um tamanho específico, determinado após os testes do sistema. Tanto a segmentação do documento como sua distribuição serão automáticas e, portanto, será necessária também a geração dos metadados relativos aos segmentos gerados. Os metadados devem conter informações suficientes para a recuperação e recomposição do documento original quando e no momento em que requisições forem realizadas aos documentos armazenados.

A proposta deste trabalho é implementar uma infra-estrutura que possibilite a execução de uma aplicação em um ambiente de *Grid* computacional tentando obter economia de recursos disponíveis neste *Grid*, como espaço em disco e largura de banda de comunicação, sem, no entanto, perder disponibilidade do documento requisitado. Outro ponto é o possível emprego desta proposta nas metodologias de fragmentação vertical de dados em uso.

Adotamos um ambiente baseado em *Grid* de servidores, interligados por uma rede de alta velocidade. Neste *Grid*, foi instalado um SGBD Objeto Relacional[1]. O cenário adotado para o desenvolvimento deste trabalho será detalhado no capítulo 4.

Iniciamos com o estudo do problema de fragmentação de dados, e gerenciamento de cópias de dados num SGBD Objeto Relacional, instalado em diversos servidores independentes, como seria o caso em um *Grid*.

Um dos principais pontos a ser validado no trabalho é se, com a segmentação desses atributos complexos, conseguiremos uma otimização dos recursos do *Grid* (espaço de armazenamento, banda de comunicação e balanceamento de carga), para um tipo específico de aplicação que envolva a demanda desses tipos de documentos. O motivo dessa abordagem é que, com a segmentação desses atributos, o conteúdo e a demanda por estes dados não ficarão sujeitos a um único servidor e, portanto, tanto a carga como a responsabilidade por estes dados poderão ser distribuídas entre os servidores do *Grid*.

Resumindo, foi desenvolvida, implementada e avaliada uma infra-estrutura que permite a segmentação de um atributo complexo (não-estruturado) em um SGBD Objeto Relacional[1] baseado em *Grid* com disseminação dinâmica dos segmentos e geração automática de metadados desses segmentos.

---

[1] **SGBD Objeto Relacional que forneça, no mínimo, armazenar BLOBS**

3.1.1. SEGMENTAÇÃO DE OBJETOS EM UM BANCO DE DADOS OBJETO RELACIONAL BASEADO EM *GRID*

Neste trabalho foi proposta e implementada uma infra-estrutura que permite a geração de segmentos de atributos complexos de uma instância de uma relação em um SGBDOR baseado em *Grid*, com geração automática de metadados desses segmentos. A finalidade é complementar as metodologias de modelagem e fragmentação de dados, pela incorporação da possibilidade de redução da granularidade da fragmentação vertical no nível de um atributo (não-estruturado ou semi-estruturado) contido em uma instância de uma relação. Com este procedimento espera-se obter um melhor desempenho dos sistemas que utilizam Banco de Dados em *Grid* e que necessitam fornecer documentos complexos, bem como otimizar o espaço de armazenamento ocupado por estes tipos de documento. Além disso, buscar-se-á avaliar a capacidade deste procedimento de proporcionar uma melhor distribuição de carga neste *Grid*.

3.2. DETALHAMENTO DO PROBLEMA X PROPOSTAS

Considere uma relação (tabela) de projetos (Proj) da Figura 5. Utilizando o método de fragmentação horizontal, baseado no número do projeto (NoProj), obtém-se quatro fragmentos (Figura 6), que poderão ser distribuídos por quatro nós de um *Grid* (Figura 7).

Proj			
NoProj	**NmProj**	**Orçamento**	**Local**
P1	Instrumentação	150.000	Montreal
P2	Projeto de BD	135.000	Nova York
P3	CAD/CAM	250.000	Nova York
P4	Manutenção	310.000	Paris

FIGURA 5. RELAÇÃO (TABELA) DE PROJETOS.

Para este exemplo, considere uma relação com tipos de dados tradicionais e que não necessitam de nenhuma estrutura especial para armazená-los, utilizando um método de fragmentação horizontal simples dessa relação, e necessário apenas uma operação de união para reconstruir a relação original.

Assim para a relação **Proj** com a fragmentação

Fproj = {Proj1 , Proj2, ...Projn}, Proj = U Proji, V Proji ϵ Fproj.

Proj1

NoProj	NmProj	Orçamento	Local	TAD
P1	Instrumentação	150 000	Montreal	Doc

Proj2

NoProj	NmProj	Orçamento	Local	TAD
P2	Projeto de BD	135 000	Nova York	SQL

Proj3

NoProj	NmProj	Orçamento	Local	TAD
P3	CAD/CAM	250 000	Nova York	CAD4

Proj4

NoProj	NmProj	Orçamento	Local	TAD
P4	Manutenção	310 000	Paris	Doc

Figura 6. Particionamento Horizontal da Relação Proj.

Neste caso, com a distribuição adotada dos dados, ocorreu também uma distribuição do processamento e do armazenamento. Contudo, no caso da relação **Proj** possuir um item a mais com um tipo de dado complexo, (Figura 8), onde o conteúdo possa variar de projeto para projeto em tamanho e tipo, a solução utilizando fragmentação horizontal não será mais adequada.

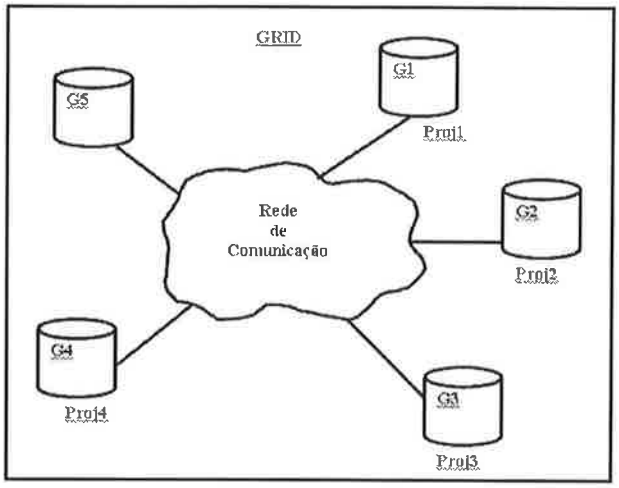

Figura 7. Distribuição dos projetos no *Grid*.

Considerando cada projeto individualmente, nota-se que o projeto **P3** (**NoProj** = **P3**) possui o campo **TCD** = **CAD** (Figura 8), este documento em relação aos demais documentos contidos no **TCD** será mais complexo e possivelmente terá um tamanho muito maior. Qual procedimento ou método então se pode adotar para realizar uma distribuição eqüitativa dos dados no *Grid*? Os métodos de fragmentação de dados, neste caso, não contemplarão estes tipos de dados no momento da distribuição dos fragmentos e, portanto, não será uma solução adequada.

Proj				
NoProj	NoProj	Orçamento	Local	TCD
P1	Informatização	150 000	Montreal	Doc
P2	Projet GRID	135 000	New York	SQL
P3	CAD/CAM	250 000	New York	CAD
P4	Maintenção	310 000	Paris	Doc

FIGURA 8. RELAÇÃO PROJ COM UM TIPO COMPLEXO DE DADOS.

Uma solução possível seria a replicação da linha inteira para todos os nós. Isso resolveria o problema de distribuição eqüitativa dos dados e da utilização de espaço de armazenamento entre os nós do *Grid*. Contudo, não resolveria o problema de fornecimento do dado (**CAD**), quando este fosse requisitado. Neste caso, o nó que estivesse incumbido pelo fornecimento desse documento ficaria com toda a carga, não podendo compartilhá-la com os demais nós.

3.2.1. PROPOSTA I - SEGMENTAÇÃO DO ATRIBUTO COM REPLICAÇÃO DO RESTANTE DA LINHA

Para esta proposta considere a replicação de todos os dados contidos no fragmento **P3** para todos os nós participantes do *Grid*, exceto o conteúdo do campo **TCD**, que contém o documento **CAD**; codifica-se o documento CAD pela técnica LDPC [PLANK04] para então segmenta-lo de acordo com o número de fragmentos de **P3** existentes (Figura 9).

Analisando esta proposta, nota-se que o problema apresentado anteriormente foi sanado, pois a carga foi distribuída entre os participantes. Contudo, esta solução originou um outro problema, o da consistência das múltiplas cópias de parte de uma linha. No caso do fragmento **P3**, conforme mostrado na Figura 9, a operação de reconstrução da relação original não será tão simples como mostrada no item 3.2, ou seja, a união das partes dará muito mais que o todo. Para este caso, a consulta será mais complicada que numa fragmentação horizontal simples.

Proj1

NoProj	NmProj	Orçamento	Local	TAD
P1	Instrumentação	150.000	Montreal	.Doc
P3	CAD/CAM	250.000	Nova York	.CAD1

Proj2

NoProj	NmProj	Orçamento	Local	TAD
P2	Projeto de BD	135.000	Nova York	.SQL
P3	CAD/CAM	250.000	Nova York	.CAD2

Proj2

NoProj	NmProj	Orçamento	Local	TAD
P2	Projeto de BD	135.000	Nova York	.SQL
P3	CAD/CAM	250.000	Nova York	.CAD2

Proj 4

NoProj	NmProj	Orçamento	Local	TAD
P3	CAD/CAM	250.000	Nova York	.CAD4
P4	Manutenção	310.000	Paris	.Doc

FIGURA 9. PROPOSTA DE REPLICAÇÃO DA LINHA E SEGMENTAÇÃO DO DOCUMENTO CAD.

3.2.2. Proposta II – Segmentação do Atributo sem Replicação da Linha e distribuição utilizando o IBP

Nesta proposta[2], considere um ambiente conforme mostra a Figura 10, onde se tem um *Grid* contendo um SGBD trabalhando em conjunto com o IBP (*Internet Backplane Protocol*) [PLANK04]. Neste ambiente, inicialmente identificamos o atributo que necessita ser segmentado, direcionamos tal atributo para um cliente IBP onde ele será codificado, segmentado e distribuído entre os nós disponíveis do IBP, não necessariamente os mesmos nós do *Grid*. No lugar onde ficaria o atributo segmentado, é armazenado um documento XML contendo os endereços de todos os segmentos gerados.

O IBP possui seu próprio sistema de codificação, que pode utilizar os códigos LDPC [PLANK04], segmentação e distribuição bem como o processo inverso partindo do documento XML gerado durante o processo de segmentação. Nesta proposta o SGBD não armazena nenhum dos segmentos, somente os metadados destes e, portanto, a responsabilidade pela consistência, armazenamento e acesso a estes segmentos ficaria aos cuidados do IBP. Apesar da vantagem da segmentação e armazenamento de um determinado documento em centenas de nós, facilitando seu *download* o SGBD seria subutilizado. Outro detalhe é a restrição do sistema de segmentação que só permite criar segmentos entre 500 KB e 1 MB obrigando, dependendo do tamanho do documento e do número de servidores, que sejam armazenados mais de um segmento por servidor. Essa obrigatoriedade desconsidera o contínuo crescimento das velocidades das redes de comunicação, impedindo o ajuste do tamanho máximo para um documento ser armazenado, sem a necessidade de segmentá-lo, evitando com isso a obtenção de um melhor desempenho no sistema como um todo.

[2] **Proposta inicial para este trabalho.**

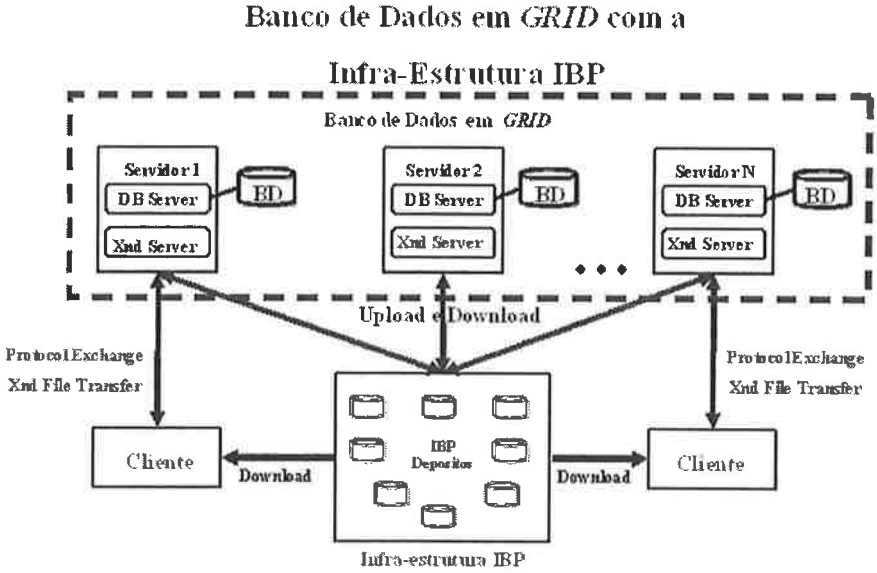

FIGURA 10. PROPOSTA DE BANCO DE DADOS EM *GRID* COM A INFRA-ESTRUTURA IBP [LOURENSO05].

3.2.3. PROPOSTA III – SEGMENTAÇÃO DO ATRIBUTO SEM REPLICAÇÃO DA LINHA UTILIZANDO APENAS O SGBDD DISPONÍVEL

Esta foi a proposta escolhida para este trabalho. Ela é semelhante à proposta anterior, contudo ela não utiliza o IBP como estrutura de armazenamento auxiliar. Somente o próprio SGBD é utilizado para armazenar os segmentos gerados do documento identificado como sendo de tamanho elevado e, portanto, incompatível com o processo de fragmentação de dados adotado.

Neste caso, após a identificação do documento, é realizada a sua codificação pelo código LDPC; em seguida segmenta-se em pedaços conforme o número de instâncias do Banco de Dados do *Grid* disponíveis no momento da segmentação.

Notemos que esquemas semelhantes ao proposto [PLANK04] [HOLUB04], não se baseiam nem SGBD, mas em sistemas de arquivos distribuídos, e nem na segmentação proporcional de acordo com o número de servidores ativos no momento da segmentação. Alguns outros trabalhos [BAIÃO98] [MATTOSO98] [BAIÃO00]

empregam esquemas de fragmentação para Banco de Dados Distribuídos, não para Banco de Dados em *Grid,* e nem contemplam a fragmentação vertical no nível de granularidade proposto neste trabalho, ou seja, em nível de atributo.

3.3. CONCLUSÃO

Neste capítulo foram apresentadas algumas propostas estudadas, inclusive uma proposta prévia deste trabalho que utilizava a estrutura IBP em conjunto com um SGBD, e aquela utilizada na implementação deste trabalho. No capítulo 4 detalharemos nossa proposta, bem como o cenário utilizado e algumas das soluções encontradas para resolver os problemas em relação ao ambiente em *Grid,* distribuição e controle dos atributos segmentados. Também serão apresentados os testes e resultados obtidos.

Capítulo 4. Implementação, Testes e Resultados.

"A ciência se compõe de erros que, por sua vez, são os passos até a verdade".

Julio Verne

Este capítulo apresenta a descrição da infra-estrutura, desenvolvida e implementada no âmbito desta tese para demonstrar as técnicas de segmentação, distribuição e controle dos atributos segmentados em um SGBDOR baseado em *Grid*, bem como os testes e resultados obtidos.

A arquitetura de implementação utilizada como base é mostrada na Figura 11. O projeto consiste em um ambiente de Banco de Dados Objeto Relacional baseado em *Grid* com um Escalonador/Controlador que controla os servidores, usuários e os dados inseridos neste *Grid.*

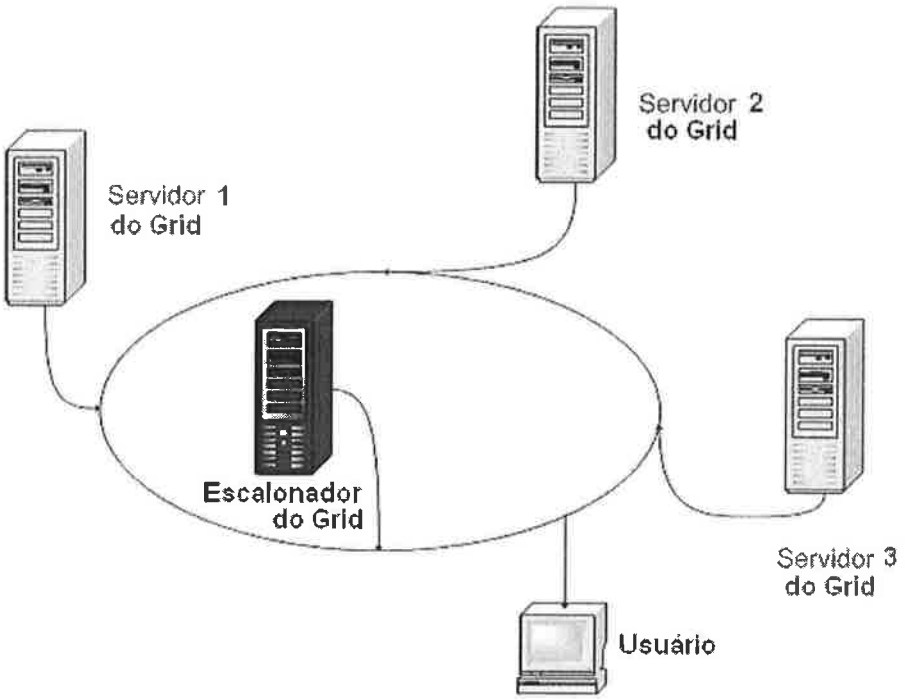

FIGURA 11. ARQUITETURA DO BANCO DE DADOS EM *GRID*.

Nossa intenção foi implementar um protótipo de uma infra-estrutura, baseado no modelo de gerenciamento definidos para *Grids* [CIRNE02], utilizando uma rede ***peer-to-peer*** de recursos possuídos por uma comunidade. Cada *peer* dá acesso aos recursos possuídos por um dono disposto a disponibilizá-lo, segundo seu conjunto de políticas visando, no nosso caso, uma aplicação específica utilizando Banco de Dados. Neste contexto, um dono é um usuário que tem o poder de disponibilizar e configurar o *peer* via um SGBD.

Uma vez disponibilizado, um *peer* **P** será acessado por usuários nativos e usuários estrangeiros. Os usuários nativos são os que utilizam o sistema através de **P**. Um usuário utiliza o sistema através dos serviços providos por um *peer*, que mantém comunicação com outros *peers* e se utiliza dos serviços da comunidade. Usuários estrangeiros utilizam **P** através da comunidade. Para **P** um outro *peer* **P2** utilizado por um usuário estrangeiro para o acesso aos seus recursos é um consumidor. Através de **P** os usuários locais acessam os **P2** e vêem-no como um servidor.

Antes do dono disponibilizar seu *peer* ou mesmo tentar se beneficiar do sistema de Banco de Dados em *Grid*, este terá que se cadastrar no sistema fornecendo um usuário válido do seu sistema com o mesmo nível de acesso dos já cadastrados em confiança. Após o cadastramento, o novo usuário estará apto a utilizar os recursos do Banco de Dados em *Grid*.

Após o cadastramento, todos os usuários de um determinado dono, que tenham permissão deste para acessar a aplicação em *Grid*, poderão acessá-la, sem a necessidade de cadastramento individual no escalonador. Isso foi possível pela **concessão de permissão**, ou seja, o dono cederá localmente aos seus usuários o seu usuário em confiança para que todos tenham acesso ao ambiente em *Grid*. Quando uma interface ou aplicação de um usuário tiver a necessidade de acessar outro domínio administrativo, o escalonador, conforme a política de utilização pré-estabelecida fornecerá em confiança um usuário válido para cada outro domínio administrativo. Este processo é totalmente transparente para qualquer usuário do ambiente.

A distribuição dos dados no *Grid* foi realizada por dois métodos basicamente:
(i) O primeiro método considerou-se os dados a serem inseridos com um único documento, neste caso, procurou-se realizar uma distribuição eqüitativa entre os servidores ativos do Grid pelo uso do método *Round-Robin*;
(ii) O segundo método considerou-se os dados a serem inseridos com um documento segmentado em vários pedaços, conforme a disponibilidade de servidores, neste caso, utilizou-se uma mistura de técnicas onde a parte

referente aos dados comuns foi inserida de maneira semelhante ao item anterior e a parte relativa aos segmentos gerados distribuídos entre os servidores ativos.

Dentro do problema proposto foi possível realizar os seguintes testes:

- **Validação:** onde foi confrontada a nossa solução com as funções definidas para uma infra-estrutura do *Grid*. Os testes deste tipo foram realizados em dois ambientes:
  - Em uma única máquina, com duas instancias de BD diferentes,
  - E no ambiente simulado do *Grid* no *cluster*.
- **Avaliação:** para contrastar se os resultados obtidos pelo sistema, estavam dentro do esperado. Os testes deste tipo só foram realizados no ambiente simulado do *Grid* no *cluster*, onde foi possível realizar tomadas de tempo para os casos estudados, procurando-se levar em conta os seguintes fatores:
  - Desempenho do sistema em cada um dos casos, com e sem segmentação, com e sem codificação;
  - Distribuição da carga de processamento de entre os servidores;
  - Escalabilidade.

Os testes realizados e os resultados obtidos serão descritos no item 4.7.

Para o estudo do comportamento da infra-estrutura implementada, durante a realização dos testes, a aplicação utilizada foi um servidor genérico de dados baseados em servidores WEB.

A implementação do protótipo da infra-estrutura foi dividida nas seguintes etapas:

- Caracterização do cenário e seus parâmetros;
- Modelagem e refinamento lógico do protótipo;
- Instalação e configuração do Sistema de Gerência de Banco de Dados;

- Definição, desenvolvimento e implementação dos elementos do protótipo.

A seguir, cada passo realizado na implementação do sistema proposto e seu funcionamento serão descritos com mais detalhes.

4.1. CENÁRIO

Para este trabalho assumiremos o seguinte cenário:

1. Os equipamentos base utilizados foram os do sistema de Cluster Itautec integrante do sistema PAD (Processamento de Alto Desempenho), que são da família de sistemas de processamento paralelo baseada em servidores (*commodity servers*) com interconexão ponto-a-ponto de alta velocidade (*Myrinet* ou Gigabit Ethernet).

2. Os nós utilizados são compostos por Dual Pentium 4 XEON 2.2 GHz, com um total de 18 GB de HD SCSI, 512KB de *cache* e 1GB de RAM, tanto para os servidores (3) como para o nó de controle (1).

3. Também usamos como nós cliente (2), *Dual Pentium* 3 de 1 GHz, com um total de 18 GB de HD SCSI, 512KB de *cache* e 256MB de RAM.

4. A rede de interconexão é uma Gigabit Ethernet.

5. O sistema operacional utilizado foi o *WINDOWS* 2003 *ENTERPRISE EDITION*, compatível com a versão do SGBD adotada.

6. Os SGBD adotados foram o SQL/SERVER 2005 e o SQL/SERVER 2000, contudo, poderia ter sido empregado qualquer SGBD que fornecesse suporte para armazenamento de objetos. Apesar dos SGBD permitirem

ambiente distribuído com gerenciamento integrado, os mesmos foram implementados isolados uns dos outros como ocorreria em um sistema do *Grid* real, onde cada SGBD teria um dono diferente.

Para o desenvolvimento, foi utilizada a ferramenta *Visual Studio* 2003 com VC# e VC++ sobre a infra-estrutura *ADO.Net* (***Access Data Object***). Esta infra-estrutura fornece um padrão para ambientes baseados na Web com facilidades de conexão a qualquer tipo de SGBD existente, pela simples escolha da opção por conexões otimizadas SQL, já disponíveis para o próprio SQL/SERVER, DB2 e ORACLE, ou via *OLE.DB* para outros tipos de BD.

4.2. Modelo Lógico das Tabelas do Protótipo do Sistema

Conforme o diagrama do modelo relacional da Figura 12 foi definido para o protótipo as seguintes tabelas:

- Tabela de servidores do *Grid* "**gservers_tb**" – possui a identificação de cada servidor existente no *Grid*, com uma descrição do seu sistema operacional, uma *string* de conexão para o SGBD com um usuário e senha válidos, o tipo (SQL ou OLE), o status (Ativo ou Desativado) do SGBD existente e o status (Ativo ou Desativado) do servidor;
- Tabela de metadados da aplicação "**metadoc_tb**" – guarda os identificadores do documento incluído no Banco de Dados do *Grid* bem como o identificador do servidor aonde ele foi armazenado;
- Tabela de situação "**numeradoc_tb**" – possibilita gerar uma numeração unificada para os documentos incluídos no sistema bem como fornece o tamanho máximo permitido para que um documento possa ser armazenado sem ser segmentado, possui também a informação de quantos servidores estão cadastrados e qual será o próximo servidor a ser utilizado quando um novo documento for incluído;

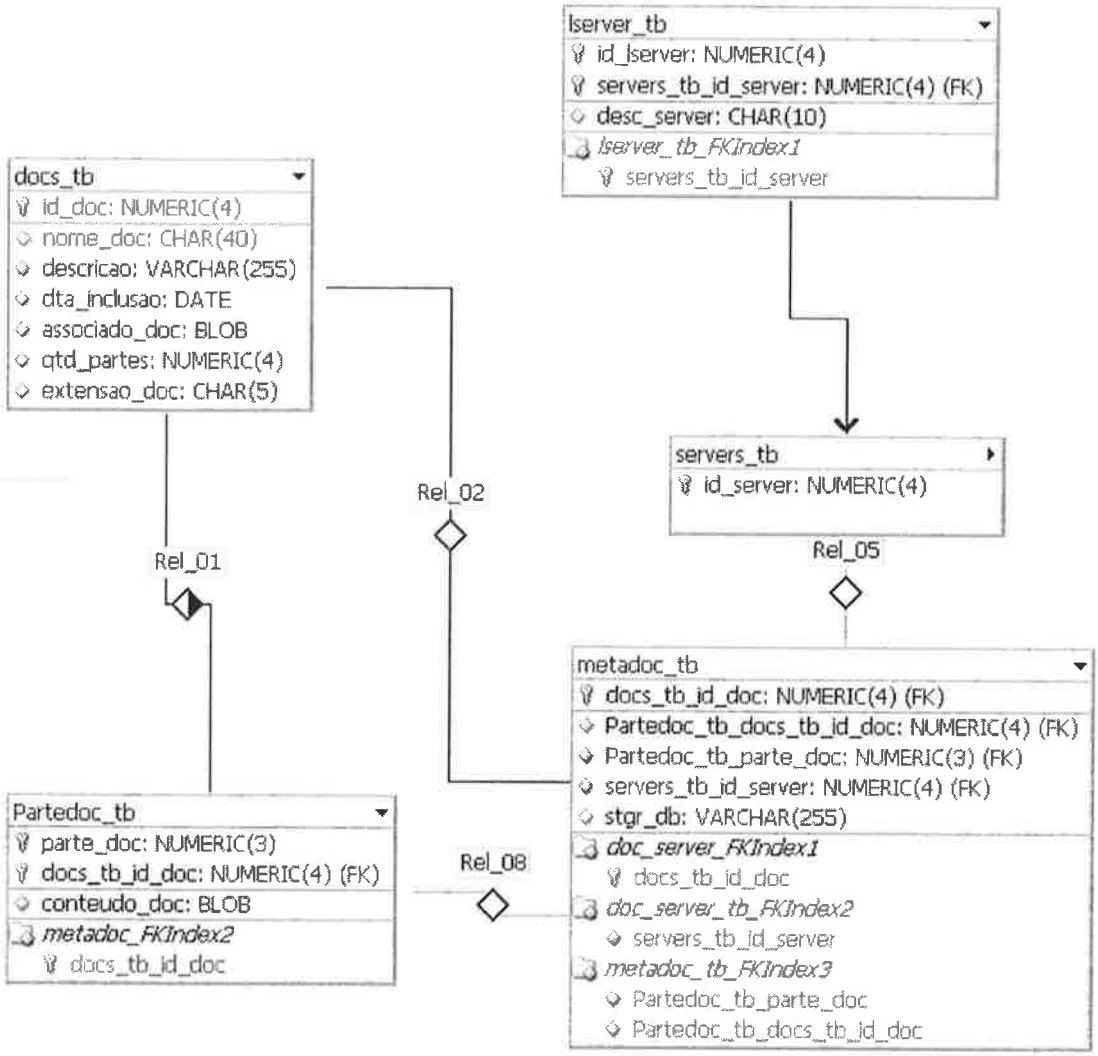

FIGURA 12. MODELO CONCEITUAL/LÓGICO DAS TABELAS DO SISTEMA.

- Tabela de documentos "**docs_tb**" – esta tabela é distribuída para todos os servidores e simula uma possível aplicação do sistema; ela contém a parte principal de todos os documentos inseridos no sistema podendo também armazenar os anexos conforme o problema proposto;
- Tabela de partes do documento "**patedoc_tb**" – ela contém o identificador, o nome e o conteúdo de cada segmento gerado pelo sistema e é distribuída para todos os servidores;

- Tabela do servidor local "**lservers_tb**" – esta tabela funciona mais como um registro no Banco de Dados local em que ele está cadastrada no sistema. Ela mantém o número fornecido pelo escalonador ao servidor bem como o usuário e senha fornecido ao escalonador e montado em uma *string* de conexão para o escalonador, (esta tabela também pode ser mantida em um arquivo independente do BD).

4.3. Requisitos do Protótipo do Sistema

Como requisito do protótipo do sistema, não foi considerado nenhum equipamento específico que pudesse impor um obstáculo significativo na portabilidade do sistema. Apenas computadores pessoais ou servidores, que permitam a utilização das aplicações às quais este trabalho é direcionado, são considerados. Assim, os resultados obtidos poderão ser facilmente reproduzidos em laboratórios ou empresas.

A comunicação entre os processos foi facilitada pela utilização da infra-estrutura do *ADO.Net* que fornece alguns objetos para a Conexão, Recuperação, Inserção e Atualização de dados em um Banco de Dados.

Apesar da implementação ter sido feita em uma arquitetura típica Cliente/Servidor, a proposta poderá ser facilmente estendida para a plataforma baseada em servidores Web, típica de sistema baseados em *Grid* [FOSTER01] [CIRNE02].

4.4. Arquitetura de Programas

Como já mencionamos no item 4.1 a infra-estrutura. Net forneceu os recursos básicos para o desenvolvimento dos módulos básicos do sistema - **monitoramento, entrada** e **recuperação.** Estas aplicações são responsáveis pela situação dos

servidores, escolha, segmentação, distribuição, recuperação e unificação dos dados. A Figura 13 apresenta o diagrama de Classes do protótipo do sistema.

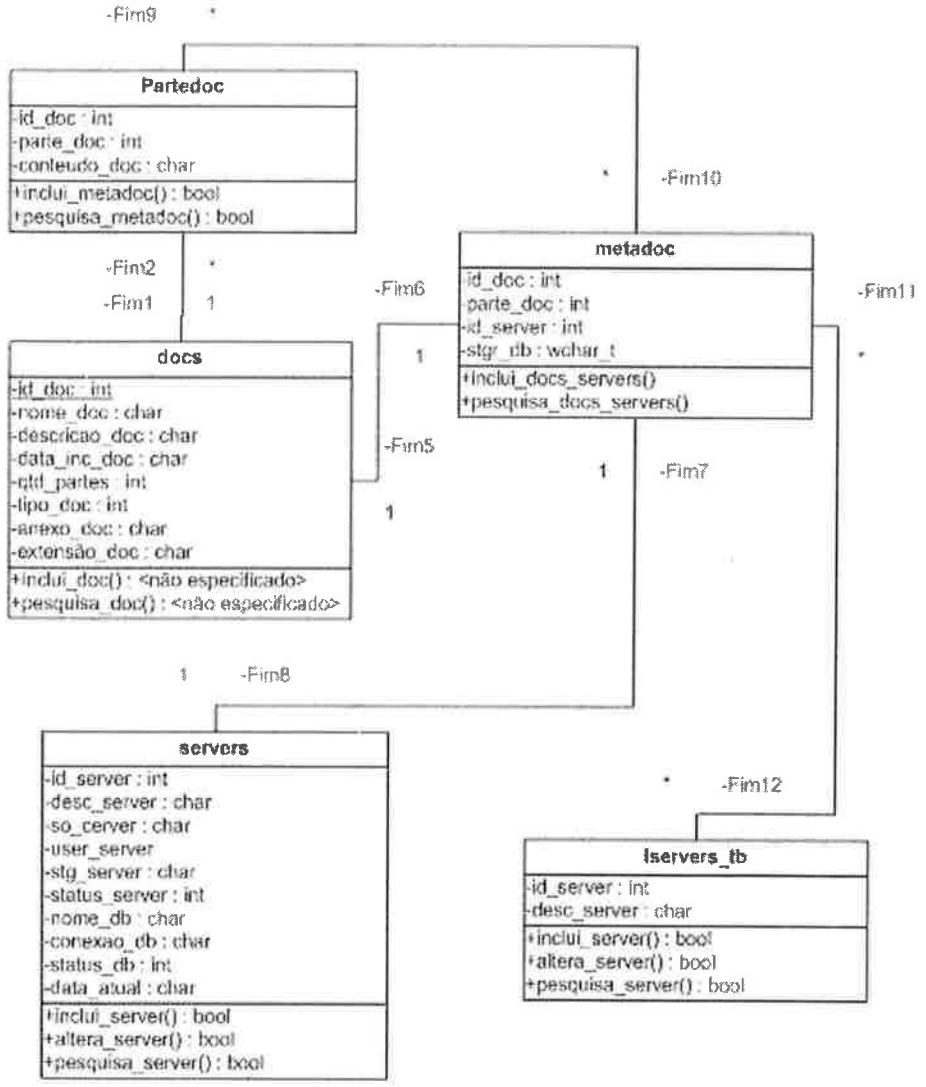

FIGURA 13. MODELO DE CLASSES DOS MÓDULOS DO PROTÓTIPO DO SISTEMA.

4.4.1. Agentes de Monitoramento

Os agentes de monitoramento são instalados em cada um dos servidores membros do *Grid*. Cada agente mantém um registro local com a numeração do servidor em que está instalado, que é fornecida no momento do seu cadastramento no

Grid. Sua função é de tempos em tempos verificar a situação do Banco de Dados local (ativo, desativo), atualizando em seguida esta situação no escalonador do *Grid.*

4.4.2. Módulo de Entrada

Este módulo possui várias funções: validar o usuário junto ao ambiente; receber os parâmetros do ambiente; verificar o tamanho do anexo junto ao limite; disparar os processos de codificação, segmentação e inclusão dos segmentos do documento que tiver tamanho além do limite especificado nos servidores. Este processo ainda possui uma opção que possibilita a inclusão do anexo sem a sua codificação e segmentação.

A primeira função, validação do usuário, é inerente a todos os demais módulos que possuem interface com o usuário e compreende principalmente a conexão ao escalonador.

O recebimento dos parâmetros referentes à numeração do novo documento que estará sendo incluído, tamanho limite para a inclusão de anexos (aplicativos, vídeo, etc.), identificador do servidor que receberá o corpo principal do documento é fornecido pelo escalonador do *Grid.* A numeração do documento permitirá manter uma numeração unificada dos documentos incluídos no sistema, visto que esta atividade é determinante quando necessitamos ter uma tabela de metadados unificada de todos os documentos incluídos. O tamanho limite foi utilizado para determinar se um documento terá que ser segmentado. Desta forma, se o documento estiver acima do tamanho limite, ele influenciaria consideravelmente no desempenho do sistema e, portanto, é necessário que o mesmo seja segmentado. O identificador do servidor, que também é fornecido pelo escalonador, permitiu uma distribuição eqüitativa dos documentos incluídos no sistema, promovendo com isso o balanceamento de carga durante a sua inclusão e transparência do ambiente.

O módulo de codificação "***encode***", que é chamado pelo processo de entrada, codifica os documentos selecionados propiciando uma maior segurança, quando o processo de segmentação e unificação for acionado, pois permite a verificação da integridade dos arquivos segmentados.

A função de segmentação chama outro processo, o ***split*** [CYGWIN05] que permite segmentar um arquivo em múltiplos do seu tamanho total. Portanto, antes de chamar tal processo foi necessário primeiro obter o tamanho do anexo (isso já tinha sido feito durante a verificação do anexo junto ao limite). O passo seguinte foi a obtenção do número de servidores ativos naquele instante. Isso é obtido por meio de uma consulta a tabela de servidores do *Grid* no escalonador, após isso, foi utilizada a seguinte fórmula: **TamSeg = (TamDoc / QdtServ) / 1024**; onde **TamSeg** é o tamanho do segmento, **TamDoc** é o tamanho do documento e **QtdServ** é a quantidade de servidores disponíveis, que é diferente da quantidade total de servidores associados ao *Grid*, o valor **1024 K**, (deve-se a estarmos trabalhando no ambiente *Windows)* - só então o processo ***split*** pode ser disparado.

O processo de inclusão dos segmentos é disparado e recebe como parâmetro o endereço de cada um dos segmentos individualmente e o endereço de um dos nós servidores disponíveis. A inclusão do segmento é realizada na tabela (**partesdoc_tb**) diferente da tabela (**docs_tb**) onde o corpo do documento foi incluído (Tabela 4 e Tabela 6 respectivamente). Tanto a tabela docs_tb como a partesdoc_tb estão distribuídas pelos servidores que compõe o *Grid.*

A última função a ser executada é a de inclusão dos metadados, que também existe no processo de inclusão de cada segmento; no caso do módulo de entrada ele inclui uma entrada para o corpo principal do documento. A Figura 14 apresenta a janela do módulo de entrada.

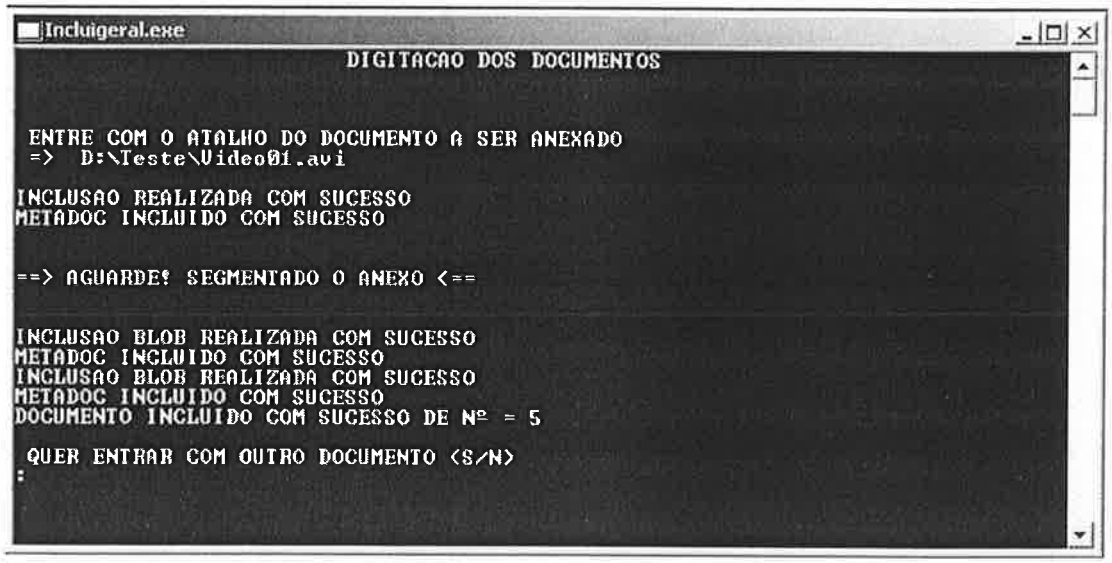

FIGURA 14. JANELA DO MÓDULO DE ENTRADA.

Passos a serem seguidos na inclusão de um determinado documento:

1. É requisitada a entrada do usuário e senha;
2. É requisitada a digitação das informações gerais, nome do documento, descrição, etc.;
3. É requisitada a informação se o usuário terá algum documento, arquivo, etc., a ser anexado junto com as outras informações;
   - 3.1. Caso afirmativo será requisitado do usuário, nome e extensão e caminho onde se encontra o documento a ser anexado, e então as informações serão incluídas;
   - 3.2. Caso negativo os dados serão inseridos sem o anexo;
4. No final é fornecido o número (chave) do documento incluído.

4.4.3. Modulo de Recuperação

Similarmente ao módulo de entrada, este módulo possui várias funções: validação do usuário junto ao ambiente; descoberta e recuperação do documento principal e de seus anexos, caso tiver; descoberta e recuperação dos anexos em paralelo; unificação dos segmentos e, por último, a decodificação do documento.

Este módulo, ainda possui uma opção que possibilita a não recuperação do documento que esteja anexado.

A função de validação do usuário é idêntica àquela do módulo de entrada e compreende, principalmente, a validação da conexão junto ao escalonador do *Grid*.

A função de descoberta faz uma consulta ao escalonador do *Grid* por intermédio do identificador do documento. O processo de descoberta também informa se o possuidor do documento em questão está ativo.

Após ter obtido a localização do documento principal no *Grid* é executada a função de pesquisa para obter o documento solicitado. Esta função, além de retornar o documento principal, retornará seu anexo no caso do mesmo não ter sido segmentado. Caso contrário será executada outra função para descoberta dos segmentos do documento anexado.

A função de descoberta realiza um novo acesso ao escalonador, a fim de obter o endereço de cada segmento e a situação do servidor que a possui. Para cada segmento é então executado um processo de recuperação, que retornará o identificador e o endereço do segmento desejado, bem como o caminho onde este deverá ser colocado. Na impossibilidade da recuperação de um dos segmentos o processo todo é interrompido.

O processo de recuperação de cada uma dos segmentos é disparado por um processo dentro do módulo de recuperação, e simplesmente recupera o segmento a ele especificado; ele também gera um arquivo registrando quanto tempo gastou para a recuperação de cada um dos segmentos. Esta última parte foi necessária para registrar os tempos gastos em cada processo de recuperação individualmente.

A função de reunificação chama o processo "***cat***" [CYGWIN05] para a recomposição do documento original que foi anexado.

O processo de decodificação "***decode***" é o último a ser chamado pelo módulo de recuperação, que decodificara o documento selecionado fazendo uma verificação da integridade.

O módulo de recuperação gravará um arquivo com o tempo total do processo de recuperação do documento anexado. Tanto o arquivo gerado pelo módulo de recuperação como o gerado pelos processos de recuperação dos segmentos foram utilizados para armazenar os tempos do protótipo do sistema. A Figura 15 apresenta a janela do módulo de recuperação.

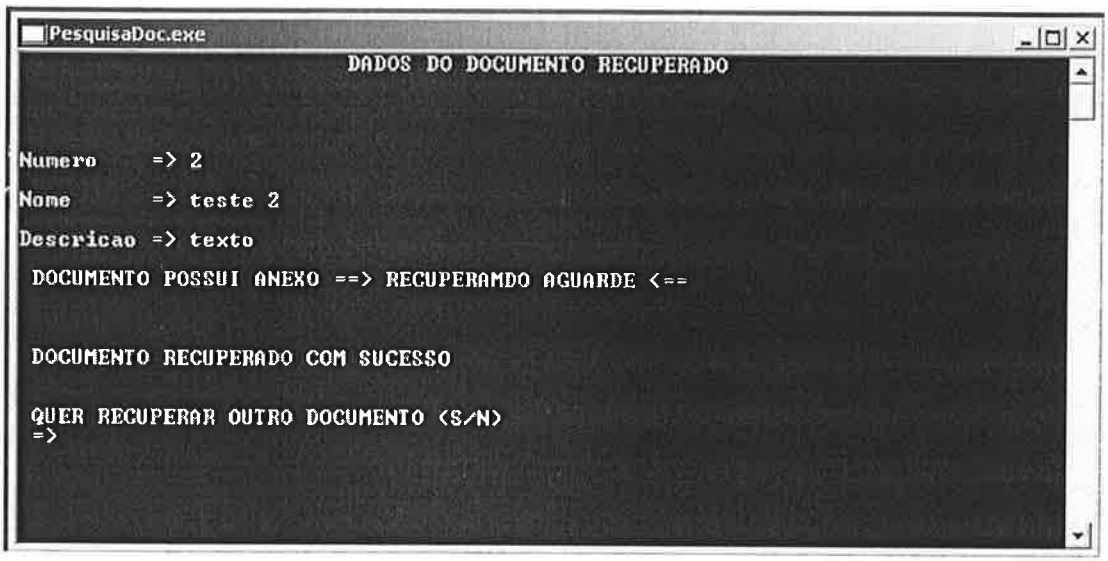

FIGURA 15. JANELA DO MÓDULO DE RECUPERAÇÃO.

A recuperação das informações armazenadas, conforme Figura 15 apresenta o número do documento solicitado, o nome e sua descrição. Se a informação armazenada possuir um documento anexado, este será recuperado e colocado em uma área temporária em disco. Como já mencionado, não é o intuito deste trabalho utilizar qualquer editor para o documento recuperado.

As funções **cat** e **split** pertencem ao ambiente do Cygwin, [CYGWIN05] enquanto que as funções **encode** e **decode** foram conseguidas dos aplicativos de teste do LDPC fornecidas por Plank [PLANK04].

4.5. Fluxo de Processamento

Para todas as opções abordadas, com ou sem segmentação e com e sem codificação, todas as Inclusões e as Requisições têm que passar, obrigatoriamente, pelo escalonador do *Grid* e, portanto, este assume todo o processamento inicial dentro do *Grid.*

Conforme mencionado anteriormente, com todos os nós servidores ativos, os agentes de monitoramento acompanham continuamente a situação de cada um dos servidores de Banco de Dados do *Grid*, atualizando em intervalos de tempo a tabela **gservers_tb** localizada no escalonador do *Grid.* Esta informação é importante tanto para o processo de entrada quanto para o processo de recuperação dos documentos, pois evita tentativas de conexão a servidores não disponíveis.

Ao tentar incluir um documento (Figura 16), o processo de inclusão verificará os parâmetros do sistema referentes à numeração e tamanho limite. No caso das informações a serem incluídas possuírem um documento a ser anexado, o processo confrontará o tamanho limite com o tamanho desse documento. No caso do tamanho do documento ser menor ou igual ao tamanho limite, o processo obterá o próximo servidor disponível e incluirá as informações junto com seu anexo em um mesmo servidor, gerando em seguida uma única entrada na tabela de metadados (**metadoc_tb**), relativa às informações incluídas. Caso contrário, o processo codificará e segmentará o documento considerando o número de servidores ativos no momento. Cada segmento, então, será enviado para um único servidor. Neste caso, será gerada uma entrada na tabela de metadados para cada segmento e mais uma entrada para as informações principais.

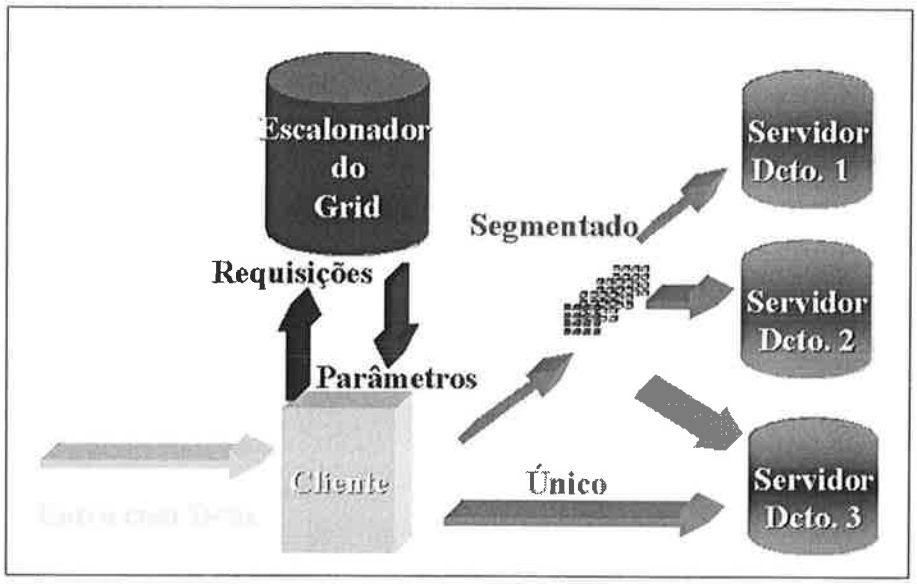

FIGURA 16. FUXO DE INCLUSÃO DE DADOS NO *GRID*.

O processo de recuperação (Figura 17) trabalha independentemente da maneira como o documento foi incluído (com ou sem anexo, segmentado ou não, codificado ou não). Quando uma determinada informação for requisitada, o processo de recuperação buscará no escalonador a sua localização. Inicialmente, o processo retornará somente os dados principais. Após a conclusão desta operação, o processo verificará se existe ou não anexo e se este está ou não segmentado. No caso do anexo não estar segmentado, ele estará no mesmo local da parte principal do documento e, portanto, a recuperação será direta, sem a necessidade de novo acesso ao escalonador. Antes do início da recuperação do anexo, é registrado o tempo inicial do processo e após a sua total recuperação é registrado o tempo final, este intervalo de tempo foi utilizado para análise comparativa da recuperação com os anexos segmentados. No caso do anexo ter sido segmentado, um novo acesso da tabela de metadados será necessário para recuperar os servidores onde cada segmento está localizado. Depois de recuperada esta informação, é registrado o tempo inicial e disparado o processo de recuperação dos segmentos de forma paralela. Quando todas as partes estiverem recuperadas, é executado o processo de unificação e, em seguida, o de decodificação. Por fim, o tempo final do processo é registrado.

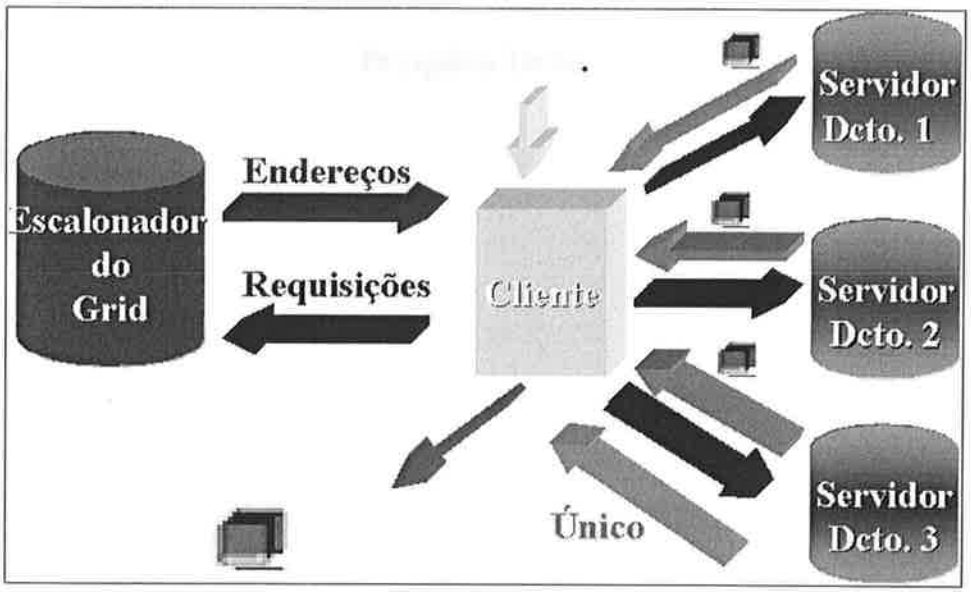

FIGURA 17. – FUXO DE RECUPERAÇÃO DE DADOS NO *GRID*.

4.6. CONFIGURAÇÃO DA DISTRIBUIÇÃO DOS DADOS

Como já mencionado no início deste capítulo, os dados foram distribuídos por dois métodos, independentes do usuário e visando uma distribuição eqüitativa dos mesmos. O primeiro método utilizado foi o *Round-Robin,* no qual além do servidor ser fornecido pelo escalonador sem interferência do usuário, o mesmo é feito de maneira rotativa entre os servidores disponíveis. O segundo método é relativo à distribuição dos segmentos de um documento anexado, no qual o documento é segmentado levando-se em conta todos os servidores disponíveis no momento da segmentação, o que evita o acúmulo de volume de dados em um único servidor. Neste caso, utilizamos uma mistura dos métodos onde a parte referente aos dados comuns foi inserida semelhante ao método anterior e à parte relativa aos segmentos gerados distribuídos entre os servidores ativos.

4.7. Testes e Resultados

Foram realizados dois tipos de testes, na seguinte ordem:

- **Validação**: - onde foi verificado o funcionamento correto da infra-estrutura desenvolvida para o gerenciamento dos dados, conforme as definições de *Grid* [CIRNE02] e *Data Grid* [FOSTER01]. Estes testes iniciais foram realizados em dois ambientes distintos: (i) Em uma única máquina, com duas instâncias de BD diferentes, (ii) e no ambiente simulado de *Grid* no *cluster*. Na seguinte seqüência:

  1. Definição do modelo a ser implementado e testado;
  2. Teste de inclusão dos documentos, enfocando a inclusão correta, codificação, segmentação e distribuição dos segmentos;
  3. Teste de recuperação dos documentos, enfocando na pesquisa, extração, recomposição e decodificação dos documentos utilizados;
  4. Especificação dos parâmetros, (limite máximo para segmentação dos arquivos anexados).

- **Avaliação:** - para contrastar se os resultados obtidos pelo sistema, estavam dentro do esperado. Os testes deste tipo só foram realizados no ambiente simulado de *Grid* no *cluster*, onde foi possível realizar tomadas de tempo para os casos estudados, procurando-se levar em conta os seguintes fatores:
  1. Desempenho do sistema em cada um dos casos, com e sem segmentação e, com e sem codificação. Com as tomadas de tempo para cada caso;
  2. Distribuição da carga de processamento de entre os servidores;
  3. Escalabilidade.

4.7.1. Testes de Validação

Como mencionado anteriormente, estes testes servirão para a verificação do funcionamento correto da infra-estrutura desenvolvida. No caso da aplicação que incluiu os documentos, ela foi desenvolvida visando uma distribuição eqüitativa entre os servidores. Assim, buscou-se uma distribuição onde cada servidor possua não só a mesma quantidade de documentos, mas também, que os documentos ocupem, aproximadamente, o mesmo espaço em disco de cada um dos servidores. Isso foi possível pela utilização da técnica *Round-Robin* em conjunto com a segmentação dos documentos anexados conforme a quantidade de servidores disponíveis.

Alguns dos documentos utilizados nesta fase dos testes são apresentados na Tabela 02. Os elementos desta tabela foram utilizados para a validação e testes dos módulos do protótipo e para a determinação do parâmetro (limite máximo). A especificação desse parâmetro foi feita através da medida dos tempos de acessos para cada servidor de dados e para a estação de controle (conforme mencionado no item 4.3). Estas medidas iniciais também serviram para verificar o comportamento de todos os processos necessários ao teste do protótipo.

Para os testes, utilizou-se uma estação dentro da mesma rede do *cluster*, onde foi possível executar os processos de Entrada e Recuperação dos documentos. Esse procedimento foi necessário, pois o protótipo trabalhava dentro de uma arquitetura Cliente/Servidor.

Visto que a determinação ou não da segmentação dos documentos anexados no protótipo é controlado pelo parâmetro "tamanho limite", este pode ser configurado de acordo com as necessidades e características do sistema em que se esteja operando.

Tabela-02. EXEMPLO DOS DOCUMENTOS UTILIZADOS.

Nome e Código do Documento	Tamanho (bytes)
ItunesSetup.exe	**21.525**
MP10Setup.exe	**12.397**
Video01.avi	**96.644**
Video02.avi	**107.944**
Pb1.mov	**4.170**
Definições.doc	**1.517**

As Tabelas 03 a 07 mostram a situação das tabelas dos bancos de dados do *Grid* após alguns documentos terem sido incluídos, o teste utilizou apenas dois nós como servidores mais o escalonador.

Tabela-03. CONTEÚDO DA TABELA METADOC_TB

Id_doc	parte_doc	Id_server
1	0	1
1	1	1
1	2	2
2	0	2
3	0	1
4	0	2
4	1	1
4	2	2
5	0	1
5	1	1
5	2	2

A Tabela 03 apresenta os dados de uma das tabelas de controle da infra-estrutura do sistema a tabela **metadoc_tb,** localizada no escalonador do *Grid.* Nesta tabela podemos identificar três campos o Id_doc, parte_doc e o Id_server. O **Id_doc** é o identificador do documento incluído, este campo é gerado pelo sistema no momento da inclusão do documento; o **parte_doc,** que também é gerado pelo sistema, indica a parte do documento armazenado, no caso do valor Zero (0), este

valor esta relacionado com o corpo principal do documento; o **Id_server** indica em qual servidor está localizado o documento.

Tabela-04. CONTEÚDO DA TABELA DOCS_TB (1)

id_doc	nome_doc	desc_doc	qtd_partes	tam_doc	nome_anexo	ext_anexo	conteudo_doc
1	teste de inclusao	video	2	71584341	Video03.wmv	wmv	0x
3	teste03	texto	1	27	Teste.txt	txt	
5	Teste de inclusao	video	2	98963456	Video01.avi	avi	0x

Tabela-05. CONTEÚDO DA TABELA DOCS_TB (2)

id_doc	nome_doc	desc_doc	qtd_partes	tam_doc	nome_anexo	ext_anexo	conteudo_doc
2	teste 2	texto	1	1553408	Definicoes.doc	doc	
4	teste 03	teste de inclusao	2	110534656	Video02.avi	avi	

As Tabelas 04 e 05 apresentam os dados das tabelas **Docs_tb**, que estão distribuídas em cada um dos servidores do *Grid*. Estas tabelas armazenam as informações do corpo principal do documento, simulando uma possível aplicação, nelas podem ser observados vários campos, alguns deles como **qtd_partes**, **tam_doc**, **nome_anexo**, **ext_anexo** e **conteúdo_anexo** se referem a um possível documento que poderá estar anexado. Estes campos ajudam no controle da aplicação e recuperação do documento anexado no *Grid*.

Tabela-06. CONTEÚDO DA TABELA PARTEDOC_TB (1)

id_doc	parte_doc	Nome_anexo	Conteúdo_doc
1	1	Video03.001	0x53797374656D2E427974655B5D
4	1	Video02.001	0x53797374656D2E427974655B5D
5	1	Video01.001	0x53797374656D2E427974655B5D

Tabela-07. CONTEÚDO DA TABELA PARTEDOC_TB (2)

id_doc	parte_doc	nome_anexo	Conteúdo_doc
1	2	Video03.002	0x53797374656D2E427974655B5D
4	2	Video02.002	0x53797374656D2E427974655B5D
5	2	Video01.002	0x53797374656D2E427974655B5D

As Tabelas 06 e 07 apresentam os dados armazenados na tabela **partesdoc_tb**, distribuídos pelos servidores do *Grid*. Nesta tabela só são armazenados os segmentos dos documentos que excederam ao limite especificado. Nelas podem ser observados os campos **id_doc**, **parte_doc**, **nome_anexo** e **conteúdo_doc, n**este último, é armazenado um ponteiro para o local onde realmente o segmento está armazenado. Os demais campos armazenam, respectivamente: o identificador que o anexo pertence, a parte do segmento e o nome desta parte. Todas estas informações auxiliam na recuperação de cada um dos segmentos e na reconstrução do documento original.

4.7.2. Testes de Avaliação

Como mencionados anteriormente, estes testes foram utilizados para contrastar se os resultados obtidos pelo sistema desenvolvido, estavam dentro do esperado.

Os testes foram realizados na seguinte ordem:

1. Inclusão dos documentos sem segmentação independente do tamanho;
2. Inclusão dos documentos com segmentação, também independente do tamanho;
3. Inclusão dos documentos codificados e segmentados, também independente do tamanho;
4. Recuperação de cada um dos documentos com as respectivas tomadas de tempos para cada documento que foi incluído:
   - 4.1. Tempo total de processamento;
   - 4.2. Tempo total de *download*.

Considerou-se como tempo total como o somatório dos tempos de *download* mais o de reunificação dos documentos segmentados mais o de decodificação dos documentos codificados.

Para o tempo-total-de-*download* dos documentos segmentados foi considerado como tempo-inicial o registro de tempo do processo mais jovem e, para o tempo-final, o registro de tempo do último processo.

Tabela-08. DOCUMENTOS UTILIZADOS NOS TESTE DE AVALIAÇÃO

N°	Nome	Tamanho Total (K-bytes)	Tipo
1	Teste01.wmv	23.303	U
2	Teste01.wmv	23.303	M
3	Teste02.ogm	47.859	U
4	Teste02.ogm	47.859	M
5	Teste03.ogm	69.907	U
6	Teste03.ogm	69.907	M
7	Teste00.wmv	11.652	U
8	Teste00.wmv	11.652	M
9	Teste08.wmv	5.826	U
10	Teste08.wmv	5.826	M
12	Teste09.wmv	2.913	U
13	Teste09.wmv	2.913	M
14	Teste04.ogm	90.000	U
15	Teste04.ogm	90.000	M
16	Teste05.ogm	122.240	U
17	Teste05.ogm	122.240	M
-	Teste06.ogm	180.000	NI
-	Teste07.ogm	244.467	NI
19	Teste00.wmv	11.652	MC
20	Teste01.wmv	23.303	MC
21	Teste02.ogm	47.859	MC
22	Teste03.ogm	69.907	MC
23	Teste04.ogm	90.000	MC
24	Teste05.ogm	122.240	MC
25	Teste08.wmv	5.826	MC
26	Teste09.wmv	2.913	MC

A Tabela 08 apresenta os documentos nesta fase de testes, onde o tipo mostra como o documento foi incluído no Banco de Dados em *Grid*. A seguir é apresentado o significado de cada um destes tipos:

U - Documento único;

M - Segmentados sem codificação;

MC - Segmentados com codificação;

NI - Documentos que não puderam ser incluídos.

Os resultados dos testes são apresentados em gráficos, onde são mostrados os tempos obtidos de processamento total e de *download* dos documentos anexados. Os gráficos gerados foram separados conforme o tipo de armazenamento do documento. Tabela 08.

A primeira seqüência de gráficos, Figuras 18 a 21, apresenta os resultados obtidos pelas tomadas de tempo (Tempo total de processamento e tempo de *download*) dos documentos armazenados de maneira Única X Segmentado. Os gráficos confrontam os tempos destas duas formas de armazenamento.

Na Figura 18, são confrontados os tempos totais de processamento para as operações de consulta. Pode-se notar, neste caso, que os tempos são sempre inferiores para os documentos que foram armazenados de maneira única.

Na Figura 19, são confrontados os tempos de *download*. Para este caso, pode-se notar que a diferença de tempo entre os tipos de armazenamento apresenta uma tendência de ganho para os documentos segmentados cujos tamanhos estejam além de 10 MB.

Na Figura 20 são apresentadas as diferenças calculadas entre os tempos totais. Neste caso, pela análise feita dos resultados obtidos, nota-se uma tendência para redução no ganho de tempo para recuperação dos documentos armazenados inteiros contra os documentos segmentados. Isso é devido ao aumento no ganho de tempo de *download* em favor dos documentos segmentados, como pode ser visto na Figura 21.

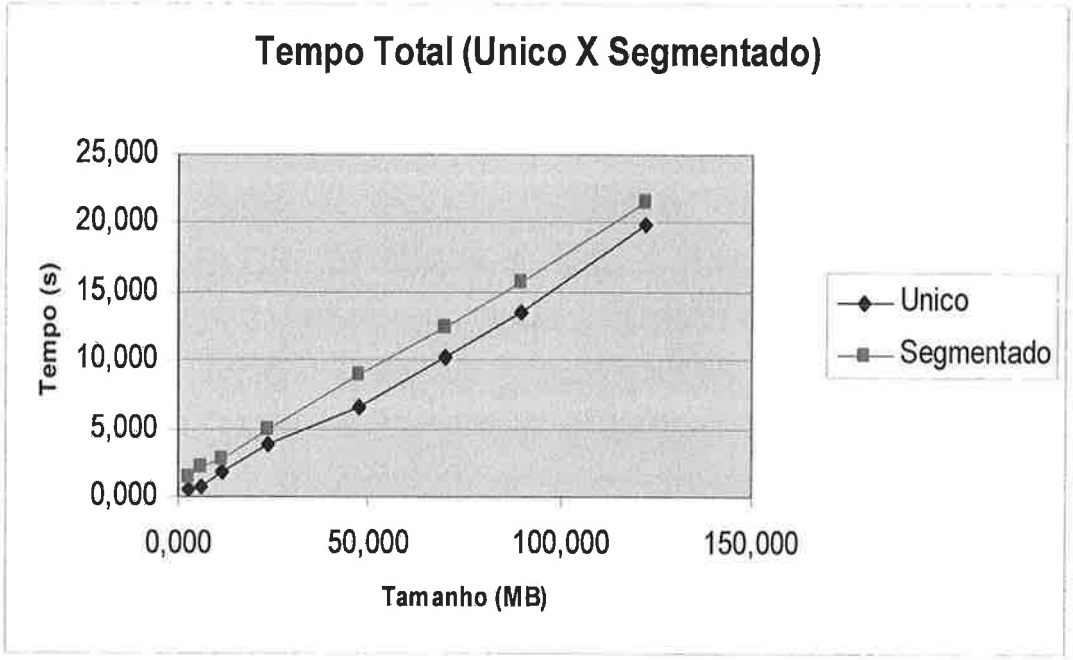

FIGURA 18. TEMPO TOTAL DE PROCESSAMENTO.

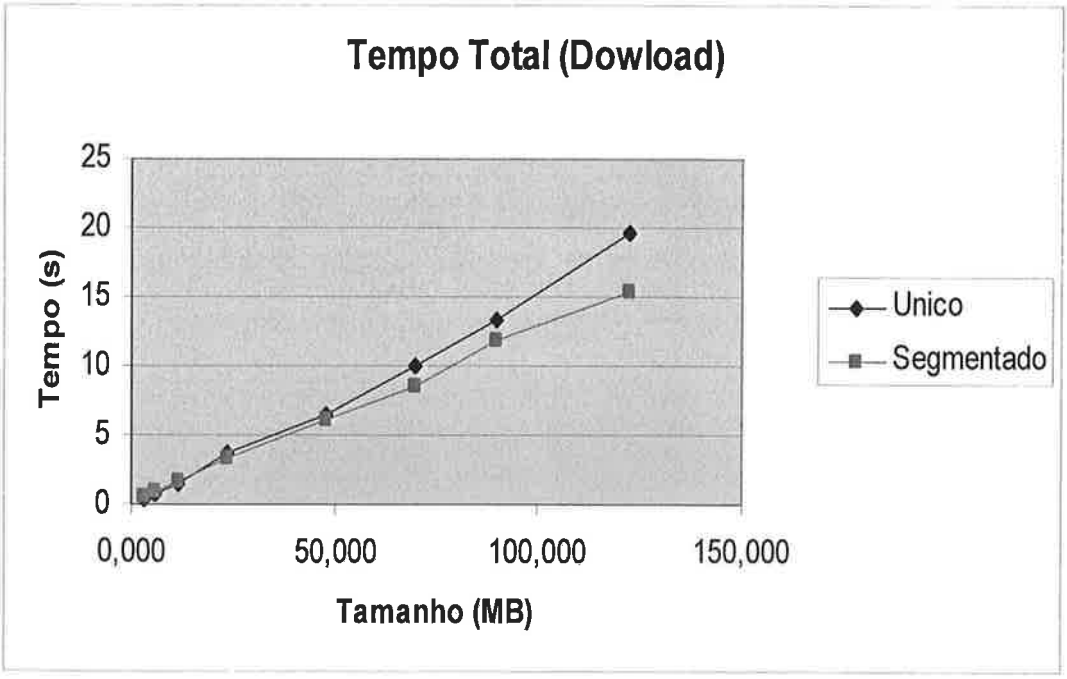

FIGURA 19. TEMPO TOTAL DE *DOWNLOAD*.

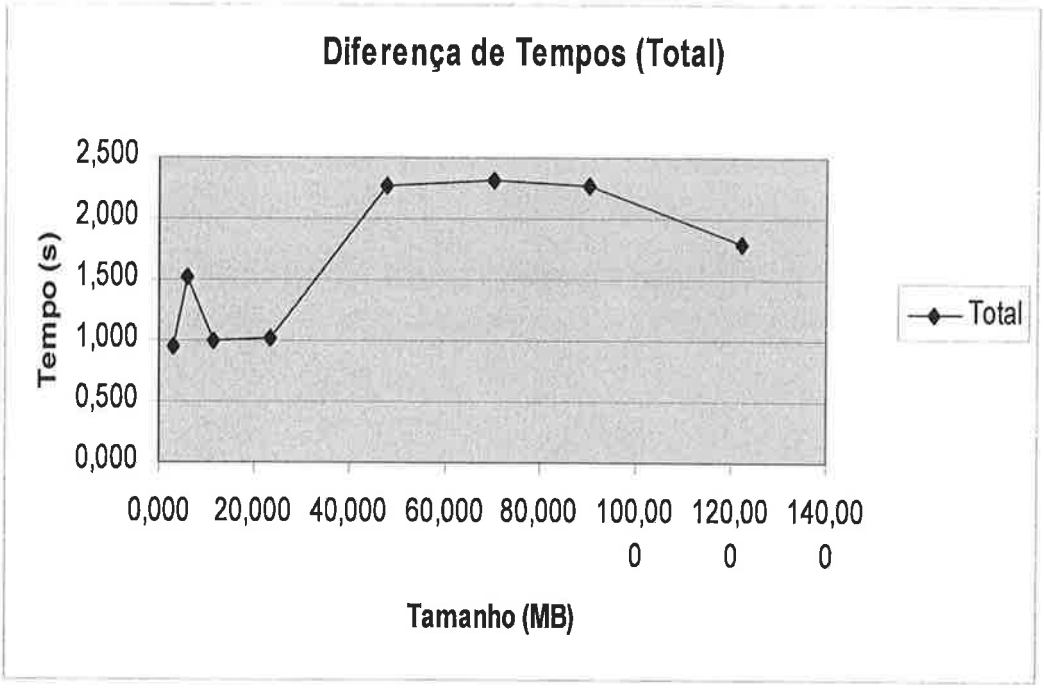

Figura 20. DIFERENÇA ENTRE OS TEMPOS TOTAIS.

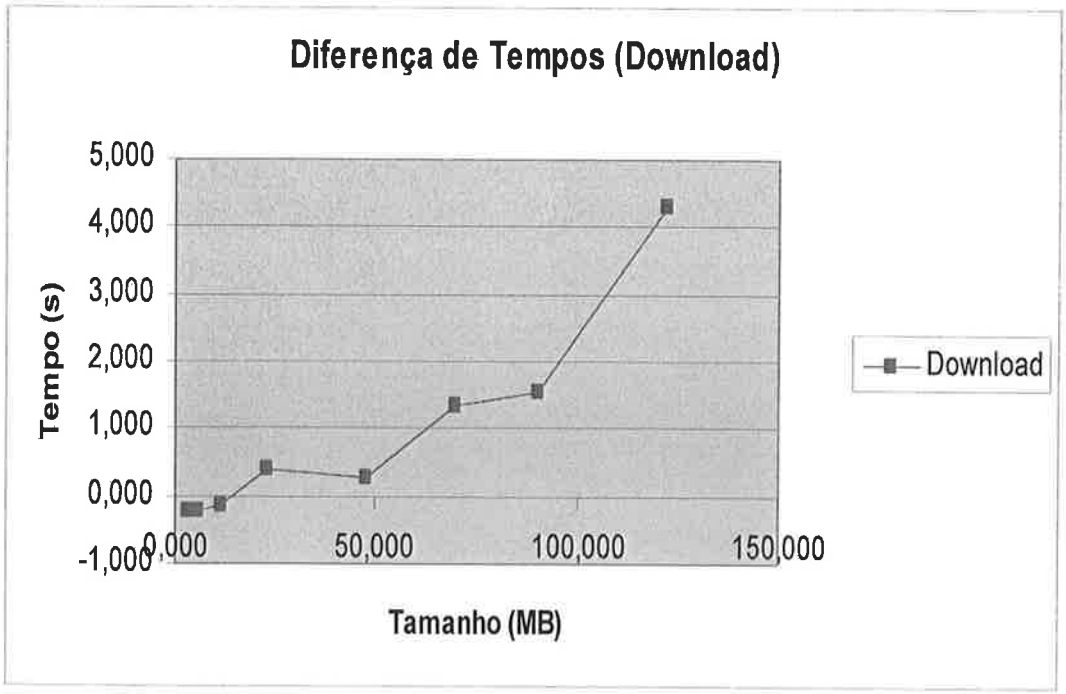

Figura 21. DIFERENÇA ENTRE OS TEMPOS DE *DOWNLOAD*.

A próxima seqüência de gráficos (Figuras 22 a 25) apresenta os resultados obtidos pelas tomadas de tempo (Tempo total de processamento e tempo de *download*) dos documentos armazenados de maneira Única X Codificado + Segmentado. Os gráficos confrontam os tempos destas duas formas de armazenamento.

Na Figura 22, são confrontados os tempos total de processamento para as operações de consulta. Nota-se, neste caso, que os tempos são sempre inferiores para os documentos que foram armazenados de maneira única, similarmente às tomadas de tempo da seqüência anterior. Contudo, nota-se uma diferença maior em favor dos documentos únicos. Isso é devido à sobrecarga local para decodificação dos documentos.

Na Figura 23, são confrontados os tempos de *download*. Para este caso, pode-se notar similarmente às tomadas de tempo da seqüência anterior, que a diferença entre os dois tipos de armazenamento apresenta também uma tendência de ganho para os documentos codificados.

Na Figura 24, são apresentadas as diferenças calculadas entre os tempos totais. Neste caso, pela análise feita dos resultados obtidos, não fica evidente a mesma tendência da seqüência anterior, apesar de uma variação ter ocorrido, o que ocasionou uma perda na linearidade. Nota-se uma redução no ganho de tempo para recuperação dos documentos armazenados inteiros, contra os documentos codificados. Isso é devido também ao aumento no ganho de tempo de download em favor dos documentos segmentados como pode ser visto na Figura 25.

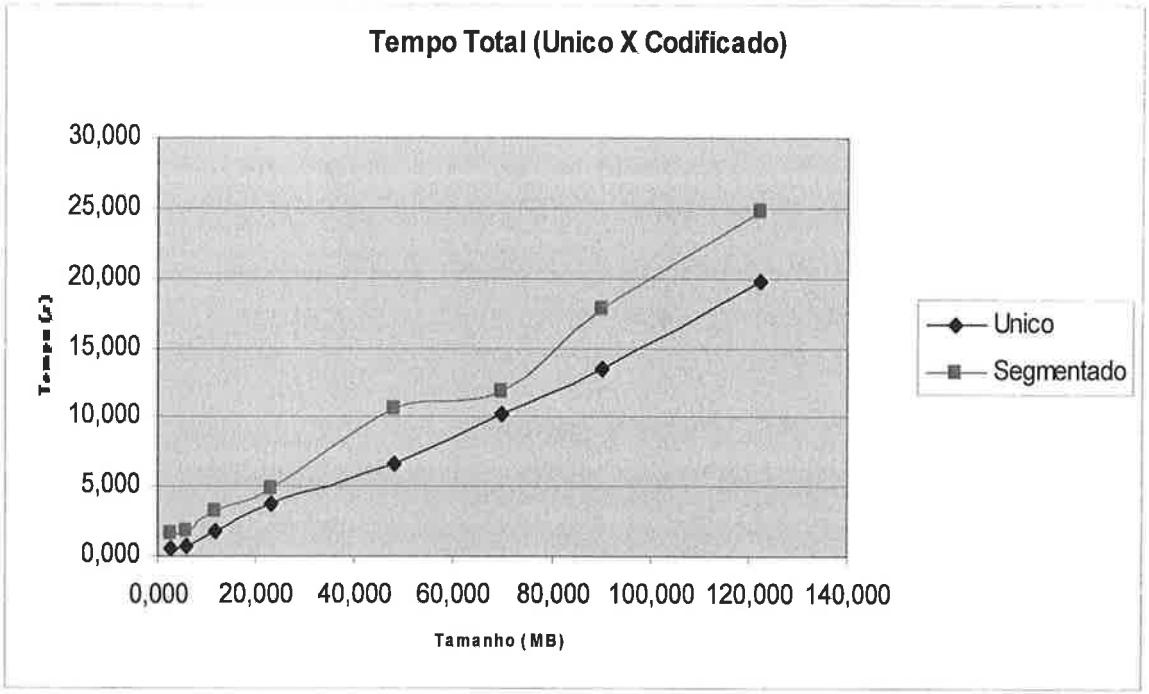

Figura 22. TEMPO TOTAL DE PROCESSAMENTO COM DOCUMENTOS CODIFICADOS.

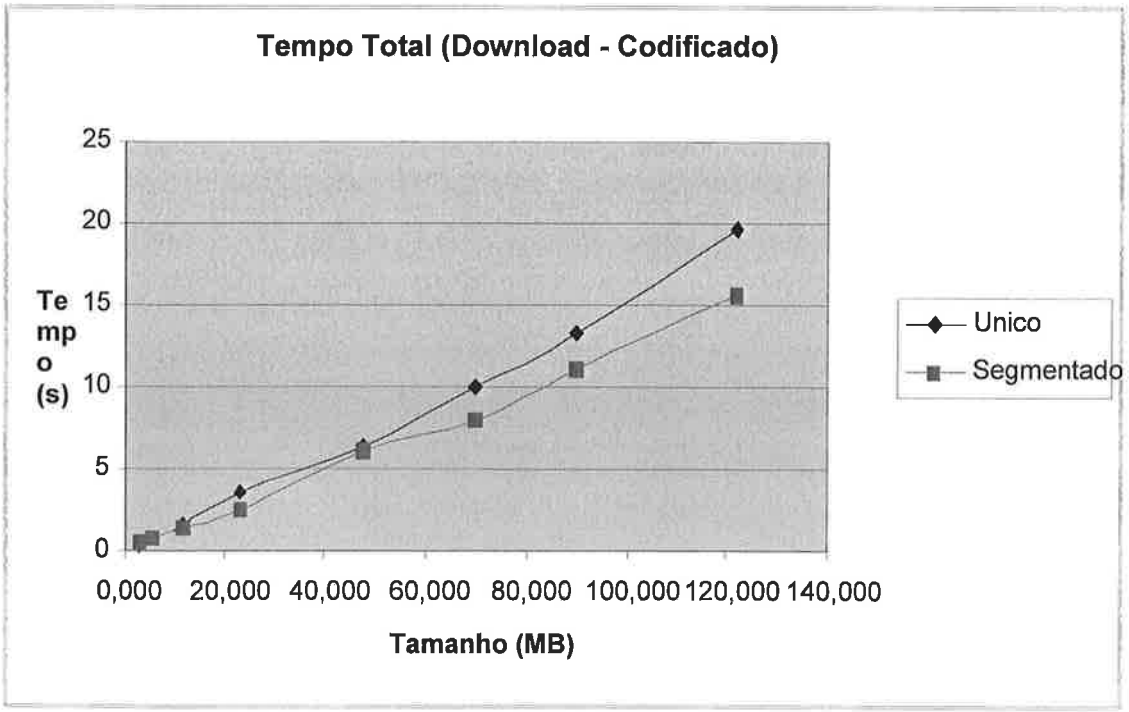

Figura 23. TEMPO TOTAL DE *DOWNLOAD* COM DOCUMENTOS CODIFICADOS.

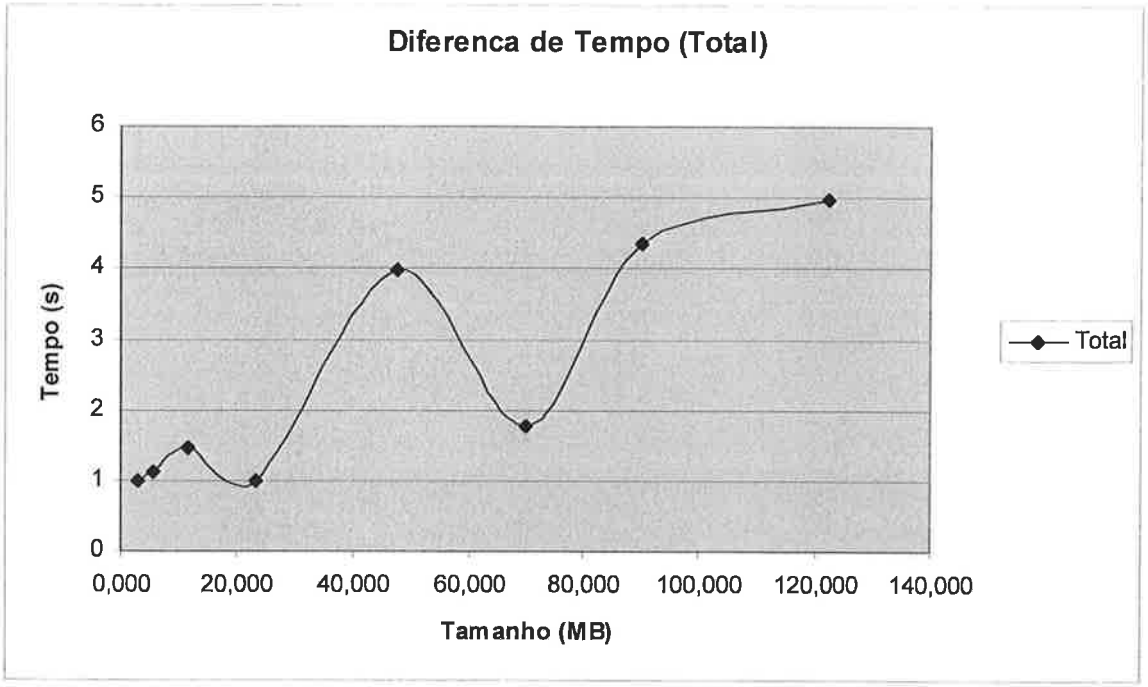

Figura 24. DIFERENÇA ENTRE OS TEMPOS TOTAIS COM DOCUMENTOS CODIFICADOS.

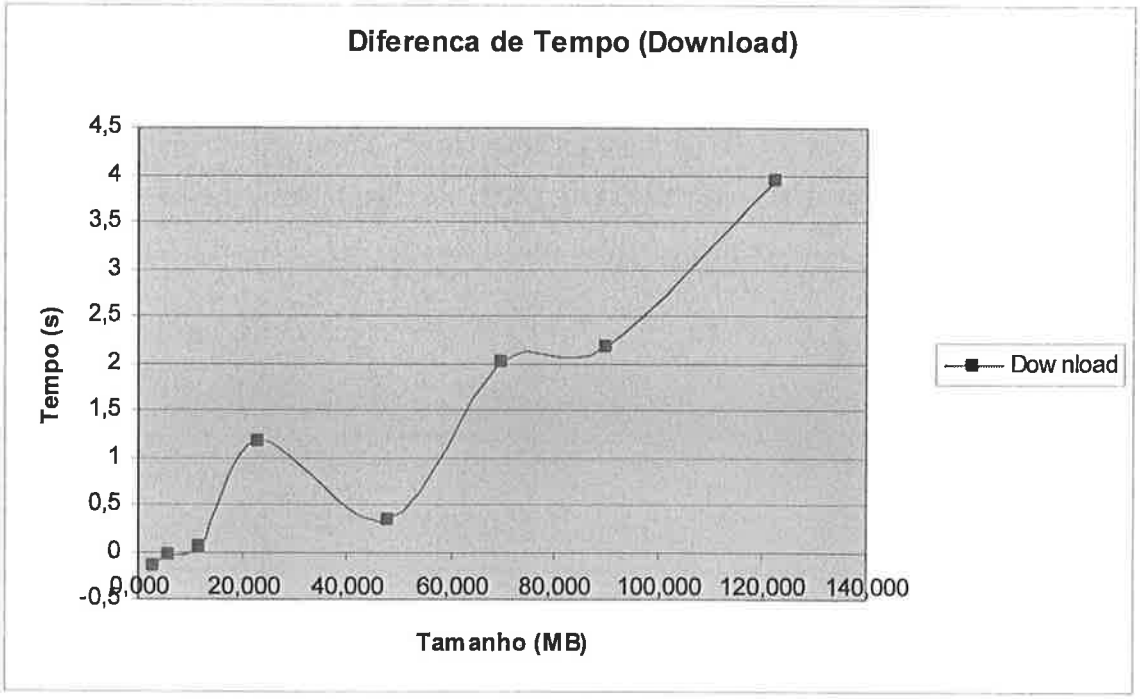

Figura 25. DIFERENÇA ENTRE OS TEMPOS DE *DOWNLOAD* COM DOCUMENTOS CODIFICADOS.

A próxima seqüência (Figura 26 a 29) apresenta os resultados obtidos pelas tomadas de tempo (Tempo total de processamento e tempo de download) dos documentos armazenados de maneira Segmentada X Codificado. Os gráficos confrontam os tempos destas duas formas de armazenamento.

Na Figura 26, são confrontados os tempos totais de processamento para as operações de consulta. Pode-se notar, neste caso, que temos um leve ganho para os documentos apenas segmentados, o que é evidente, pois os documentos codificados necessitam de um maior tempo de processamento para serem decodificados.

Na Figura 27, são confrontados os tempos totais de *download*. Para este caso, pode-se notar que praticamente não existe um ganho evidente entre os dois métodos de armazenamento.

Na Figura 28, são apresentadas as diferenças calculadas entre os tempos totais de processamento. Neste caso, pela análise realizada fica evidente que existe uma tendência de termos um melhor resultado para os documentos apenas segmentados em relação aos codificados. Isso é evidente, pois os documentos codificados necessitam de um tempo maior para serem decodificados. Em relação à diferença dos tempos de download (Figura 29), não existe ganhos que ficam evidentes para nenhum dos casos. Nota-se, ainda, na Figura 29, uma oscilação no gráfico; esta oscilação é devido às características do algoritmo de codificação, que, dependendo do tipo de documento armazenado, pode ocasionar uma redução no tamanho do mesmo, provocando então esta oscilação no gráfico.

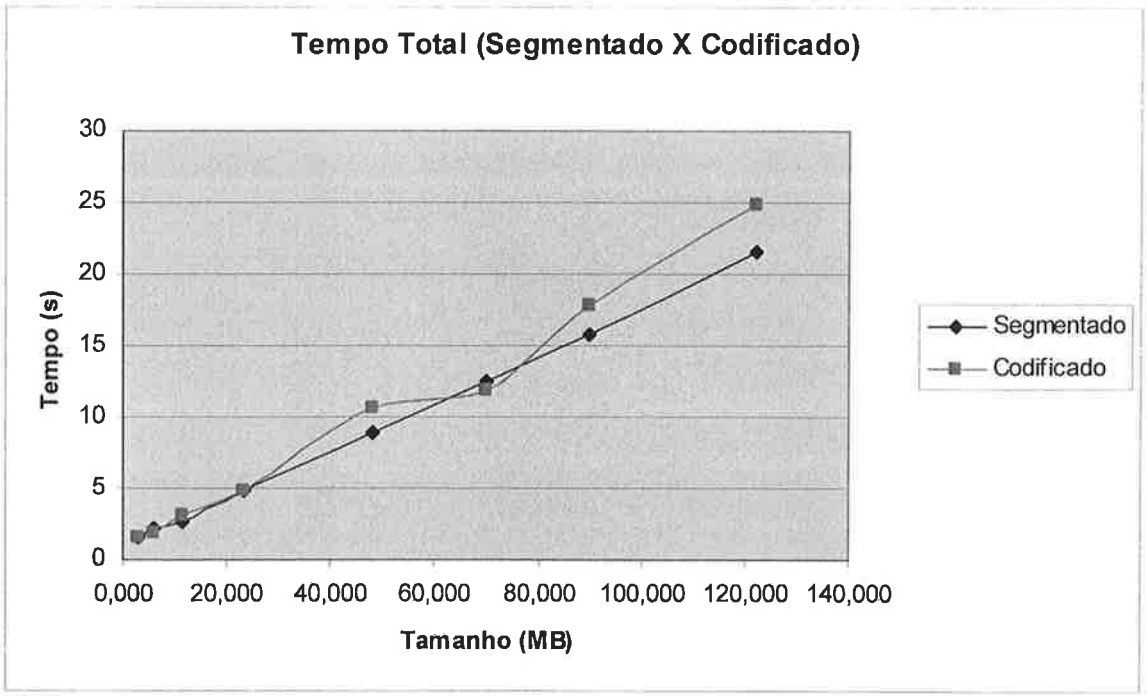

Figura 26. TEMPO TOTAL DE PROCESSAMENTO.

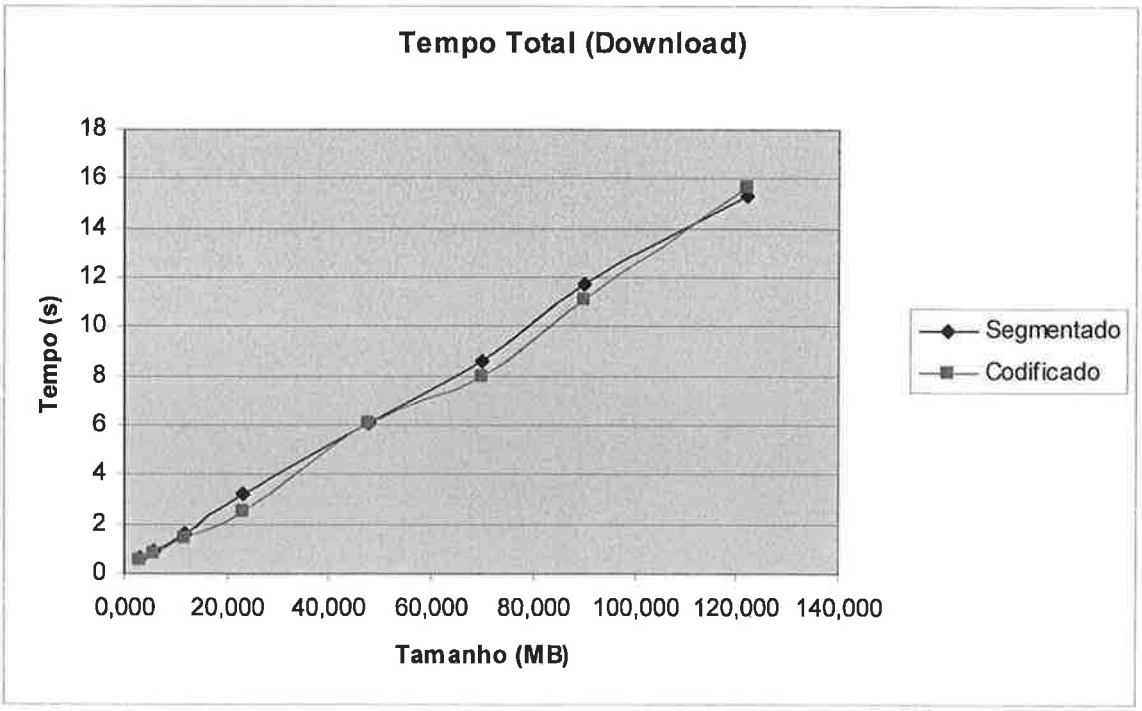

Figura 27. TEMPO TOTAL DE *DOWNLOAD*.

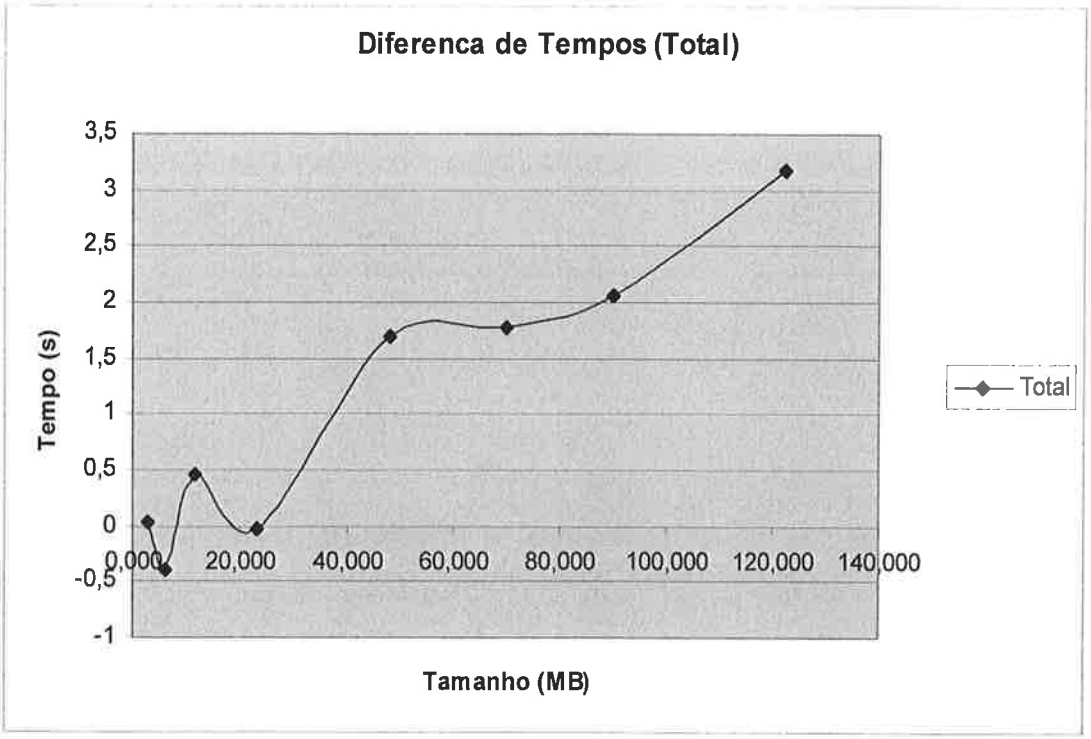

Figura 28. DIFERENÇA ENTRE OS TEMPOS DE PROCESSAMENTO.

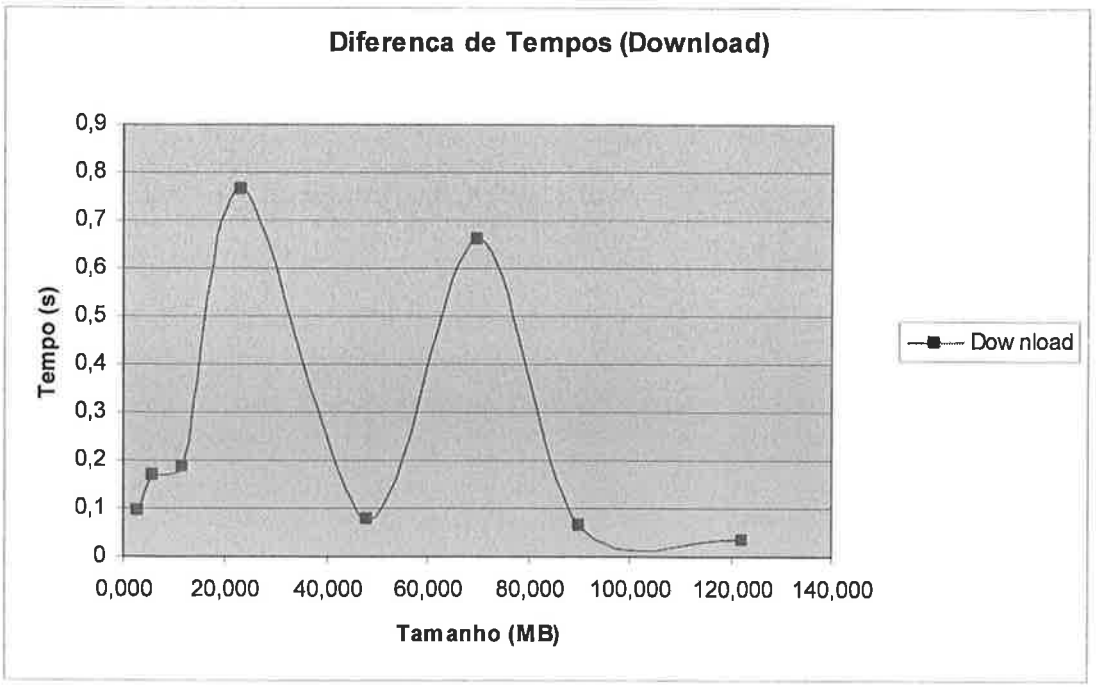

Figura 29. DIFERENÇA ENTRE OS TEMPOS DE *DOWNLOAD*.

Os resultados obtidos com os testes realizados, conforme foi mostrado neste capítulo, foram obtidos pela análise dos dados das tabelas 09 e 10.

TABELA 09. RESULTADOS OBTIDOS DOS DOCUMENTOS ÚNICOS E SEGMENTADOS

Nº.	Nome	Tamanho Total (K-byte)	Tipo	Tempos (s)
1	Teste01.wmv	23.303	U	T - 9.2687 - 9.6546 = 3.859 P - 9.2875 - 9.6531 = 3.656
2	Teste01.wmv	23.303	M	T - 4.1421 – 4.6312 = 4.890 P - 4.1687 - 4.6312 = 4.625 1 - 4.1937 - 4.5156 = 3.218 2 - 4.1906 - 4.4718 = 2.812 3 - 4.1906 - 4.4906 = 3.000 Total - 3.250 4.890 – 3.250 = 1.640
3	Teste02.ogm	47.859	U	T - 11.18406 - 11.24.984 = 6.578 P - 11.18593 - 11.24.984 = 6.390
4	Teste02.ogm	47.859	M	T - 12.39296 - 12.48.156 = 8.859 P - 12.39500 - 12.48.156 = 8.656 1 - 12.39750 - 12.45.671 = 5.921 2 - 12.39703 - 12.45.453 = 5.750 3 - 12.39843 -12.45.828 = 5.984 Total – 6.125 8.859 – 6.125 = 2.734
5	Teste03.ogm	69.907	U	T - 58.1293 - 58.22.250 = 10.156 P - 58.12296 - 58.22.250 = 9.953
6	Teste03.ogm	69.907	M	T - 58.40140 - 58.52.609 = 12.468 P - 58.40328 - 58.52.609 = 12.281 1 – 58.4131 - 58.49.546 = 8.515 2 – 58.40953 - 58.47.281 = 6.328 3 – 58.4115 - 58.48.296 = 7.281 Total – 8.593 12.468 – 8.593 = 3.875
7	Teste00.wmv	11.652	U	T - 16.19671 - 16.21.406 = 1.734 P - 16.19859 - 16.21.390 = 1.531
8	Teste00.wmv	11.652	M	T - 16.28484 - 16.31.218 = 2.734 P - 16.28500 - 16.31.218 = 2.718 1 - 16.28750 - 16.30.343 = 1.593 2 - 16.28703 - 16.30.265 = 1.562 3 - 16.28718 - 16.30.359 = 1.640 Total – 1.656 2.734 – 1.656 = 1.078
9	Teste08.wmv	5.826	U	T - 16.39687 - 16.40.421 = 0.734 P - 16.39687 - 16.40.421 = 0.734
10	Teste08.wmv	5.826	M	T - 50.27750 - 50.30.0 = 2.250 P - 50.27937 - 50.30.0 = 2.620 1 - 50.28156 - 50.29.93 = 0.937 2 - 50.28171 - 50.29.31 = 0.859 3 - 50.28250 - 50.29.156 = 0.906

				Total – 0.937 2.250 – 0.937 = 1.313
12	Teste09.wmv	2.913	U	T - 17.55593 - 17.56.156 = 0.562 P - 17.55781 - 17.56.156 = 0.375
13	Teste09.wmv	2.913	M	T - 26.49625 - 26.51.140 = 1.515 P - 26.49812 - 26.51.140 = 1.328 1 - 26.5000 - 26.50.421 = 0.421 2 - 26.5015 - 26.50.421 = 0.406 3 - 26.5093 - 26.50.609 = 0.515 Total – 0.609 1.515 – 0.609 = 0.906
14	Teste04.ogm	90.000	U	T - 54.54828 - 55.8.328 = 13.500 P - 54.5515 - 55.8.328 = 13.312
15	Teste04.ogm	90.000	M	T - 56.50843 - 57.6.625 = 15.781 P - 56.5131 - 57.6.625 = 15.593 1 - 56.51234 - 57.2.281 = 11.46 2 - 56.51218 - 57.2.984 = 11.765 3 - 56.51265 - 57.2.796 = 11.531 Total - 11.765 15.781 – 11.765 = 4.016
16	Teste05.ogm	122.240	U	T - 16.3762 - 16.56.875 = 19.812 P - 16.37250 - 16.56.875 = 19.625
17	Teste05.ogm	122.240	4.7.2.1.	T - 17.46140 - 18.7.750 = 21.609 P - 17.46328 - 18.7.750 = 21.421 1 – 17.46546 - 18.1.890 = 15.343 2 – 17.46578 - 18.1.234 = 14.656 3 – 17.46593 - 18.1.781 = 15.187 Total - 15.343 21.609 – 15.343 = 6.266

TABELA 10. RESULTADOS OBTIDOS DOS DOCUMENTOS CODIFICADOS E SEGMENTADOS

Numero	Nome	Tamanho (MB)	Tempo (s)
19	Teste00	11.652	T – 13:15:218 - 13:18:421 = 3:203 P – 13:15:234 - 13:18:375 = 3:140 1 – 13:15:453 - 13:16:843 = 1:390 2 – 13:15:421 - 13:16:843 = 1:421 3 – 13:15:500 - 13:16:859 = 1:359 Total = 1:469 - D = 1,734
20	Teste01	23.303	T – 17:39:953 – 17:44:812 = 4:859 P – 17:40:156 – 17:44:781 = 4:625 1 – 17:40:406 – 17:42:828 = 2:421 2 – 17:40:484 – 17:42:843 = 2:359

			3 – 17:40:640 – 17:42:890 = 2:250 Total = 2:484 - D = 2,375
21	**Teste02**	**47.859**	**T – 17:50:250 - 18:0:812 = 10:562 P – 17:50:250 - 18:0:781 = 10:531 1 – 17:50:812 - 17:56:671 = 5:859 2 – 17:50:625 - 17:56:546 = 5:921 3 – 17:50:750 - 17:56:640 = 5:890 Total = 6:046 - D = 4,516**
22	**Teste03**	**69.907**	**T – 18:45:310 - 18:16:468 = 11:937 P – 18:45:310 - 18:16:437 = 11:906 1 – 18:46:87 – 18:11:671 = 6:984 2 – 18:47:65 – 18:11:875 = 7:109 3 – 18:48:59 – 18:11:953 = 7:93 Total = 7:93 - D = 4,007**
23	**Teste04**	**90.000**	**T – 18:20:609 - 18:38:453 = 17:843 P – 18:20:625 - 18:38:421 = 17:796 1 – 18:20:812 - 18:31:812 = 11:0 2 – 18:20:796 - 18:30:562 = 9:765 3 – 18:20:921 - 18:31:930 = 10:171 Total = 11:134 D = 6,709**
24	**Teste05**	**122.240**	**T – 18:42:484 - 19:7:265 = 24:781 P – 18:42:500 - 19:7:234 = 24:734 1 – 18:42:750 - 18:58:375 = 15:625 2 – 18:42:703 - 18:57:953 = 15:250 3 – 18:42:765 - 18:58:312 = 15:546 Total = 15:672 - D = 9,109**
25	**Teste08**	**5.826**	**T – 19:12:531 - 19:14:390 = 1:859 P – 19:12:531 - 19:14:359 = 1:828 1 – 19:12:734 - 19:13:421 = 0:687 2 – 19:12:765 - 19:13:500 = 0:734 3 – 19:12:781 - 19:13:421 = 0:640 Total = 0,766 - D = 1,093**
26	**Teste09**	**2.913**	**T – 19:18:218 - 19:19:765 = 1:546 P – 19:18:234 - 19:19:734 = 1:500 1 – 19:18:453 - 19:18:921 = 0:468 2 – 19:18:406 - 19:18:875 = 0:468 3 – 19:18:500 - 19:18:921 = 0:421 Total = 0,515 - D = 1,031**

4.8. TRABALHOS CORRELATOS

Neste item será apresentado um comparativo entre este trabalho e duas outras propostas;

- Um Sistema de Banco de Dados Distribuídos Orientado a Objetos;
- Um sistema de arquivos distribuídos com a infra-estrutura IBP em conjunto com o ambiente de *Grid*.

Para o estudo do comportamento do mecanismo implementado, durante a realização dos testes, a aplicação utilizada foi uma Biblioteca Digital, que poderia fornecer vários tipos de documentos, os quais poderiam ter uma parte textual e uma outra parte, que poderia incluir imagens de alta definição, vídeos e aplicativos para serem instalados no usuário, conforme mostra a Tabela 02.

Em relação ao *Grid,* consideramos que ele é constituído por um modelo baseado em trocas, sem moeda, onde o compartilhamento tanto das informações armazenadas quanto do poder de processamento no atendimento aos seus usuários, será a retribuição mais que necessária, ou seja, os participantes do sistema cooperam apenas no intuito de maximizar seu ganho.

Um trabalho semelhante ao proposto [HOLUB04], implementa um ambiente para transcodificação de vídeo distribuído baseado em infra-estruturas de *Grid* computacionais e na infra-estrutura de armazenamento IBP (*Internet Backplane Protocol*). O ambiente é basicamente utilizado para encadear *jobs* em relação à localização do dado no armazenamento distribuído. Ele ainda fornece uma interface de acesso uniforme.

Verificamos que esse mesmo ambiente pode ser melhor implementado utilizando SGBD. Isso porque no ambiente proposto por Holub [HOLUB04], apesar de fornecer uma boa infra-estrutura de armazenamento e processamento, ele não disponibiliza consultas sobre os elementos armazenados, tarefa essa básica em qualquer sistema de Banco de Dados. Esse fator por si só pode ocasionar perda de

desempenho se, para avaliar o conteúdo, for necessário movimentá-lo para onde se encontra a ferramenta de análise.

Na nossa implementação utilizando um SGBD, além do acesso ser indexado ele poderá ser paralelizado. Os SGBD são desejáveis por facilitarem e tornarem ágeis as pesquisas, por balancearem a carga de trabalho e por gerenciarem grandes volumes de dados e, com isso, executarem melhor uma tarefa em relação a outros tipos de sistema de armazenamento.

No nosso trabalho, quando codificamos um determinado documento, prejudicamos a realização de pesquisas diretas no conteúdo dos arquivos codificados e segmentados onde ele está armazenado. É necessário executar todo o processo de recuperação para verificar se o documento serve ou não aos nossos propósitos. Contudo, isto apenas representa uma dificuldade não uma impossibilidade visto que a codificação (similarmente como a utilizada no IBP) serve para verificar a integridade dos arquivos no momento da sua reconstrução.

Özsu [OZSU02] em seu trabalho apresenta um Banco de Dados específico para vídeo, onde cada vídeo é segmentado em *clips* com intervalo de tempos comuns. Apesar deste trabalho tratar com fragmentos de um mesmo objeto de vídeo, ele define previamente o tamanho que cada segmento terá, a fim de possibilitar a sua reconstrução por meio de uma consulta baseada nos atributos semelhantes contidos em cada *clip* desse vídeo. O trabalho considera que o vídeo já foi segmentado e distribuído previamente e trata principalmente da sua reconstrução.

Se compararmos os trabalhos considerando apenas a forma de armazenamento, vemos que no nosso caso quem especificará o tamanho de cada fragmento é o ambiente. Não consideramos a possibilidade de trabalharmos com tamanhos fixos de fragmentos. Dentro da nossa proposta, isso representaria, dependendo do tamanho do documento que seria segmentado, um número quase ilimitado de servidores.

Ao considerarmos o sistema de gerência de Banco de Dados utilizado por Ozsu [OZSU02], notamos que o mesmo é orientado a objetos, o que dificulta consideravelmente o projeto, sua implementação e manutenção. O SGBD utilizado em nosso trabalho é relacional, com extensão para armazenamento de objetos e, portanto, seu projeto, implementação e manutenção é muito mais simples, inclusive se for necessário realizar correções de falhas da fase do projeto.

Um outro ponto importante a ser considerado é o ambiente. Apesar do trabalho de Ozsu utilizar Banco de Dados Distribuídos, o ambiente por nós implementado se comporta diferentemente de um ambiente previsto de Banco de Dados Distribuídos, pois cada servidor trabalha totalmente independente um do outro, tendo apenas o escalonador como integrador da aplicação.

Apesar de considerarmos para a implementação a replicação das tabelas de controle em cada um dos servidores, notamos que isso prejudicaria consideravelmente o desempenho do sistema como um todo visto que a atualização do status de cada um dos servidores teria que ser feita "de" e "para" todos os servidores do *Grid*. Esse tipo de implementação já tinha se mostrado inviável, em [LOURENSO00]. Contudo, para resolver o problema da realização de uma consulta integrada utilizamos o nó escalonador, semelhante ao coordenador de uma consulta em um ambiente distribuído de Banco de Dados. A diferença é que em um ambiente distribuído normalmente quem recebe a consulta fica responsável por ela, no nosso caso a responsabilidade sempre será do escalonador.

A finalidade dos *Grids* que possuem grandes bases de dados é a de disponibilizarem facilidades onde os usuários possam ter seus próprios bancos de dados, criando suas tabelas, índices e procedimentos, etc. Contudo, isso ainda não é uma realidade. Dentro desse contexto pode-se considerar que o objetivo deste trabalho, dentro da aplicação proposta, tenha sido alcançado.

4.9. CONCLUSÃO

Neste capítulo, foram descritas as características específicas da implementação e do ambiente de teste, como foi realizado o processamento, a certificação do usuário, a distribuição dos dados, seu relacionamento e como é a seqüência dos processos existentes neste protótipo de infra-estrutura de um ambiente de Banco de Dados baseado em *Grid*.

No capítulo 5, serão apresentadas as conclusões obtidas ao longo do estudo e os trabalhos futuros planejados para continuar este projeto.

Capítulo 5. CONCLUSÕES E TRABALHOS FUTUROS

"O único que podemos saber é que nada sabemos, e nisso está o vôo mais elevado da sabedoria humana".

Leon Tolstoi

Nesta seção, são apresentadas as conclusões obtidas com este trabalho e os possíveis trabalhos futuros.

5.1. Resultados

Neste trabalho, foi apresentada uma proposta de infra-estrutura com segmentação e distribuição de atributos complexos de dados em um Sistema de Banco de Dados baseado em *Grid*, que poderiam comprometer o desempenho do sistema, por possuírem um tamanho acima de um valor limite especificado. Assim, com equipamentos convencionais e programas de fácil aquisição, o projeto aumenta a aplicabilidade de sistemas com bases grandes de dados permitindo que entidades de pequeno porte possam investir nesta área.

Alguns dos problemas encontrados para a implementação do trabalho, como a certificação global do usuário, de um domínio administrativo poder utilizar um sistema que possui vários domínios administrativos diferentes, foi solucionado pela troca dos usuários válidos para cada domínio administrativo (seção de privilégios). Isso sem a ciência do usuário e, portanto, dispensando que o mesmo tivesse que ser cadastrado em cada um dos domínios administrativos existentes. Essa troca foi possível pela obrigatoriedade de cada domínio administrativo de se cadastrar no escalonador do *Grid*, fornecendo algumas características do ambiente como sistema operacional, tipo do Banco de Dados e um usuário e senha válida para a aplicação que será compartilhada.

Constatou-se que o método proposto para distribuição de dados (*Round-Robin*) em conjunto com a segmentação de atributos pelo número de servidores disponíveis é viável e poderá ser utilizado em diversas aplicações que necessitem trabalhar com dados de tamanho elevado em sistemas de Banco de Dados em *Grid* ou não.

Como o principal aspecto que determina a qualidade da solução encontrada para o problema de alocação de dados é a eficiência do projeto de alocação, ele deve

minimizar, tanto quanto possível, o custo das consultas executadas. O método desenvolvido visa possibilitar uma consulta desvinculada de um documento anexado em um único nó servidor, tentando distribuir o custo da consulta entre todos os servidores.

Em relação à economia de recursos, o método utilizou melhor os recursos disponíveis no sistema, principalmente em relação a espaço de armazenamento disponível no *Grid*. O sistema inserirá os dados de acordo com a política especificada de maneira transparente para o usuário.

A transparência do sistema é outra característica importante, como comprovado pelo módulo de Entrada e o módulo de Recuperação - o usuário saberá que os seus dados estão armazenados no sistema, mas não saberá onde estão. Contudo, ele poderá efetuar qualquer operação sobre seus dados de forma transparente (Entrada, Recuperação Atualização e Exclusão) [3].

Apesar da implementação ter sido feita em um ambiente homogêneo, tanto de hardware como de software, o desenvolvimento do projeto e do seu protótipo foi feito pensando em um ambiente de *Grids* reais, geralmente heterogêneos.

Nossa solução foi implementada utilizando o ambiente *PtoNet* (*ADO.Net*) e considera a utilização dos códigos LDPC para demonstrar o funcionamento das técnicas apresentadas permitindo receber novas propostas e implementações. Isto resulta em uma proposta viável de aplicação que pode ser aplicada em sistemas que necessitem gerenciar grandes bases de dados em ambientes de Sistema de Banco de Dados em *Grid*.

Verificamos que com a utilização de um SGBD, pode-se melhorar o desempenho de aplicações de processamento intensivo de dados. Contudo,

[3] **Os módulos de Alteração e Exclusão não foram implementados**

geralmente, para a obtenção de um melhor desempenho as aplicações necessitam ser customizadas para ambiente.

5.2. LIMITAÇÕES DO TRABALHO

No nosso trabalho, o funcionamento do sistema não foi especificado visando a determinação da escalabilidade. Contudo, esta característica poderá ser abordada em outros trabalhos, a fim de determinar efetivamente as limitações desse tipo de solução.

Outro ponto a ser considerado, é a máquina onde as aplicações cliente estão sendo rodadas. A sobrecarga de processamento ocasionada pela codificação / decodificação e pela segmentação/recomposição pode acarretar queda no desempenho ou, até mesmo, a interrupção do processamento, se a quantidade de memória de trabalho não for suficiente.

Foi verificado que o esquema proposto possui um ponto de falha, que é o escalonador do sistema. É nele que estão as principais tabelas de controle da infra-estrutura, a utilização de controle de concorrência e alternativas para aumentar a tolerância às falhas deste servidor, seriam umas das possíveis soluções.

Outra limitação é o uso do modelo Cliente/Servidor no protótipo. No caso da proposta ser direcionada para o ambiente *Web Services*, essa limitação seria contornada.

5.3. TRABALHOS FUTUROS

O protótipo implementado da infra-estrutura mostrou-se viável. Contudo, algumas mudanças devem ser realizadas para utilizá-lo em ambientes de *Grid*s reais tais como:

- Convertê-lo da plataforma Cliente/Servidor para a plataforma de Servidores Web, essa mudança permitirá utilizar dentro do ambiente previsto de aplicações em *Grid*, contudo, a solução utilizando o ambiente Cliente/Servidor não será totalmente descartada podendo ser utilizado como uma outra forma de abordagem;
- É necessário concluir os demais módulos do sistema (alteração e exclusão) bem como os de gerenciamento (Verificação e monitoramento dos servidores), e cadastramento dos usuários do *Grid*. Porém, o módulo mais complicado de ser desenvolvido será a interface de consultas, considerando a transparência para o usuário do ambiente e a possível diversidade de sistemas.
- Avaliação em ambiente real usando a infra-estrutura de rede Gigabit nacional ou estadual.

Anexos

ANEXO - 1

Sistema de Banco de Dados Distribuídos

Conforme C. J. Date [DATE00], um sistema distribuído é qualquer sistema que envolva múltiplas localidades conectadas umas às outras por uma espécie de rede de comunicações, na qual um usuário de uma das localidades possa acessar os dados armazenados em outra localidade transparentemente.

Este conceito foi pioneiramente usado no final dos anos 70 no projeto de pesquisa da IBM o R* STAR.. Contudo, somente a partir de 1987, com o INGRES/STAR, foi empregado este conceito em um sistema de Banco de Dados comercial.

As Doze Regras para um SGBDD.

1. ***Autonomia Local***: Os locais em um sistema distribuído serão autônomos, ou seja, serão independentes uns dos outros;
2. ***Não existirá um local centralizador***: Um SGBD. Distribuído não poderá ser confiado a um local centralizador, pois isto cria um ponto de falha no sistema, prejudicando o desempenho e criando gargalos no acesso aos dados;
3. ***Operações Contínuas***: Um SBD Distribuído poderá nunca ser desligado.
4. ***Transparência de Localização e Independência de localização***: Aplicações e usuários não precisarão conhecer onde o dado é armazenado;
5. ***Independência de Fragmentação***: Para um sistema de DB. Relacional, tabelas poderão ser divididas em fragmentos e armazenadas em diferentes localizações, transparentes para usuários e aplicações.;
6. ***Independência de Replicação***: Dados poderão estar disponíveis para serem replicados, transparentemente, para múltiplos sistemas de computadores em ambientes distribuídos;

7. ***Processamento de query Distribuída***: O desempenho de uma dada query poderá ser independente da localização em que a query foi submetida;
8. ***Gerenciamento de Transações Distribuídas***: Um sistema distribuído poderá estar capacitado para suportar as propriedades do ACID para transações;
9. ***Independência de Hardware***: Um SBD. Distribuído poderá estar capacitado para operar e acessar dados através de uma grande variedade de plataformas de hardware;
10. ***Independência de Sistemas Operacionais***: Um SBD. Distribuído poderá estar capacitado para rodar em diferentes sistemas operacionais;
11. ***Independência de Rede***: Um SBD. Distribuído poderá estar capacitado para rodar sem considerar os protocolos de comunicação ou a topologia usada para interconexão de vários nós do sistema;
12. ***Independência de SGBD***: Um sistema ideal poderá estar capacitado a suportar interoperabilidade entre sistemas de SGBD heterogêneos rodando em diferentes nós.

Um verdadeiro SGBDD necessita que o sistema suporte atualizações em qualquer nó em uma rede. As regras descritas acima fornecem alguns dos requisitos que um SGBDD deve suportar, conforme requisitos propostos por C. J. Date [DATE00].

Banco de Dados Distribuídos tem como proposta, permitir aos usuários trabalharem com múltiplos Bancos de Dados, possivelmente localizados em diferentes locais, como se eles estivessem trabalhando com um simples Banco de Dados compartilhado, integrado em um sistema simples. Em resumo, Banco de Dados Distribuídos permite ao usuário atualizar uma cópia simples do dado sem considerar onde o dado está armazenado. Redundância e dados inconsistentes são eliminados, como se as operações estivessem sendo realizadas em um BD simples, embora potencialmente espalhado por múltiplos sistemas.

Qualquer aplicação necessitando acesso remoto paga uma severa penalidade no desempenho. Como a transmissão de requisitos e dados são de ordem de magnitude menor que o acesso local e o usuário deve aguardar seu término mais a transmissão adicional do *2-phase-commit* para assegurar integridade. O declínio da

disponibilização de todo ambiente das transações, agora envolve não só o sistema local, mas a rede e todos os nós contidos nestas.

A complexidade administrativa intensifica-se, como cada local deve coordenar a manutenção com todos os outros para permitir indisponibilidade dos dados. Alterações nas aplicações devem ser coordenadas entre todos os usuários como se estivessem em um simples BD local.

Adicionalmente *Queries* tem sido sempre um problema no sistema operacional. Com o Banco de Dados Distribuídos o problema é exacerbado, com *Joins* distribuídos através da rede. Isto pode resultar em movimentação de um grande volume de resultados parciais de dados entre locações em tempos não previstos. Com o número de nós aumentando, tempo de resposta e desempenho do sistema degradam, tornando a solução Banco de Dados Distribuídos impraticável para muitas aplicações.

Banco de Dados Distribuídos é mais complicado pelo fato de que com novos sistemas sendo implementados através das empresas, usuários tentam selecionar ótimas plataformas e SGBDD para seus ambientes. Consequentemente, muitas das grandes empresas agora têm oito ou nove SGBD diferentes instalados. A solução do BDD descrita aplica-se estritamente para BDD homogêneos, onde todos os nós têm o mesmo SGBD. As complicações de gerenciar uma rede heterogênea e aplicações adaptativas para o menor denominador comum de linguagem de SGBD, tornam tal ambiente quase impraticável, consequentemente, implementações de BDD heterogêneas são muito raras.

ANEXO - 2

Os Códigos LDPC

Conforme Ryan [RYAN04] os códigos LDPC possuem três tipos distintos, todos baseados em grafos bipartidos que são gerados aleatoriamente por uma distribuição de probabilidade.

Os grafos têm **E + D** nós, particionados em dois conjuntos – os nós esquerdos $\mathbf{e_1,...,\ e_E}$, e os nós direitos, $\mathbf{d_1,...d_D}$. Cada um dos nós esquerdos é conectado a cada um dos direitos. Uma classe de grafos G é definida por duas distribuições de probabilidade, Λ e P. Estes são vetores compostos de elementos $\mathbf{\Lambda_1}$, $\mathbf{\Lambda_2,...}$ e $\mathbf{P_1,P_2,...,}$ tal que $\mathbf{\Sigma\Lambda_i = 1}$ e $\mathbf{\Sigma\ _i = 1}$. Seja g um grafo em **G.** $\mathbf{\Lambda_i}$ é a probabilidade que um nó esquerdo em g tenha exatamente i arestas saindo, similarmente, P_i é a probabilidade que um nó direito de g tenha exatamente i arestas chegando. Dados E, D, Λ e P, gerando um grafo g é em teoria uma tarefa de busca.

Para descrever os códigos, considera-se que tenham quatro entradas, as quais podem gerar grafos bipartidos com elas. Para descrever os códigos, vamos considerar que tenhamos **n** blocos de dados de tamanhos iguais, os quais nós desejamos codificar em **n + m** blocos iguais, e os quais nós distribuiremos em uma rede. Os nós dos grafos LDPC carregam tais blocos de dados e, portanto, usaremos os termos "nós" e "blocos" intercambiáveis. Cada nó pode inicializar seus valores de blocos de dados, ou eles podem calculá-los de outros blocos de dados. A única operação usada para esses cálculos é da paridade, como é comum em *arrays* de disco de RAID nível 5. Cada método de geração de código utiliza esses grafos para definir uma codificação de **n** blocos de dados em **n + m** blocos que serão distribuídas na rede.

Na decodificação, normalmente se assume que nós baixaremos fn blocos fechados, $B_{1,...,} B_{fn}$ na ordem. Sobre estes nós pode-se calcular o bloco de dados **N** original.

No caso dos Códigos Sistemáticos (***Systematic Codes***): normalmente assume-se $E = n$ e $D = m$. Neste caso, cada nó esquerdo c_i carrega os **i**-enésimos blocos de dados, e cada nó direito d_i é calculado sendo exclusivo ou de todos os nós esquerdos que são conectados a este. Um exemplo simples é mostrado na Figura 30(a).

Esses códigos podem cascatear pela exploração $d > 1$ níveis de grafos bipartidos, $g_{1,...,} g_d$, onde os nós direitos de g_i são também os nós esquerdos de g_{i+1}. Os grafos de nível 1 têm $L = n$, e seus nós contém n blocos de dados. O restante dos blocos de codificados estão nos nós do lado direito do grafo **d**. Assim $m = P_{d\ i=1}\ D_i$.

Um simples nível de árvore cascateada de códigos Sistemáticos é mostrado na Figura 30(b)

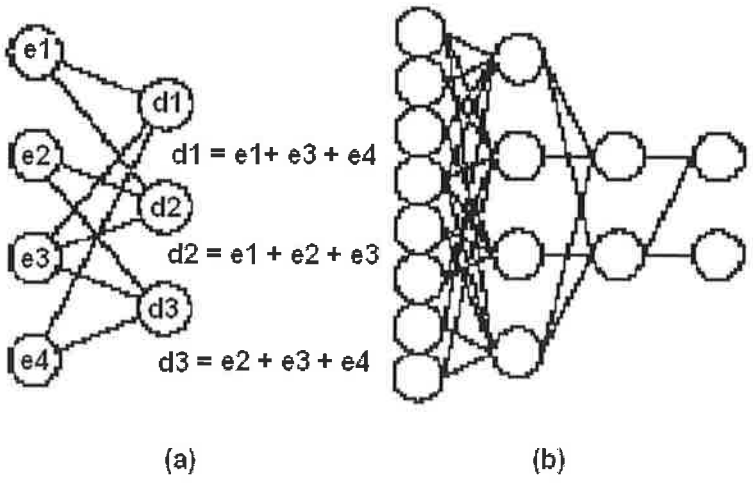

FIGURA 30. : (A) EXEMPLO 1 - NÍVEL DE CÓDIGOS SISTEMÁTICOS PARA N = 4, M = 3. (B) EXEMPLO 3 - NÍVEL DE CÓDIGOS SISTEMÁTICOS PARA N = 8, M = 8 [RYAN04]

A codificação e decodificação, regular e cascateada dos códigos Sistemáticos, são operações de busca e ambas as operações possuem tempo linear conforme o número de arestas no grafo.

Para o caso dos Códigos Gallager não Sistemáticos (***Unsystematic***), os quais foram introduzidos em 1960, adota-se **E = n + m**, e **D = m**. O primeiro passo da criação dos códigos Gallager é o uso **g** para gerar uma matriz **M** de **(n + m) × n**. Isto é empregado para calcular a codificação dos blocos **n + m** do bloco de dados **n** original. Estes blocos são armazenados nos nós esquerdos de **g**. Os nós direitos de **g** não carregam dados, mas no seu lugar estão as restrições dos nós – cada $\mathbf{d_i}$ tem a propriedade (garantida pela geração de **M**) exclusividade ou de todos os nós o incidente para isto é zeros. Um simples código Gallager é mostrado na Figura 31(a).

A codificação, nesse caso, é uma operação cara, envolvendo a geração da matriz **M**, e os cálculos dos blocos codificados. Felizmente, os grafos têm baixa densidade (i.e. a cardinalidade media dos nós é pequena), **M** é uma matriz esparsa, é usada na codificação e decodificação, não sendo tão custosa como uma matriz densa (como é o caso dos códigos *Reed-Solomon*). Decodificação é linear conforme o número de arestas no grafo.

Casualmente, **M** necessita somente ser gerada um por grafo; e, então, esta pode ser usada para todas as operações de codificação / decodificação.

Código IRA (*Irregular Repeat-Accumulate*): Estes códigos são códigos Sistemáticos como **E = n** e **D = m**, e os blocos de informação são armazenados nos nós esquerdos. Contudo, um conjunto extra de **m** nós, $\mathbf{z_1}$,..., $\mathbf{z_m}$, são adicionados para o grafo na seguinte direção. Cada nó $\mathbf{d_i}$ tem uma aresta para $\mathbf{z_i}$. Adicionalmente, cada nó $\mathbf{z_i}$ tem uma aresta para $\mathbf{z_{i+1}}$, para $\mathbf{i < m}$. Estes nós extras são chamados de nós acumuladores. Para codificar, somente os blocos nos nós acumuladores e esquerdos, serão armazenados – os nós em **D** são simplesmente usados para calcular a codificação e a decodificação e estes cálculos procedem exatamente como nos códigos Sistemáticos. Um exemplo de grafo IRA é mostrado na Figura 31(b).

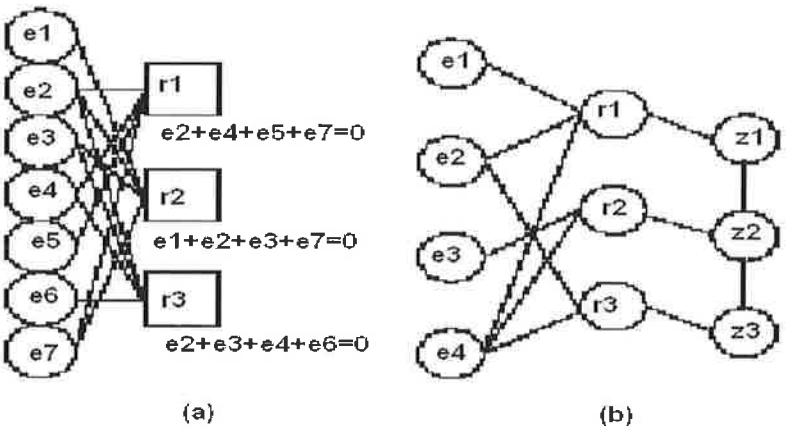

FIGURA 31: (A) EXEMPLO DO CÓDIGO GALLAGER PARA N = 4, M = 3. NOTE QUE OS NÓS DIREITOS DEFINEM RESTRIÇÕES ENTRE OS NÓS ESQUERDOS, E NÃO ARMAZENAM BLOCOS DCODIFICADOS. (B) EXEMPLO CÓDIGOS IRA PARA N = 4, M = 3. OS NÓS ACUMULADORES ESQUERDOS SÃO ARMAZENADOS COMO BLOCOS CODIFICADOS. OS NÓS DIREITOS SÃO USADOS PARA CÁLCULOS [RYAN04].

CÓDIGOS DE PROPRIEDADES ASSINTÓTICAS

Todas as três classes de códigos LDPC têm recebido análises assintóticas que são descritas a seguir. Uma taxa $D = n + m$ é selecionada, e então os vetores Λ e P são designados. Esse pode ser comprovar que os grafos gerados das distribuições em Λ e P podem assintoticamente alcançar a capacidade total do canal de comunicação.

Em outras palavras, eles podem ser decodificados com sucesso com **fn** blocos baixados, onde **f** aproximações para **1** de **n** aproximações para ∞.

Infelizmente, no mundo real, desenvolvedores de grandes sistemas de armazenamento não podem quebrar seus dados em infinitos pequenos pedaços. Limitações no número de dispositivos físicos, mas o fato de que pequenos blocos de dados não são transmitidos tão eficientemente como grandes blocos, considerando que **n** pode atingir um único dígito em milhares. Entretanto, outra questão sobre os códigos LDPC, é que eles executam melhor quando **n** está no intervalo.

ANEXO - 3

Determinação da Intensidade Computacional

Segundo Taurion [TAURION04], é possível determinar o quanto uma tarefa pode necessitar de processamento em um ambiente baseado em *Grid*. Essa verificação pode ser feita utilizando a fórmula que calcula a intensidade computacional (IC). Para verificar se uma dada aplicação aproveitará os recursos computacionais de um *Grid,* o indicador IC representa o percentual relativo ao tempo gasto ao mover dados de e para o PC, comparando ao tempo despendido executando as computações nestes dados.

Esse indicador pode ser determinado pela seguinte fórmula:

IC = (4 * duração da unidade de trabalho em segundos) / (entrada em KB + saída em KB).

Considerando uma aplicação que executa uma unidade de trabalho em 20 minutos (1.200 segundos), consumindo 1 MB (1.000 KB) de dados de entrada e produza 0,2 MB de saída. Pela fórmula teremos:

IC = (4 * 1200) / (1000 + 200) = 4.

Um outro exemplo, onde uma aplicação executa uma unidade de trabalho em 10 minutos (600 segundos), consumindo 2 MB (2.000 KB) de entrada e gerando 0,8 MB (800 KB) de saída. Pela fórmula teremos:

IC = (4 * 600) / (2000 + 800) = 0,85.

De forma geral, considera-se que, para ser adequado operar em um ambiente baseado em *Grid*, o IC devera se maior que 1. Quanto mais elevado for o IC de uma aplicação, a torna mais adequada a operar em *Grid*.

É evidente que inúmeras variáveis podem influenciar o cálculo do IC, como velocidade da rede interna, equipamentos utilizados, etc. Contudo, devido a sua simplicidade, é uma importante ferramenta para determinar se uma aplicação poderá ou não obter vantagens da sua implantação em um sistema baseado em *Grid*.

Referências

"Eu penso 99 vezes e não chego à conclusão alguma, mas, quando paro de pensar, surge a verdade."

(A. Einstein)

[ATKINSON03] ATKINSON, M.P.; DIALANI, V.; GUY, L.; NARANG, I.; PATON, N.W.; PEARSON, D.; STOREY, T. and WATSON, P. *Grid* Database Access and Integration: Requirements and Functionalities, Global *Grid* Forum Informational Document, GFD.13, 2003.

[BAIÃO98] BAIÃO, F.A; MATTOSO, M.L.Q. A Mixed Fragmentation Algorithm for Distributed Object Oriented Databases - Proc. 9th Int. Conf. on Computing and Information (ICCI'98)- also to appear in Special Issue of the Journal of Computing and Information, Winnipeg, Canada, June 1998, pp.141-148.

[BAIÃO00] BAIÃO, F.A; MATTOSO, M.L.Q.; ZAVERUCHA, G. Horizontal Fragmentation In ODMSS: New Issues and Performance Evaluation - 19th IEEE International Performance, Computing, And Communications Conference -- IPCCC 2000, IEEE CS Press, Phoenix, Arizona, Feb. 2000, ISBN 0-7803-5979-8, ISSN 1097-2641, pp.108-116.

[BAIÃO04] BAIÃO, Fernanda; WILDEMBERG, Matheus; MATTOSO, Marta; .Alocação de Dados em Bancos de Dados Distribuídos – Simpósio Brasileiro de Banco de Dados – SBBD – Manaus, Amazonas, Brasil, Anais/Proceedings. UFAM 2003.

[BREITBAR97] ANDERSON, Todd; BREITBART, Yuri; KORT, Henry F.; WOOL, Avishai. Replication, Consistency, and Practicality: Are These Mutually Exclusive? ACM-SIGMOD'98, Seattle, junho 1998.

[CASANOVA85] CASANOVA, M. A.; MOURA, A. V. Princípios de Sistemas de Gerência de Bancos de Dados Distribuídos. Rio de Janeiro: Campus, 1985.

[CIRNE02] CIRNE, Walfredo. *Grid*s Computacionais: Arquitetura, Tecnologia e Aplicações - Proceedings of the Third Workshop on High Perfomance Computing Systems, October 2002.

[COUCEIRO84] COUCEIRO, L. A.; BARRENECHA, H. F. Sistemas de Gerência de Banco de Dados Distribuídos. Rio de Janeiro: Livros Técnicos e Científicos Editora, 1984.

[CYGWIN05] Cygwin. Emulador de LINUX para ambiente WINDOWS http://www.cygwin.com - Acessado em 17/07/2005.

[DATE00] DATE, C.J. Introdução a Sistemas de Banco de Dados, (traduzido da 5ª edição americana) – Editora Campus – 1999, capítulos 1, 2.

[EICKLER97] EICKLER, André; KEMPER, Alfons; KOSSMANN, Donald. Finding Data in the Neighborhood. Disponível por WWW em 17-Mai-1999. Em: http://www.db.fmi.uni-passau.de/~eickler/DNames/dnames.html

[FOSTER99] FOSTER, Ian. Computacional *Grids* - Chapter 2 of the *Grid*: Blueprint for a New Computing Infrastructure, 1999.

[FOSTER01] FOSTER, Ian. The Anatomy of the *Grid* - 2001- http://www-fp.mcs.anl.gov/~foster/ . Acessado em 20/09/2004.

[GENOME05] The Human Genome Project. http://bio.lsd.ornl.gov/mgd/index.html Acesso em 20/07/2005.

[GRAY02] GRAY, J.; SZALAY, A.S.; THAKAR, A.; KUNSZT, P.; STOUGHTON, C.; SULTZ, D.; VANDENBERG, J. "The Sloan Digital Sky Survey Science Archive: Migrating a Multi-Terabyte Astronomical Archive from Object to Relational DBMS", Distributed Data & Structures 4: Records of the 4th International Meeting, pp 189-210 W. Litwin, G. Levy (eds), Paris France March 2002, Carleton Scientific 2003, ISBN 1894145-13-5, also MSR-TR-2002-01, January 2002.

[GRAY04] GRAY, Jim. The Revolution in Database Architecture - Association for Computing Machinery, Inc. Extended abstract of talk at ACM SIGMOD 2004, Paris, France, June 2004.

[HACKATHORN93] HACKATHORN, Richard D. Enterprise Database Connectivity. New York: John Wiley & Sons, 1993. 326 p.

[HOLUB04] HOLUB, Petr and HEJTMANEK, Lukas. Distributed Encoding Environment Based on *Grids* and IBP Infrastructure - Terena Networking Conference 2004, Rhodes, Greece, June, 2004.

[JAGATHEESAN02] JAGATHEESAN, A.; MOORE, R.; RAJASEKAR, A. and ZHU, B. "Virtual Services in Data *Grids*", In the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC), July 2002, Scotland.

[KORT99] KORTH, H.F. and SILBERSCHATZ, A. - Sistemas de Banco de Dados. McGraw-Hill, 3ª edição, 1999.

[LIU03] LIU, David T.; FRANKLIN, Michael J. and PAREKH, Devesh. GridDB: A Relational Interface to the *Grid*, Demo at ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003.

[LORENZO04] BOTE-LORENZO, M. L.; DIMITRIADIS, Y. A. and SÁNCHEZ, E. Gómez. "*Grid* Characteristics and Uses: A *Grid* Definition", Across Grids 2003, LNCS 2970, pp 291-298, 2004. eds F. Fernández Rivera et al. (Springer-Verlag Berlin Heidelberg 2004).

[LOURENSO01] LOURENSO R. "Object Replication On-Demand in DDBMS for Load Balance Cluster" presented in "5th. World Multiconference on Systemics, Cybernetics and Informatics" de 22 a 25 de julho de 2001 – Orlando, Florida, USA.

[LOURENSO05] LOURENSO R. "Relatório Técnico" elaborado como proposta inicial para esta tese e como trabalho em conjunto com os pesquisadores do IBP em setembro de 2004.

[MALAIKA03] MALAIKA, Susan; EISENBERG, Andrew ; MELTON, Jim. Standards for Databases on the *Grid* - ACM SIGMOD, ISSN:0163-5808, 2003, pages 92-100.

[MARÍA04] MARÍA, A. Nieto-Santisteban; SZALAY, Alexander S.; THAKAR Aniruddha R.; O'MULLANE William J.; GRAY, Jim; ANNIS, James. When Database Systems Meet the *Grid*, 2004.

[MATTOSO98] BAIÃO, F.A; MATTOSO, M.L.Q.; ZAVERUCHA, G. Towards an Inductive Design of Distributed Object Oriented Databases - Proc. Third IFCIS International Conference on Cooperative Information Systems (CoopIS'98), IEEE CS Press, New York, E.U.A., August 1998, pp.188-197.

[MATTOSO00] BAIÃO, F.A; MATTOSO, M.; ZAVERUCHA, G. A Methodology for the Design of Distributed OODBMS - In: Proceedings of the ECOOP 2000 Workshop for PhD Students in Object-Oriented Systems (PhDOOS'2000), Sophia Antipolis, France, June 2000.

[MOORE04] MOORE R.; Evolution of Data *Grid* Concept – Submetido para o GLOBAL *GRID* FORUM DATA AREA WORKSHOP, Janeiro, 2004.

[MONTANI92] MONTANI, C.; PEREGO, R.; SCOPIGNO, R. Parallel volume visualization on hypercube architecture. In: SYMPOSIUM ON VOLUME VISUALIZATION, Boston, 1992. Proceedings. New York: ACM Press, 1992, p. 9-16.

[NAVATHE05] NAVATHE, SHAMKANT B.; RAMEZ ELMASRI. Sistema de Banco de Dados. 4ª edição – Pearson – 2005, capítulos 1,3,4,6,8.

[NIEH92] NIEH, J.; LEVOY, M. Volume rendering on scalable shared-memory MIMD architectures. In: SYMPOSIUM ON VOLUME VISUALIZATION, Boston, 1992. Proceedings. New York: ACM Press, 1992, p. 17-24.

[OZSU99] ÖZSU, M. Tamer; VALDURIEZ, Patrick. Priciples of Distributed Database Systems, second edition - Prentice Hall – 1999, capítulos 1, 4, 5, 6, 12, 14.

[OZSU02] CHEN L. and ÖZSU M. T. Modeling of Video Objects in a Video Databases - IEEE International Conference on Multimedia, Lausanne, Switzerland, August 2002, 217-221.

[PATON03] PATON, N.; ATKINSON M.; DIALANI, V.; PEARSON, D.; STOREY, T. and WATSON, P. "Databases Access and Integration Services on the Grid." UK e-Science Programme Technical Report UKeS-2002-03. National e-Science Centre. http://www.nesc.ac.uk/technical_papers/dbtf.pdf.

[PLANK04] PLANK, James S. and THOMASON, Michael G. A Practical Analysis of Low-Density Parity – Check Erasure Codes for Wide – Area Storage Applications – The International Conference on Dependable Systems and Networks, IEEE, Florence, Italy, June, 2004.

[RAJASEKAR02] RAJASEKAR, Arcot.; WAN, M. and MOORE, R. MySRB & SRB – Components of a Data *Grid*. In High Performance Distributed Computing, (2002).

[RAJASEKAR04] RAJASEKAR, Arcot; WAN, Michael; MOORE Reagan; SCHROEDER, Wayne. Data *Grid* Federation - San Diego Supercomputer Center 9500 Gilman Drive, MC-0505, January 2004.

[ROURE03] ROURE, D.; BAKER, M.;0 JENNINGS, N. R. and SHADBOLT, N. (2003) The evolution of the Grid, in Berman, F., Fox, G. and Hey, A. J. G., Eds. Grid Computing - Making the Global Infrastructure a Reality, pages pp. 65-100. John Wiley and Sons Ltd.

[RYAN04] W. E. RYAN. "An Introduction to LDPC Codes," in CRC Handbook for Coding and Signal Processing for Recording Systems (B. Vasic, ed.) CRC Press, to be published in 2004.

[SCHOREDER04] MOORE, R.W.; JAGATHEESAN, A.; RAJASEKAR, A.; WAN, M. and SCHROEDER, W. Data Grid Management Systems, Proceedings of the 21st IEEE/NASA Conference on Mass Storage Systems and Technologies (MSST) , April 13-16, 2004, College Park, Maryland, USA.

[SMOOT95] SMOOT, Stephen R. and FLASTER, Michael E. Coded Replication: A Space-Efficient Technique for Increasing File Availability – Dpto. Of Electrical Engineering and Computer Sciences – University of California – Berkeley 1995.

[SKYSURVEY05] The Sloan Digital Sky Survey. www.sdss.org. Acesso em 17/07/05.

[TAURION04] TAURION, César Chede. Grid Computing: Um novo paradigma computacional, Rio de Janeiro. Editora Brasport, 2004.

[WOLFSON97] WOLFSON, Ouri; JAJODIA, Sushil; HUANG, Yixiu. An Adaptive Data Replication Algorithm, University of Illinois and NASA/CESDIS Goddard Space Flight Center - 1997.