Universidade Federal Fluminense

Sandoval da Silva Gonçalves

Reorganização de Objetos Multimídia em Tempo Real quando da Remoção e Inserção de Unidades de Armazenamento do Servidor RIO

NITERÓI

2006

Universidade Federal Fluminense

Sandoval da Silva Gonçalves

Reorganização de Objetos Multimídia em Tempo Real quando da Remoção e Inserção de Unidades de Armazenamento do Servidor RIO

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Computação. Área de concentração: Processamento Paralelo e Distribuído.

Orientador:
Anna Dolejsi Santos

NITERÓI

2006

Reorganização de Objetos Multimídia em Tempo Real quando da Remoção e Inserção de Unidades de Armazenamento do Servidor RIO

Sandoval da Silva Gonçalves

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Computação. Área de concentração: Processamento Paralelo e Distribuído.

Banca Examinadora:

Profa. Anna Dolejsi Santos, D.Sc. - Orientadora
UFF - Universidade Federal Fluminense

Prof. Edmundo Albuquerque de Souza e Silva , Ph.D.
UFRJ - Universidade Federal do Rio de Janeiro

Profa. Rosa Maria Meri Leão, Dr.
UFRJ - Universidade Federal do Rio de Janeiro

Agradecimentos

Agradeço a Deus por toda a força que me foi dada para vencer todas as dificuldades e desafios para concluir este trabalho.

Agradeço a minha orientadora Professora Anna que me deu a oportunidade de participar deste trabalho, que depositou em mim toda confiança, que me ajudou nas correções do texto da tese, e primordialmente colaborou com todos os recursos possíveis para alcançar as metas predeterminadas.

Um agradecimento especial ao Comandante Ricardo Marzullo e ao amigo Sérgio Rodrigues Neves que me orientaram para o ingresso no curso de Pos-Graduação da UFF.

Agradeço aos meus familiares pela paciência, compreensão, pelos momentos ausentes, e pelo sacrifício imposto pela minha dedicação ao curso de pós-graduação.

Um agradecimento especial ao meu pai, que sempre acreditou em minha capacidade e sempre teve orgulho de minha dedicação.

Agradeço especialmente ao Leandro que teve uma contribuição importantíssima no entendimento e confecção das ferramentas criadas ao longo do trabalho.

Agradeço a Vera Helena por toda ajuda, colaboração e paciência que sempre me dedicou nos momentos mais difíceis.

Agradeço a Carlos e Rafael que aceleraram o meu aprendizado com o Sistema Operacional Linux e tornaram possível utilizar todas as ferramentas necessárias ao trabalho.

Agradeço também a Jacques, Daniela, Viviane, Alexandre, Rodrigo, Stênio, Luciana e tantos outros amigos da pós-graduação, que sempre me deram toda força, ajuda e apoio em todas as dificuldades que eu encontrei na confecção dos trabalhos.

Agradeço ao Comandante BentMuller do CASNAV(MB) - Centro de Análise de Sistemas Navais, Marinha do Brasil que proporcionou a oportunidade de fazer o curso de Mestrado na UFF, depositando toda sua confiança no meu esforço prometido para corresponder as expectativas de todos os demais que me recepcionaram tão bem na minha

chegada àquela OM.

Agradeço ao Bernardo a toda ajuda para o entendimento e aos procedimentos corretos para a adequação ao servidor RIO. Agradeço ao professor Edmundo as orientações necessárias para a condução correta deste trabalho.

Agradeço ao chefe do meu setor no CASNAV, Sr. Victor Caruso por todo seu apoio, força e consideração que ajudou a concluir este trabalho. Agradeço também ao Comandante Buarque que durante todo o tempo, me orientou para que eu pudesse corretamente proceder em todo o trabalho de tese e nos momentos mais difíceis esclarecendo e direcionando para as melhores soluções possíveis.

Com certeza, estarei pecando ao não citar tantas outras pessoas que direta ou indiretamente me ajudaram a trilhar este caminho do saber na UFF, para que eu pudesse estar aqui hoje apresentando este trabalho, e a todas peço humildemente muito obrigado.

Resumo

O servidor multimídia distribuído tratado neste trabalho é composto de um Nó Servidor, vários Nós de Armazenamento e vários Nós Clientes. O algoritmo de distribuição dos objetos multimídia no espaço de armazenamento do servidor multimídia, pode variar entre os modelos *Data Striping* e *Random Data Allocation*. Com a finalidade de aumentar o desempenho e prover tolerância a falhas, podem ser acrescentados mecanismos de redundância para armazenamento dos objetos multimídia.

O Servidor Multimídia RIO é um servidor multimídia distribuído que permite acesso simultâneo aos seus Nós de Armazenamento. O algoritmo de distribuição utilizado é o *Random Data Allocation* e implementa o mecanismo de redundância no armazenamento dos blocos dos objetos multimídia, sob a denominação de réplicas.

Uma eventual remoção ou falha de conexão de um Nó de Armazenamento pode causar falhas ou interrupção na exibição dos objetos multimídia nos Nós Clientes, dependendo do número de réplicas definido nos arquivos de configuração do Servidor RIO. Por outro lado, a inserção de Nós de Armazenamento com a finalidade de aumentar o espaço de armazenamento, não causa problemas ao funcionamento do servidor mas não é detectada pelo Servidor RIO. Atualmente a restauração dos serviços do Servidor RIO, em virtude da modificação da configuração dos Nós de Armazenamento exige a parada do servidor, para alterar os registros de configuração, o reinício do servidor para a remoção dos objetos multimídia e posterior reinserção dos objetos multimídia nos Nós de Armazenamento da nova configuração. Somente após completada estas operações é que o fornecimento do objetos multimídia para os clientes pelo Servidor RIO, é liberado, deixando os clientes sem atendimento durante todo esse processo.

A solução adotada neste trabalho para redistribuir os objetos multimídia, quando ocorre uma modificação na configuração dos Nós de Armazenamento é composta por três procedimentos básicos. A interrupção do Servidor RIO, a atualização *off-line* dos arquivos de configuração, e a reinicilização do Servidor RIO em paralelo com uma rotina de redistribuição implementada, que gradativamente redistribuí randomicamente os objetos multimídia no novo cenário de Nós de Armazenamento, ao mesmo tempo que os clientes são atendidos. Foram realizados diversos experimentos e apuradas as quantidades de blocos movimentados durante a operação e o tempo consumido para a completa execução da rotina implementada. Assim sendo, é possível após breve interrupção do Servidor RIO, retomar o fornecimento dos objetos multimídia pelo Servidor RIO aos seus clientes.

Abstract

The treated distributed server multimedia in this work is composed of a Server Node, several Storage Nodes and many Client Nodes. Algorithms for the distribution of multimedia objects in the multimedia server storage space can be divided into two groups: Data Striping and Random Data Allocation models. Multimedia object storage redundancy mechanisms can be added with the purpose of increasing performance and providing fault tolerance.

The RIO Multimedia Server is a distributed multimedia server that allows for simultaneous access to its Storage Nodes. Distribution is achieved by means of a Random Data Allocation algorithm. The RIO Multimedia Server implements a redundancy mechanism for the storage of multimedia object blocks known as replica.

The accidental removal or, for that matter, a connection fault of a Storage Node can cause a failure or the interruption in the exhibition of multimedia objects in the Client Nodes, depending on the number of replicas defined in the RIO Server configuration archives. Conversely, the insertion of a Storage Node with the purpose of increasing the storage space, will not cause problems to the functioning of the server but it will not be detected by the RIO Server. Currently, the recovery of the services provided by the RIO Server, in virtue of the modification of the configuration of a Storage Node, demands that the server should be stopped to modify the configuration registers, then restarted for the removal of multimedia objects and then the reinsertion of multimedia objects multimedia according to the new configuration of the Storage Node. Only after these operations are completed, will the supply of objects multimedia for the customers in the RIO Server be re-established. The customer will be left without attendance during all this process.

The solution adopted in this work for the redistribution of multimedia objects, when an modification in the configuration of a Storage Nodes occurs, is composed for three basic procedures: the interruption of the RIO Server, the off-line update of the configuration archives, and the re-initialization of the RIO Server in parallel with a redistribution procedure that gradually redistributes the multimedia objects randomly in the new Storage Node scenario, at the same time that the clients are taken care of. A number of experiments have been conducted, in which the number of multimedia object blocks moved around during the operation and the time consumed for the complete execution of the implemented routine were evaluated. Therefore, after a brief interruption of the RIO Server, it is possible to re-establish the supply of multimedia objects from the RIO Server to its clients.

Palavras-chave

1. Ensino a Distância
2. Servidor Multimídia RIO
3. A Real-Time Multimedia Object Server
4. Rede GIGA
5. Nós de Armazenamento
6. Manutenção de Nós de Armazenamento em Tempo Real

Glossário

QoS	:	*Quality of Service* (Qualidade de Serviço);
RT	:	*Real Time* (Tempo Real);
NRT	:	*Non Real Time* (Não é de Tempo Real);
RIO	:	*Randomized I/O* (Operações de entrada e saída aleatórias);
RDA	:	*Random Duplicated Assignment*;
RAID	:	*Redundant Arrays of Inexpensive Disks*;
DIVERGE	:	Distribuição de vídeo em larga escala sobre redes Giga com aplicações na educação;
CEDERJ	:	Centro de Educação Superior a Distância do Estado do Rio de Janeiro;
COPPE	:	Coordenação dos Programas de Pós-graduação de Engenharia da UFRJ;
COPPETEC	:	Fundação Coordenação de Projetos, Pesquisas e Estudos Tecnológicos ligada a COPPE;
CBR	:	*Constant Bit Rate* (Taxa Constante de Bits);
VBR	:	*Variable Bit Rate* (Taxa Variável de Bits);
FIFO	:	*First In First Out*;
MTBF	:	*Mean Time Between Failure*;
MTTR	:	*Mean Time To Repair*;
EAD	:	Ensino a Distância;

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Estado da Arte

A Internet possui características que restringem o seu uso à aplicações que tenham rigorosos limites para taxas de erros, perdas de dados e retardo. A qualidade de serviço (*Quality of Service* - QoS) [1] em aplicações multimídia, é obtida através de um conjunto de procedimentos que visam manter a regularidade na transmissão de dados, minimizando através de algoritmos os erros e retardos causados pela rede, permitindo a exibição aceitável dos objetos multimídia no cliente.

Para que as aplicações multimídia possam executar satisfatoriamente, diversas restrições devem ser atendidas para garantir a QoS necessária. Em vista da imprevisibilidade do tráfego na rede, podem ocorrer retardos e perdas de dados em percentuais que oscilam no decorrer do tempo. Esforços vêm sendo feito por pesquisadores no desenvolvimento de técnicas para transmitir com maior velocidade possível a informação [2], compartilhar os recursos de largura de banda que a Internet oferece [3], atenuar o retardo das transmissões [4] reduzir a variação de retardo entre pacotes consecutivos (*jitter*), e recuperar informações nos casos de perda de dados [5] .

Outro fator importante na execução de aplicações multimídia, diz respeito ao armazenamento dos seus conteúdos em dispositivos de armazenamento sujeitos a eventuais falhas de acesso e/ou conexão. Quando ocorrem panes nesses dispositivos, a recuperação e a restauração do fornecimento dos dados é uma questão crítica para a aplicação. Para minimizar este problema, são criados por exemplo, mecanismos de tolerância a falhas, formas de distribuição das informações em diferentes máquinas, e várias cópias da mesma informação gravadas em unidades de armazenamento distintas.

O fator relevante após qualquer interrupção ou falha de acesso do meio de arma-

zenamento, é a restauração do fornecimento de dados para os clientes no menor tempo possível. Assim sendo, a remoção ou falha de conexão de uma unidade de armazenamento num sistema distribuído de informação, pode significar redistribuir a informação no espaço restante de armazenamento, para a restauração do fluxo das informações aos usuários, causando um retardo que compromete a manutenção da qualidade de serviço do servidor multimídia.

O Servidor Multimídia RIO [6] tem seu sistema de armazenamento tal que um dado objeto multimídia está distribuído em diferentes discos, podendo o RIO armazenar simultâneamente diversos tipos de objetos multimídia, controlando ainda as requisições das aplicações concorrentes que podem ter ou não restrição de tempo. Em outras palavras, o sistema de armazenamento do RIO permite manter a vazão dos discos próximo do máximo, sem degradar o fornecimento de dados a múltiplos clientes concorrentes. O esquema de armazenamento se baseia em uma distribuição de dados aleatória (randômica) com a utilização de redundância de dados.

Em [7] Berson et al. apresenta um trabalho relacionando a utilização do modelo RAID em conjunto com a técnica de cluster, sendo que seu objetivo é aumentar a tolerância a falhas do sistema de armazenamento.

A implementação da redundância dinâmica[1], como mecanismo de tolerância a falhas no sistema de armazenamento [8], visa oferecer uma melhor adaptação nos acessos ao conjunto de discos durante a mudança de carga. Neste tipo de abordagem, em geral a redundância é proporcional a popularidade do objeto multimídia.

Com o objetivo de otimizar a transmissão dos objetos multimídia, em [9], é abordada a transmissão de um conjunto de fluxos de vídeos heterogênos [2] provindos de um servidor multimídia remoto através do uso de *proxy* para o atendimento de múltiplos clientes. A intenção é fornecer inicialmente os primeiros blocos do objeto multimídia a um conjunto de clientes, aproveitando-se de uma única transmissão, e transmitindo os blocos restantes gradualmente. O objetivo é reduzir a latência no acesso ao vídeo.

Servidores multimídia são uma alternativa para o ensino a distância. A educação ou ensino a distância (EAD) é o processo de ensino-aprendizagem mediado por tecnologias, direcionado para a educação de grandes massas populacionais, geograficamente dispersas [10]. As tecnologias utilizadas podem ser o correio, o rádio, a televisão, o vídeo, CD-ROM, telefone, fax e a Internet. Os primeiros passos para o ensino a distância [11]

[1] criação de cópias temporária dos vídeos mais solicitados

[2] vídeos heterogêneos - vídeos com diferentes taxas de transmissão

ocorreram por volta de 1850 na Europa e a idéia foi trazida para o Brasil em 1934. Desde então, sua evolução tem acompanhado as mudanças tecnológicas, viabilizando a oferta de conhecimento fora dos grandes centros.

1.2 Motivação da Proposta

O Servidor Multimídia RIO é um servidor distribuído utilizado no ensino a distância [12]. O RIO guarda as informações multimídia, armazenando partes de cada objeto multimídia pelas unidades de armazenamento (discos) distribuídas, visando com isso o balanceamento de carga entre todas as unidades de armazenamento. A inicialização do Servidor RIO utiliza informações de arquivos de configuração que têm a finalidade de determinar, por meio de parâmetros, o perfil de execução do Servidor. A falha de conexão em uma das unidades de armazenamento, pode provocar a perda de dados que estejam sendo transmitido para os clientes, que pode ser atenuado caso haja redundância dos dados.

Durante o funcionamento do RIO, qualquer modificação na quantidade de discos ou falhas de conexão de alguma unidade, não pode ser tratada automaticamente porque o servidor não possui mecanismo para a detecção de falhas de conexão e a perda de acesso a um dos seus discos causa o desbalanceamento de carga no conjunto de discos e consequentemente falhas nas transmissões em curso. Se por outro lado, for adicionada uma unidade de armazenamento ao sistema, o Servidor não tem como utilizá-la de imediato, porque este não foi registrado nos arquivos de configuração, usados na inicialização.

O problema principal para o rebalanceamento das cargas nas unidades de armazenamento é a reconstrução das informações com base na nova configuração, seja pela remoção ou inserção de unidades de armazenamento. Atualmente, esse rebalanceamento demanda um intervalo de tempo que inviabiliza o reinício imediato do Servidor RIO para garantir a QoS oferecida.

O objetivo deste trabalho é definir e implementar procedimentos que permitem restabelecer o funcionamento do RIO tão logo os arquivos de configuração tenham sido atualizados pelo operador humano, fazendo a redistribuição das informações com o Servidor RIO em execução, atendendo as requisições de clientes e garantindo um gradativo balanceamento de carga entre as suas unidades de armazenamento.

1.3 Contribuição da Proposta

Atualmente, quando da remoção ou inserção de uma unidade de armazenamento do RIO, a redistribuição dos objetos multimídia é *off-line*, impedindo o atendimento de clientes por longos períodos de tempo (proporcional ao tempo necessário para tratar toda a base de objetos multimídia armazenada), já que é necessário fazer a remontagem, isto é a reinserção de todos os objetos multimídia existentes.

A implementação do presente trabalho, permite o restabelecimento dos serviços do Servidor RIO, após a intervenção do operador e a atualização dos arquivos de configuração. A redistribuição dos objetos multimídia no novo conjunto de discos do Servidor é feita durante a execução do Servidor RIO, não interferindo no atendimento dos clientes. Para tanto é construído um novo mapa dos objetos, respeitando os padrões de redistribuição randômica dos dados. Durante a redistribuição podem ser exibidos dados de acompanhamento da redistribuição em curso.

1.4 Organização da Proposta

Esta dissertação está organizado da seguinte forma: no Capítulo 2 tratamos de sistema de armazenamento de objetos multimídia; no Capítulo 3 descrevemos as característica do Servidor Multimidia RIO; no Capítulo 4 mostramos a implementação da redistribuição *on-line* dos objetos multimídia nas unidades de armazenamento do RIO; no Capítulo 5 apresentamos a descrição do ambiente de testes e os experimentos realizados; no Capítulo 6 temos as conclusões e as sugestões de trabalhos futuros. Finalmente, no anexo A temos o pseudo-código da rotina implementada e no anexo B temos o código fonte da implementação realizada.

Capítulo 2

Conceitos Básicos

Neste capítulo são descritos conceitos sobre *proxy*, RAID, transmissões *unicast* e *multicast*, vídeo sob demanda, e uma seção que mostra diferentes trabalhos nos quais as técnicas anteriores, otimizam a recuperação de dados no sistema de discos e a sua transmissão na rede.

2.1 Requisitos das Aplicações Multimídia

As aplicações multimídia, devido a sua natureza contínua e com possível interatividade, exigem limites rígidos de tempo na recuperação e entrega dos dados, visto que eventuais perdas e atrasos podem causar interrupções que comprometem a QoS exigida pela aplicação.

Diferentemente das aplicações tradicionais na Internet (Web, transferência de arquivos, correio eletrônico, etc...), as aplicações multimídia, exigem características especiais do servidor tais como: grande capacidade de armazenamento, recuperação rápida de grandes quantidades de dados nos seus discos, e a transmissão eficiente [13] dos objetos multimídia pela rede, de modo a garantir a execução satisfatória da aplicação.

É possível reduzir os requisitos de espaço de armazenamento em disco, bem como a largura de banda necessária para a transmissão das informações com o uso de algoritmos de compressão de imagem e vídeo, como MPEG (*Moving Picture Experts Group*) [14], JPEG(*Joint Photographic Experts Group*) [15], e de compressão de áudio, como GSM (*Global System for Mobile Communication*) [16] e MP3 (MPEG *Audio Layer 3*) [17].

No tocante a transmissão, existem três classes de aplicações multimídia [18]:

- Transmissão de áudio e vídeo armazenado: o contéudo já foi préviamente gravado,

armazenado e codificado no servidor. Uma vez conectado ao servidor, o cliente pode escolher o objeto a ser visualizado;

- Transmissão de áudio e vídeo ao vivo: os dados são capturados através de equipamentos e transmitidos pela rede em tempo real; e
- Transmissão de áudio e vídeo em tempo real: esta modalidade oferece a facilidade de comunicação entre os usuários em tempo real com interatividade. Neste tipo de aplicação o tempo de resposta é mais crítico.

Como a Internet, não fornece QoS diferenciado para nenhum tipo de tráfego, sempre existe a eventual perda de pacotes, retardos na entrega dos pacotes, e a variação de tempo entre a entrega de pacotes consecutivos. Algumas aplicações aceitam perdas ou retardos, já outras como aplicações multimídia são mais exigentes [18], como podemos ver na Tabela 2.1

Tabela 2.1: Aplicações na Internet e seus requisitos de serviços

Aplicação	**Tolerância a perdas**	**Largura de Banda (mínima)**	**Restrições de tempo**
Transferência de arquivo	Não	-	Não
Correio Eletrônico	Não	-	Não
Conteúdo Web	Não	-	Não
Vídeo sob demanda	Sim	Áudio: 1Kbps-1Mbps Vídeo: 10Kbps-6Mbps	Sim segundos
Ensino a Distância	Sim	Áudio: 1Kbps-1Mbps Vídeo: 10Kbps-6Mbps	Sim milisegundos

2.2 *Proxy*

Para tornar a transmissão das informações multimídia mais eficiente e reduzir a necessidade de banda passante no servidor, são usados dispositivos de armazenamento próximos aos clientes. Em geral, *Web caches* (ou servidor *proxy*) [19] são empregadas com essa finalidade. Normalmente, o mesmo *proxy* é utilizado por todos os clientes de uma determinada sub-rede, tornando possível fazer armazenamento de documentos requisitados anteriormente pelos clientes da sub-rede, como pode ser visto na Figura 2.1.

O servidor *proxy* deve ser capaz de agir tanto como um servidor como um cliente. Ele age como um servidor quando aceita requisições de clientes conectados a ele, e age como

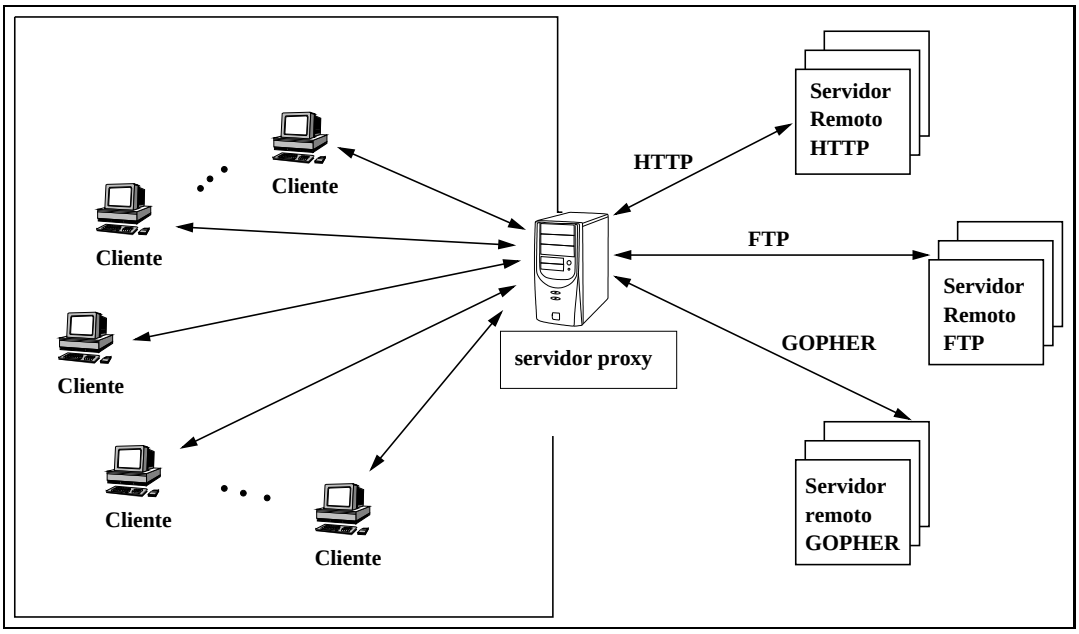

Figura 2.1: Visão geral do *Proxy*

cliente quando se conecta com servidores remotos para conseguir retornar (ou atualizar) as informações para seus clientes. A idéia básica do *cache* é simples e está apresentada na Figura 2.2: armazenar as informações retornadas em um arquivo local para uso posterior de forma que não seja necessário se conectar ao servidor remoto na próxima vez que a informação for requisitada.

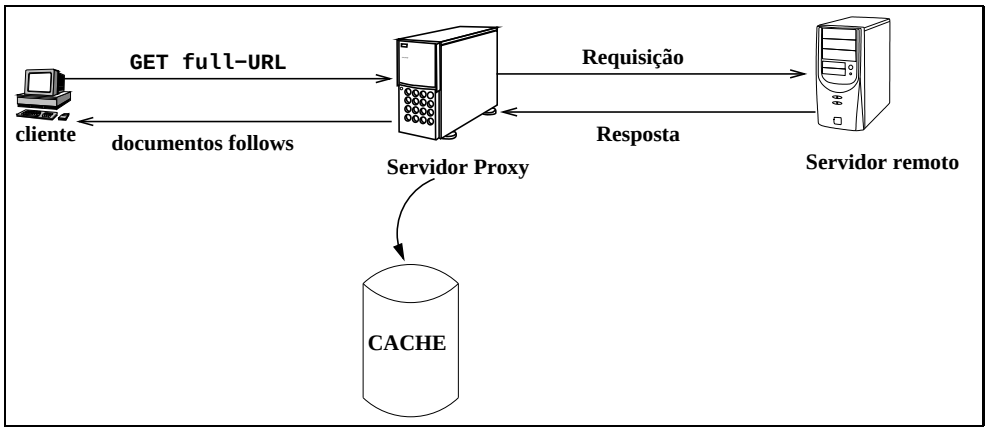

Figura 2.2: Atualização de *cache* em servidor *proxy*

O uso da *cache* no *proxy* reduz a carga no servidor, o tráfego na rede e a latência no acesso ao vídeo pelo cliente. O documento requisitado é obtido do servidor remoto e armazenado localmente para uso futuro. Se uma versão atualizada do documento é achada no *cache* do *proxy* nenhuma conexão ao servidor remoto é necessária, como mostra a Figura 2.3.

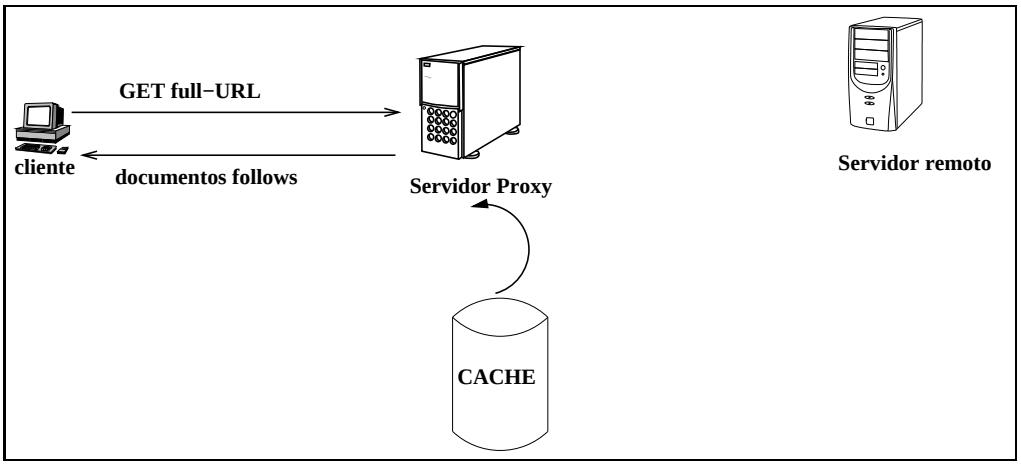

Figura 2.3: Acesso à *cache* em servidor *proxy*

Além da redução de tráfego que o uso do servidor *proxy* proporciona, as aplicações multimídia exigem também dispositivos de armazenamento de alto desempenho para aumentar a velocidade de acesso aos dados, permitindo o atendimento de maior número de clientes. As técnicas de RAID solucionam o problema.

2.3 Discos RAID

Para tornar mais rápida a recuperação da informação armazenada no servidor multimídia, são necessários dispositivos de armazenamento eficientes. O RAID - *Redundant Arrays of Inexpensive Disks* ou *Redundant Arrays of Independent Disks* foi idealizado em 1978 pela IBM[1] [20].

Basicamente, a técnica do RAID é um arranjo redundante de disco físicos visando criar uma unidade virtual em que o espaço de armazenamento e a vazão de saída dessa unidade são, respectivamente, o somatório do espaço de armazenamento e o somatório da vazão de saida de cada disco presente no arranjo. O objetivo do RAID é criar mecanismos de tolerância a falhas e aumentar o desempenho do conjunto, permitindo um melhor balanceamento de carga entre os discos físicos.

O RAID pode ser implementado via *software* ou via *hardware*. No início, somente discos SCSI[2] eram utilizados na tecnologia do RAID. Com a modernização dos discos IDE[3] e ATA[4], foi possível reduzir os custo da implementação do RAID.

[1]IBM - *International Business Machines*
[2]SCSI *Small Computer System Interface*
[3]IDE - *Intelligent (Integrated) Drive Electronics*
[4]ATA - *Advanced Technology Attachment*

Em 1988 foi formalizada a definição dos níveis de RAID de 1 a 5. A identificação do nível do RAID depende de como é feito o arranjo dos discos físicos. Cada nível enfoca como objetivo principal o desempenho, a segurança, ou ambos. Atualmente existem onze níveis de RAID, e na maioria deles é utilizada a segmentação no armazenamento das informações. Os principais níveis de RAID utilizados são:

- RAID nível 0: é composto pela junção de dois ou mais discos físicos para criar um disco virtual (Figura 2.4) cujo espaço de armazenamento é o somatório dos espaços de armazenamento dos discos envolvidos. Existem duas modalidade de implementação do nível 0, a linear e a *striping*. Na implementação linear é criado uma grande partição virtual resultado da junção das partições físicas onde os dados são gravados aonde houver blocos livres disponíveis, sem a preocupação de sua recuperação simultânea em diferentes discos. No *striping*, os dados armazenados são divididos em segmentos e distribuídos entre os discos físicos envolvidos na junção usando a política *round robin*. Nesta política, a cada segmento gravado, o próximo segmento é gravado no próximo disco e após a gravação de um segmento no último disco, o próximo segmento será gravado no primeiro disco do conjunto. A leitura/gravação do primeiro ao último disco constitue um ciclo de acesso. O acesso a um conjunto de blocos de dados pode ser feito através da leitura paralela a diversos discos;

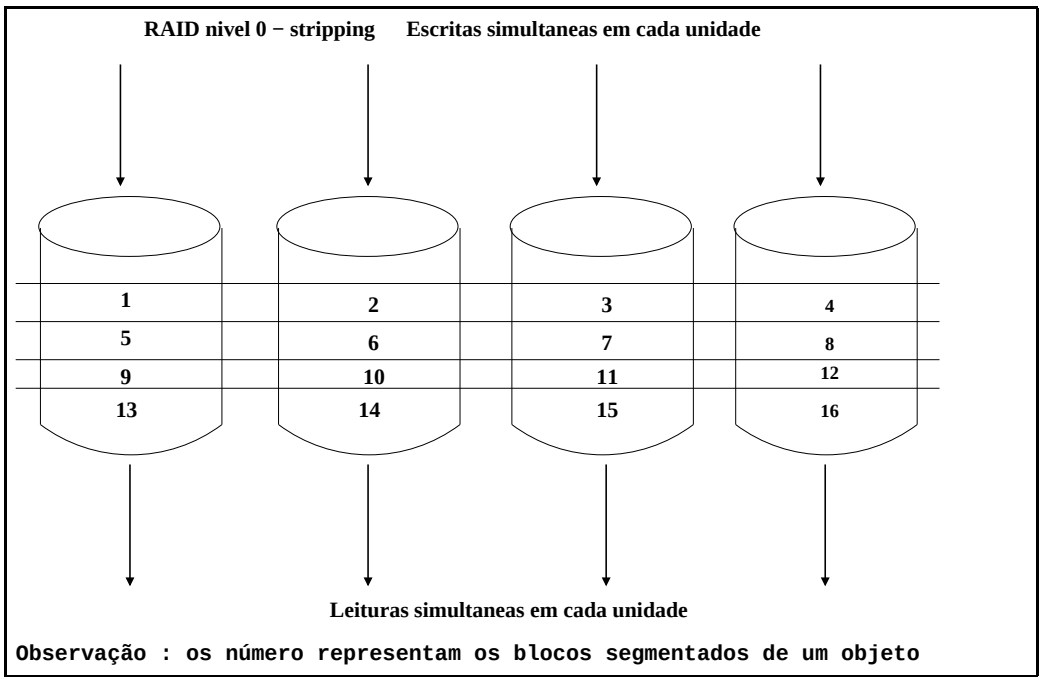

Figura 2.4: RAID nível 0

- RAID nível 1: este nível é implementado através da gravação da mesma informa-

ção em dois discos diferentes do conjunto de discos. Este processo, conhecido como espelhamento, resulta em um mecanismo de tolerância a falhas, como visto na Figura 2.5, para o caso de uma das unidades falhar. Podemos melhorar o mecanismo de tolerância a falhas colocando o disco primário e o disco espelhado em placas controladoras de discos diferentes, eliminando a placa controladora como ponto único de falha;

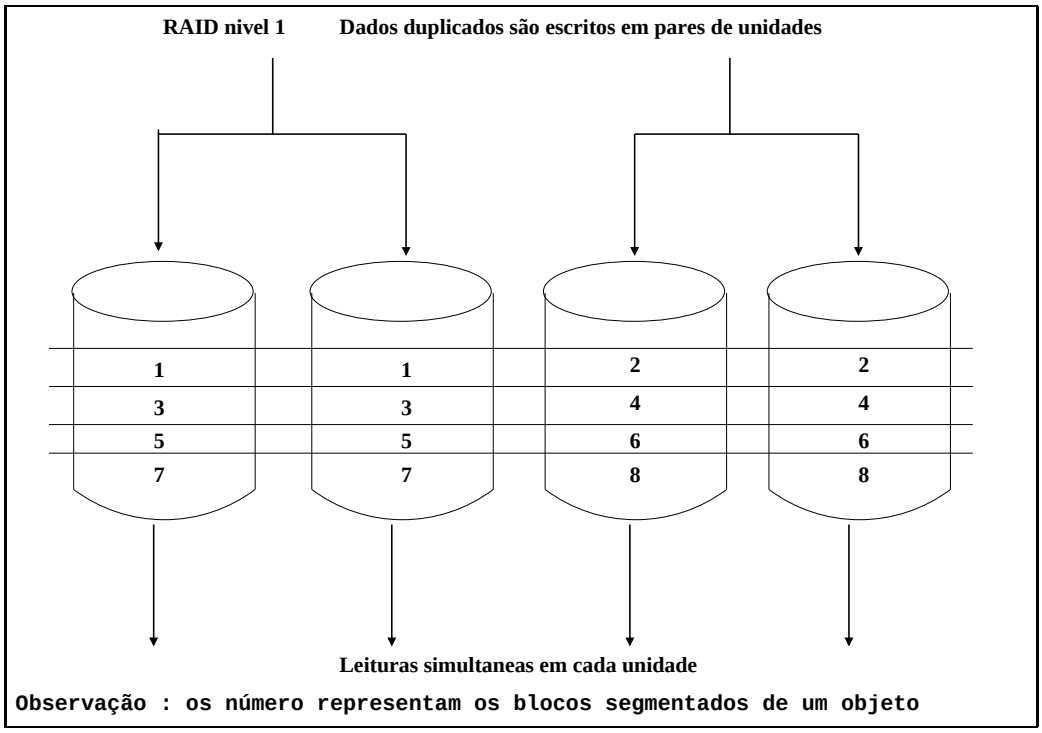

Figura 2.5: RAID nível 1

- RAID nível 3: esta implementação utiliza as informações de paridade para recuperação de informação em caso de falha em uma das unidades. Dado um *array* de n discos, [n-1] discos são utilizados para armazenar dados e um disco é utilizado para armazenar as informações de paridade (Figura 2.6);

- RAID nível 0+1: este é um nível híbrido apresentado na Figura 2.7, que é resultado da combinação do nível 0 com o nível 1. São necessários, no mínimo, quatro discos. Dois discos formam o RAID nível 0 (*striping*) e os outros dois discos formam o RAID nível 1 (*mirroring* ou espelhamento). Esta implementação apesar de mais cara é a que confere simultaneamente alta segurança e alto desempenho ao conjunto.

Existem ainda variações [7] de implementações de RAID que tentam aumentar a tolerância a falhas. Foram criados novos níveis de RAID ao implementar o esquema

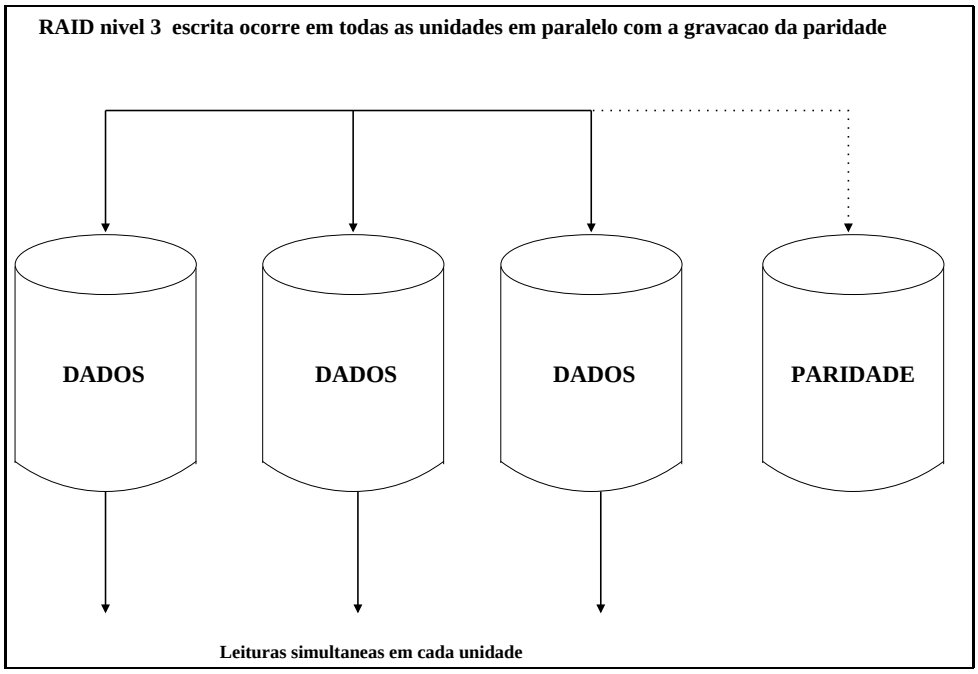

Figura 2.6: RAID nível 3

de paridade presente no nível 3 em outros níveis. Ao invés de um conjunto de discos acompanhado de um disco de paridade, são usados vários agrupamentos de discos e em cada agrupamento há um disco de paridade, formando assim um cluster de disco com paridade. Cada agrupamento ou cluster funciona como um sub-conjunto de RAID nível 3. Se um agrupamento falhar, ainda temos o sistema atendendo parte das requisições dos clientes. Se for aplicado redundância sobre os blocos em diferentes clusters, o sistema tem a robustez necessária para enfrentar falhas em algum cluster.

A despeito das vantagens do RAID, na literatura observamos que o desempenho dos processadores e das memórias tem superado a evolução dos discos e que o gargalo dos sistemas continua sendo a largura de banda dos discos [21]. Considerando que MTTF (*Mean Time Between Failures*) seja o tempo médio entre falhas consectivas de um disco, o MTTF de um conjunto de discos é igual ao MTTF de um disco dividido pelo número de discos do *array* de discos. Isto significa que aumentar o número de disco diminui a confiabilidade do conjunto em relação a um único disco. Entretanto o MTTR (*Mean Time To Repair*), o tempo médio para reparo, diminui quando possuimos discos extras no sistema.

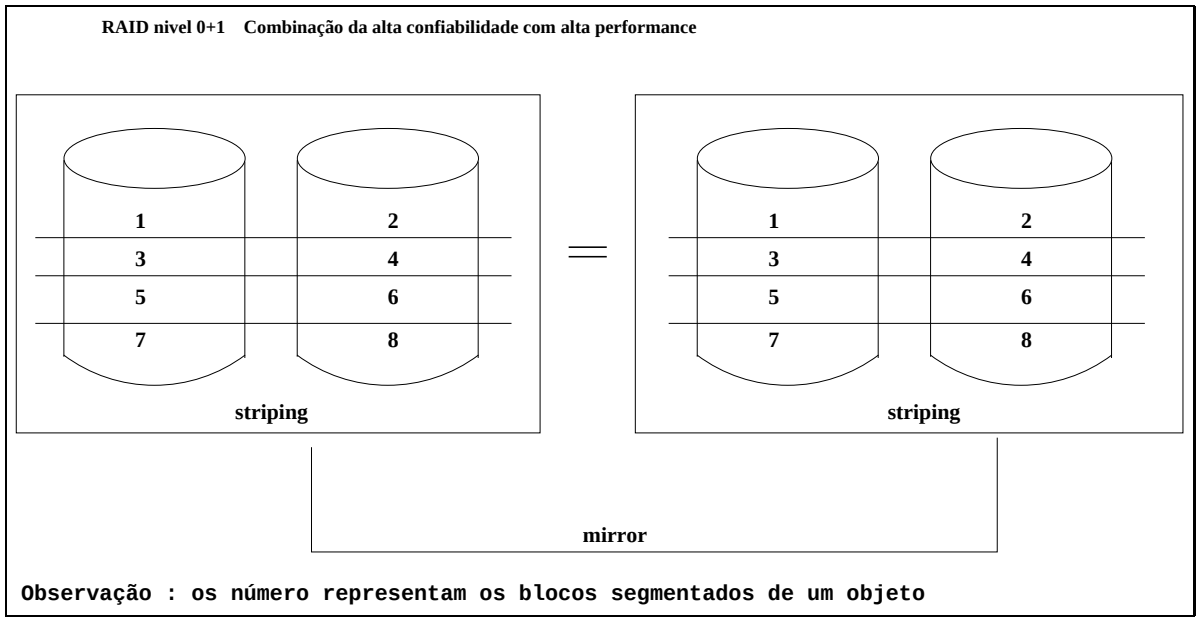

Figura 2.7: RAID nível 0+1

2.4 Transmissões *Unicast* e *Multicast*

Para que aplicações multimídia possam ser exibidas satisfatoriamente, é necessário também a otimização do uso dos recursos de transmissão na rede. Existem duas técnicas de nosso interesse usadas na transmissão de informação em redes de computadores: a transmissão *unicast* [22] e a transmissão *multicast* [23].

Na Internet prevalece a transmissão *unicast* ou uniponto, isto é, um servidor envia pacotes de dados para um determinado cliente. Porém existem situações em que um determinado conjunto de clientes, requisitam o mesmo objeto. Isto resulta em repetidas transmissões pelo servidor, de pacotes com mesmo conteúdo, sobrecarregando desnecessáriamente a banda de transmissão.

A solução para o problema da transmissão pelo servidor de múltiplos pacotes com a mesma informação para vários destinos, é a transmissão *multicast* ou multiponto. Esta modalidade é adequada para aplicações de um-para-muitos, onde um servidor envia um único pacote de dados para muitos clientes, e de muitos-para-muitos, onde cada participante do grupo *multicast*, pode enviar um pacote para todos os demais. O conceito de "grupo *multicast*", é definido como um conjunto de processos que compartilham uma mesma transmissão *multicast*. A Internet já possui um intervalo de endereços reservados para identificação de grupos *multicast* como mostra a Figura 2.8. Assim, todo pacote de dados direcionado a um endereço IP que começe com os bits 1110 configura uma transmissão *multicast* e o restante dos 28 bits identificam o grupo *multicast* para o qual o pacote

está sendo enviado.

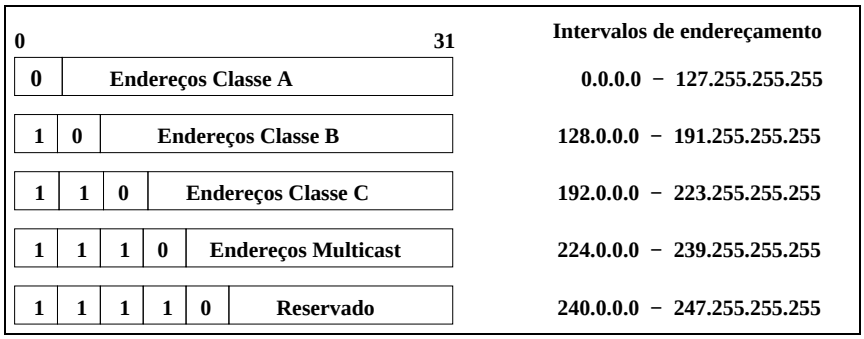

Figura 2.8: Endereçamento *multicast*

Em uma transmissão *multicast* de um-para-muitos, o servidor envia uma cópia do pacote de dados pelo canal para o endereço do grupo *multicast*. O roteador habilitado *multicast* de destino replica o pacote para cada um dos seus *links* de saída através dos quais clientes que fazem parte do grupo podem ser alcançados.

2.5 Recuperação e Armazenamento de Dados no Sistema de Discos do Servidor e na sua Transmissão

Através do uso do servidor *proxy*, das técnicas de RAID, da utilização das transmissões *multicast* e das técnicas que otimizam o vídeo sob demanda, foram desenvolvidos mecanismo, que combinado, oferecem soluções para aumentar o desempenho de acesso aos dispositivos de armazenamento melhorando ainda a sua confiabilidade.

A forma como são organizados os objetos multimídia no *array* de discos afeta o desempenho do servidor. As formas de organização são:

- Em um único disco: são armazenados todos os blocos de um objeto multimídia em um único disco. Esta modalidade causa desbalanceamento no conjunto de discos, e reduz o desempenho no atendimento das requisições dos clientes;

- Distribuído entre vários discos: esta técnica, denominada *Data Striping* [24], permite distribuír os blocos de dados do objeto multimídia de mesmo tamanho, denominados *stripe units* pelo conjunto de discos do servidor, utilizando a política de *round robin*. Dessa maneira, cada requisição de cliente é atendida por um disco diferente do servidor, compartilhando a largura de banda de todos os discos do sistema. Em função desse "espalhamento" da requisição, dizemos que o objeto multimídia a ser

armazenado está desvinculado dos discos, e denominamos este efeito de objeto desvinculado. O objeto desvinculado provê um comportamento de carga balanceada que permite o atendimento a um grande número de requisições de dados e uma melhor utilização da largura de banda dos discos. A Figura 2.9 mostra um esquema do *Data Striping*;

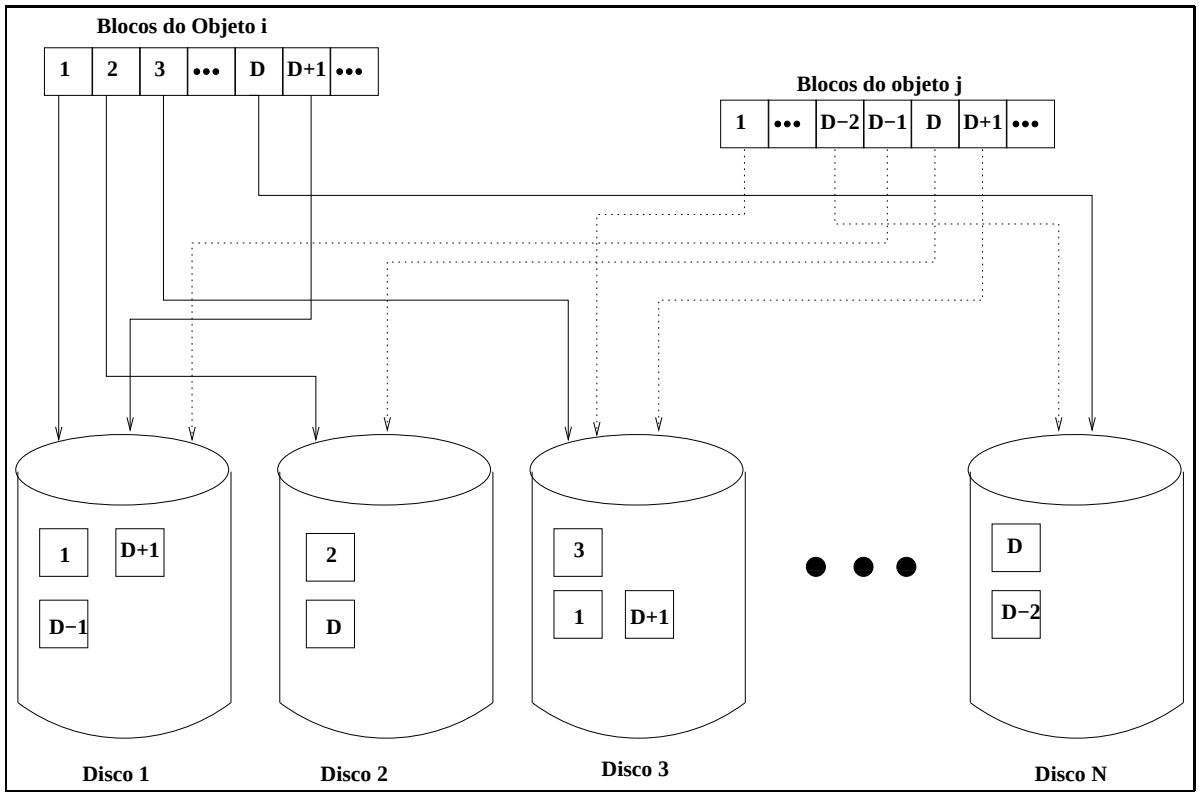

Figura 2.9: *Data Striping*

- Distribuição Randômica [25] permite distribuir os blocos de mesmo tamanho em um disco escolhido randomicamente de um conjunto de discos e, uma vez escolhido o disco, o bloco é gravado em uma posição também randomicamente escolhida. Como a alocação randômica pode produzir flutuações de cargas nos discos, resultando num desbalanceamento de carga durante alguns intervalos de tempo, é utilizada a técnica de replicação de blocos do objeto, garantindo que a réplica do objeto multimídia sempre seja gravada em um disco diferente da cópia original do bloco. Dependendo do número de réplicas, além de minimizar as flutações de carga, aumentando o desempenho do servidor, a confiabilidade também é melhorada. Na Figura 2.10 é apresentado um esquema desta técnica de armazenamento.

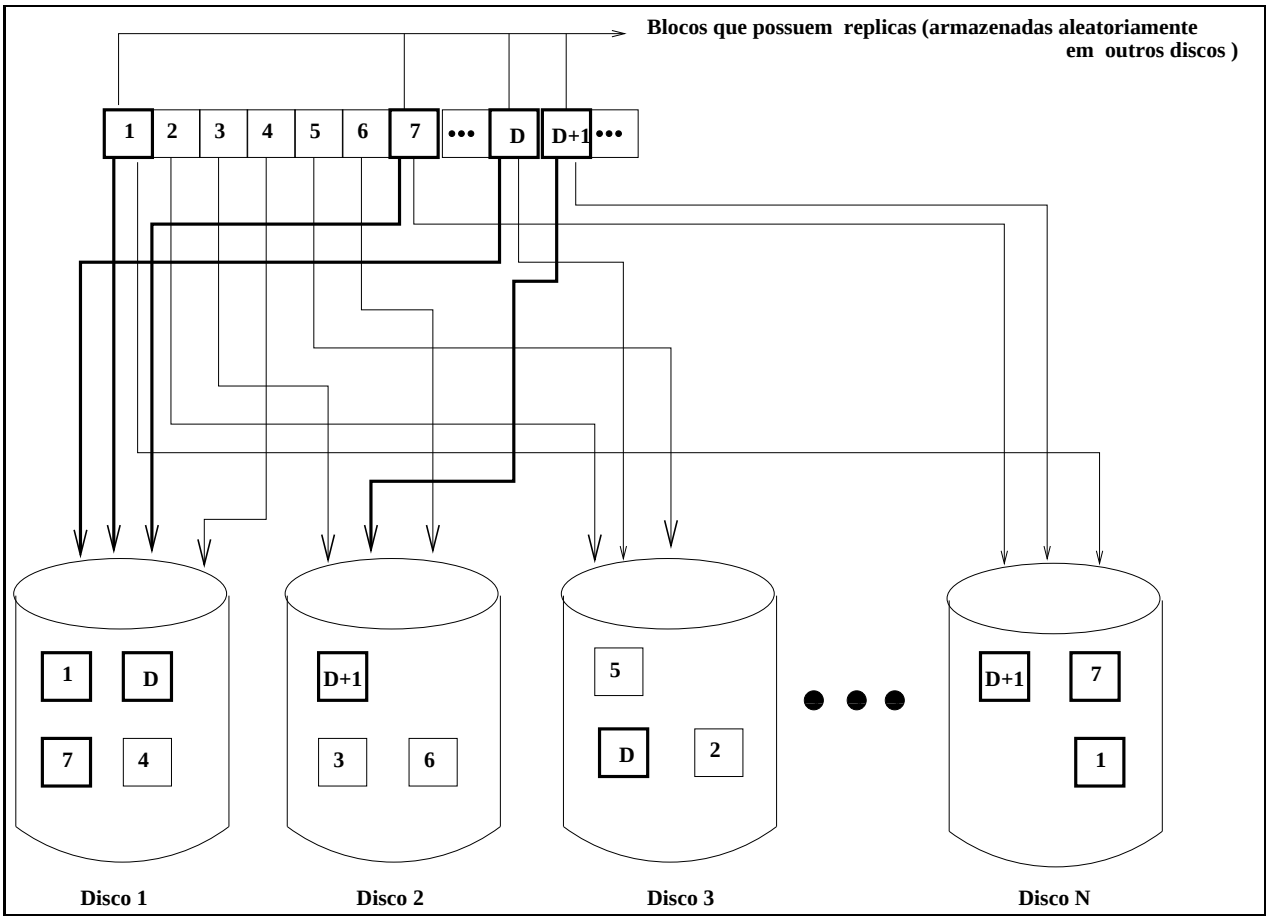

Figura 2.10: *Random Data Allocation*

2.5.1 Recuperação de Dados do Sistema de Discos

Foi observado em [24] que no modelo *Data Striping* existem dois fatores que influenciam o desempenho: o *stripe unit size* ou tamanho do bloco que corresponde ao máximo de quantidade contínua de dados acessados num mesmo disco, e o *degree of striping* ou grau de segmentação que corresponde ao número de discos através do qual é distribuído o objeto multimídia num *array* de discos.

O tamanho do bloco influência o tempo de resposta do disco. Assim, o objetivo principal é encontrar o tamanho do bloco que minimize o tempo de resposta do disco. À medida que o tamanho do bloco aumenta, o número de acessos a disco diminui, para o atendimento de um dado cliente. Por outro lado, quando a carga é máxima, aumentando o tamanho do bloco e considerando que várias solicitações de leitura estão pendentes, o tempo em que a última dessas solicitações é atendida é aumentada proporcionalmente ao tamanho do bloco. Porém se o tamanho dos blocos é muito pequeno, aumenta-se o número de leituras para atendimento de um cliente, isto é, um maior número de blocos devem

ser lidos para o mesmo cliente. A conseqüência do tamanho do bloco no desempenho do *array* de discos, e tal que, se o tamanho do bloco é reduzido, e considerando N o número de discos (grau de segmentação), um mesmo disco sofrerá um maior número de acessos para atender um dado cliente, do que quando o tamanho do bloco é maior.

Em outro estudo [8], observamos que a replicação de blocos pode ser usado como mecanismo de tolerância à falhas na distribuição dos dados pelos discos do *array*. Porém, a replicação estática não oferece flexibilidade para adaptação na mudança de carga do conjunto. A proposta sugerida é que, dependendo da popularidade de um determinado objeto multimídia, sejam replicados dinamicamente alguns blocos para uma melhor distribuição da carga atendendo assim a demanda do momento.

O RDA (*Random Duplicated Assign*) [26] é sugerido como alternativa ao *striping* para servidores de vídeo. O esquema do RDA utiliza o método de dividir o objeto multimídia em blocos e distribuí-los randomicamente pelos discos do *array*. Réplicas dos blocos são distribuídas randomicamente pelos discos. O trabalho mostra que usando *striping* o custo financeiro por usuário chega a triplicar quando o número de 200 usuários é ultrapassado. Utilizando o RDA, além de maior confiabilidade em função da replicação dos blocos, o custo financeiro por usuário cresce lentamente até atingir 400 usuários e a partir de daí o custo financeiro por usuário mantém-se constante. O esquema do RDA, exige sincronismo no acesso aos discos do *array*.

Estudos comprovam que existe uma correlação entre o tempo de requisição do cliente e o padrão de carga dos discos no modelo *Data Striping*, em virtude da previsibilidade dos próximos blocos a serem solicitados. A utilização do *layout* randômico combinado ao RAID com replicação, [27] contudo é comprovadamente superior ao *striping* no que tange a vazão de dados do *array* de discos. Isso aponta para a utilização do modo randômico no intuito de obter melhor desempenho e maior confiabilidade. A aplicação da redundância é condicionada a popularidade do vídeo e ao tamanho das filas de requisições nos discos do *array*.

Como resultado do projeto do Servidor Multimídia RIO [28], foi demonstrado que o uso de replicação dos objetos com alocação randômica em um cluster de computadores, garante suporte a todos os tipos de mídias e escalabilidade com melhor desempenho e segurança do que o modelo *striping*. Estudos realizados nos laboratório da UCLA *Multimedia* com as primeiras implementações e simulações mostram um esquema dinâmico de balanceamento de carga suportando centenas de clientes concorrentes com limite de retardo na ordem de 0,5 segundos com utilização do sistema de discos próximo do 99%.

Segundo [29], a alocação *Data Striping* é eficiente quando a carga de trabalho é altamente previsível. Na prática, vários fatores prejudicam esta previsibilidade, mesmo para os vídeos. Em função das técnicas de compressão de vídeo e para garantir a qualidade da exibição, as codificações podem gerar taxas variáveis de fluxo, gerando uma variação no padrão de I/O do disco. A multiresolução vem complicar mais ainda o *layout* de escalonamento de I/O. A alocação *Data Striping* foi projetada para trabalhar com fluxos CBR e quando se depara com fluxos VBR, o seu desempenho diminui. A sua vantagem é que desacopla a alocação de espaço em disco do uso da largura de banda do conjunto dos discos, evitando dessa forma o desbalanceamento de carga a diferentes discos, pela variação de popularidade dos vídeos. Contudo para garantir a operação em tempo real a duração do ciclo de acesso é considerada a de pior caso e o escalonamento do discos está baseado na sincronização de todos os discos. Isto causa uma redução do desempenho ao final de cada ciclo.

Um importante detalhe a observar, é que se tivermos que acomodar objetos em um *array* de discos com diferentes requisições de largura de banda, é possível variar o tamanho da unidade de armazenamento do disco. Dessa maneira, se tivermos dois objetos que foram subdivididos em blocos de tamanhos diferentes, podemos definir unidades de armazenamentos adequadas para cada um desses blocos. Porém para evitar a fragmentação, o disco deve ter todas as unidades de armazenamento com o mesmo tamanho.

Em função da dificuldade de se caracterizar o padrão de acesso aos discos do *array*, a alocação randômica com replicação é uma alternativa. É selecionado de modo randômico um dos discos e escolhida randomicamente a localização nesse disco onde o bloco será gravado. Com a alocação randômica, flutuações no padrão de acesso podem produzir curtos desbalanceamentos de carga e elevar a latência a valores não permitidos. Como a alocação randômica é aleatória, em determinados momentos um disco pode ser escolhido mais vezes do que outros para a gravação dos blocos, e isto pode gerar mais carga neste disco do que nos outros discos do *array*. A solução desse problema foi feita através da replicação de uma fração dos blocos de dados, que são também alocados randomicamente. Isto confere flexibilidade ao sistema na seleção dos discos quando da recuperação de blocos, melhorando o balanceamento de carga, entre os vários discos do *array*.

Assim sendo, a alocação randômica provê as mesmas vantagens da alocação *Data Striping*, sem impor as limitações citadas anteriormente. Com 25% de replicação, a alocação randômica já tem um desempenho superior a *striping*. Foi também comprovada uma latência no acesso a disco menor, em todos os percentuais de replicação em relação ao

striping. A replicação contudo vai exigir mais espaço para armazenamento e conseqüentemente mais investimento, atenuado pela evolução tecnológica, permitindo melhor escalabilidade com o uso do modelo randômico. Uma outra característica importante no *layout* randômico é a ausência de fragmentação [30] do espaço em disco.

A necessidade de aumentar o número de discos de um *array* de um servidor multimídia, pode criar uma situação em que teremos um ambiente heterogêneo de discos [31], visto que com a evolução tecnológica, ao longo do tempo, temos discos com maior velocidade e maior capacidade.

A adição de discos diferentes no *array* do servidor introduz assimetrias no conjunto que, num primeiro instante, pode causar um desbalanceamento de carga reduzindo a efetiva vazão do *array* de discos. Porém, em [31] mostra que o uso da replicação atenua o limite de retardo do conjunto. Com o uso de 100% de replicação, o desbalanceamento causado pela heterogeneidade é atenuado de modo que, o retardo se distribui entre os discos e o sistema se adapta as diferentes larguras de banda dos discos, alcançando um retardo uniforme para todo o conjunto.

Em instalações reais existem outros fatores importantes a serem consideradas na operação de um servidor multimídia como por exemplo a preparação de novos fluxos de vídeo, isto é a bufferização de blocos recuperados no (array) de discos. Se o fluxo a ser preparado, não estiver sendo solicitado, sua preparação será em *background*, nesse caso o custo é similar tanto para *striping* quanto para o randômico. Mas se a preparação é para atendimento de um fluxo hora solicitado, o modelo randômico consegue gerenciar melhor sua preparação e transmissão, lidando de modo mais eficiente com taxas diferentes de consumo do objeto. Essa técnica combinada ao dimensionamento do tamanho do *playout buffer* no cliente (que pode ser ajustado dinamicamente durante a sessão ou pré-fixado no ínicio da sessão), pode resolver problemas de retardo fim-a-fim da rede e do *jitter* [29].

Capítulo 3

Servidor Multimídia RIO

Considerando que o objetivo deste trabalho é definir e implementar procedimentos que permitem restabelecer o funcionamento do RIO tão logo os seus arquivos de configuração tenham sido atualizados pelo operador, fazendo a redistribuição das informações com o Servidor RIO em execução, neste capítulo são descritas as características do Servidor RIO e dos seus componentes, a comunicação entre os componentes, o seu esquema de armazenamento e as restrições para uma detecção de mudança de configuração e redistribuição automática dos objetos multimídia em tempo real no Servidor RIO.

3.1 Descrição do Servidor Multimídia RIO

Servidores multimídia são servidores que armazenam e distribuem conteúdo multimídia e controlam as solicitações de clientes que consomem estes conteúdos. Conteúdo multimídia, também designado como objeto multimídia, é composto de vídeos, textos, fotografias, músicas entre outros. Em geral, este servidores possuem um ou mais processadores e administram vários discos, como visto na Figura 3.1.

O Servidor Multimídia RIO (*Randomized I/O*) [6], foi projetado e implementado inicialmente na Universidade da Califórnia - UCLA em 1998. É um servidor com um sistema de armazenamento paralelo, isto é, distribui e acessa dados de múltiplos discos simultaneamente, baseado em alocação randômica e replicação de blocos, suportando diversos tipos de objetos multimídia, gerenciando também múltiplas aplicações concorrentes de clientes, com ou sem restrição de tempo.

Posteriormente o RIO foi modificado, sendo atualmente usado em um projeto de pesquisa que envolve as seguintes universidades: Universidade Federal do Rio de Janeiro (UFRJ), Universidade Federal Fluminense (UFF), Universidade Federal de Minas Gerais

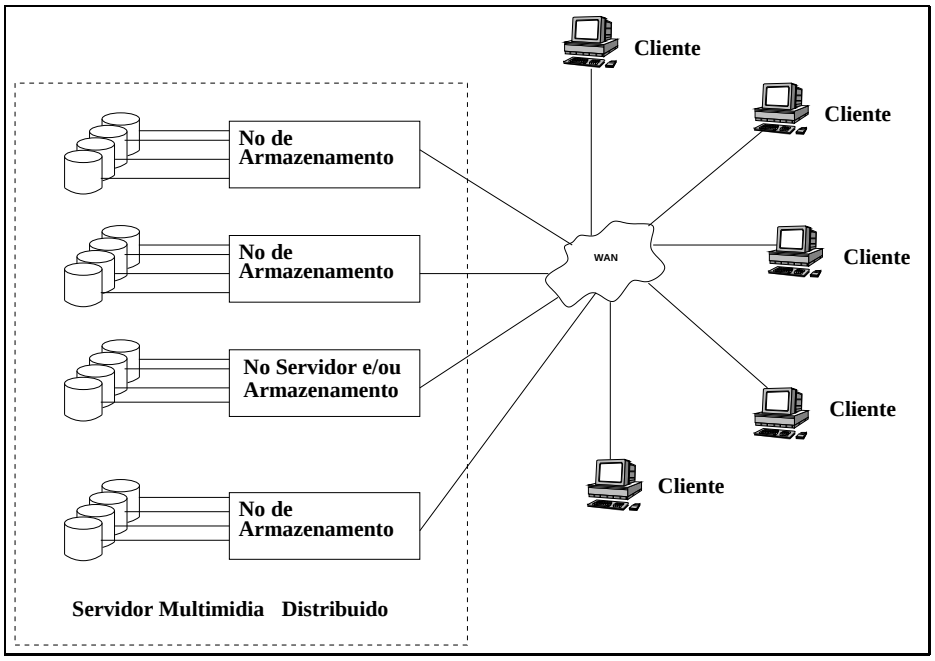

Figura 3.1: Arquitetura de um Sistema Multimídia

(UFMG).

3.2 Características

Dando continuidade a descrição do Servidor RIO, iremos descrever nesta seção as características deste servidor. O Servidor Multimídia RIO está implementado na linguagem C++ e executa sobre o sistema operacional Linux utilizando múltiplas *threads* [32]. Atualmente a distribuição do Linux utilizada é a Mandriva 10.2. Existem servidores instalados no laboratório do LAND na UFRJ, servidores instalados no laboratório da rede GIGA na UFF e nos diversos Pólos do CEDERJ.

O termo *threads* está associado à tarefas que consomem menos recursos computacionais que um processo, e que são utilizadas para execução de processamento paralelo em sistemas multiprocessados. *Threads* são múltiplos caminhos de execução concorrente na memória compartilhada e que executam sobre os mesmos recursos. O padrão para *threads* no Linux é o POSIX[1] 1003.1C.

O Servidor Multimídia RIO tem três componente básicos: Nó Servidor, Nós de Armazenamento e os Nós Clientes, conforme visto na Figura 3.2. Além disso, utiliza vários discos em paralelo para otimizar o acesso ao espaço de armazenamento e largura de banda,

[1]POSIX - *Portable Operating System Interface* com o X significando herança do Unix

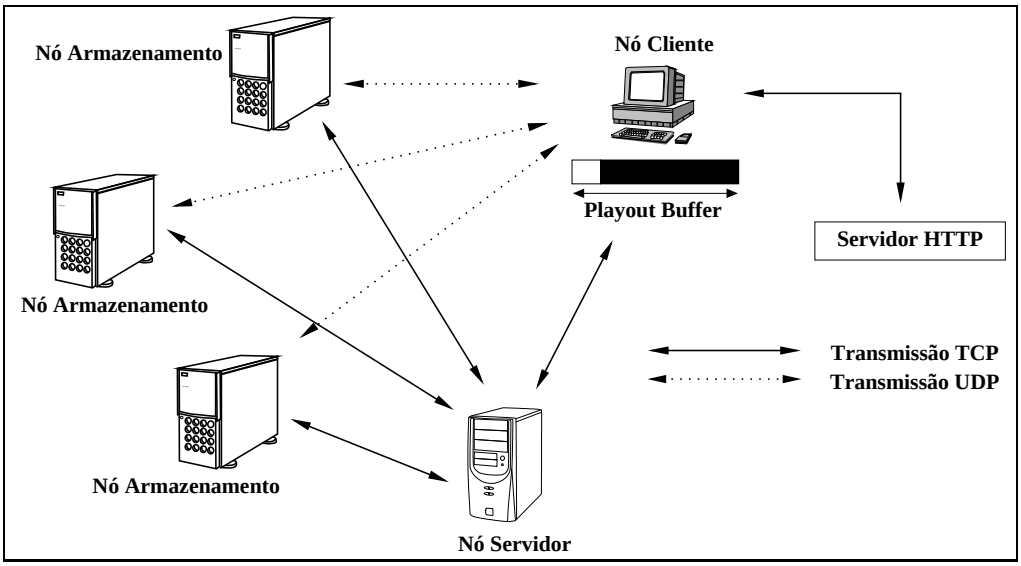

Figura 3.2: Componentes Básicos

proporcionando o atendimento a um número maior de clientes, controla ainda a admissão de novos clientes para evitar comprometer a QoS dos serviços prestados aos clientes como um todo.

O Servidor Multimídia RIO é gerenciado a partir de um único Nó Servidor, que autentica e controla os acessos dos clientes, bem como processa as solicitações de consulta e manutenção dos objetos multimídia.

Os Nós de Armazenamento são dispositivos de armazenamento conectados ao Nó Servidor ou dispositivos de armazenamento remotos conectadas a outras máquinas. Os Nós de Armazenamento mantém em seus discos, o universo de objetos multimídia disponíveis para acesso dos clientes, que são atendidos, após as suas solicitações terem sido validadas pelo Nó Servidor.

Os Nós Clientes permitem que os usuários autorizados pelo Servidor RIO consultem e façam manutenção dos objetos multimídia (gravação e exclusão) nos Nós de Armazenamento. Estas operações são feitas, através de solicitações de abertura de sessões, abertura de fluxos entre outros. As sessões são autenticadas no Nó Servidor, que é responsável por indicar se as mesmas são realizadas em tempo real ou não. As aplicações clientes interagem com o Nó Servidor, solicitando os dados requisitados pelos usuários, armazenando-os em buffers do cliente, repassando-os para um tocador de mídia e finalmente possibilitando ao usuário a sua visualização.

A Figura 3.3 nos dá uma visão da integração dos componentes do Servidor RIO.

Nas próximas seções, são feitas descrições mais detalhadas de cada um dos Nós que

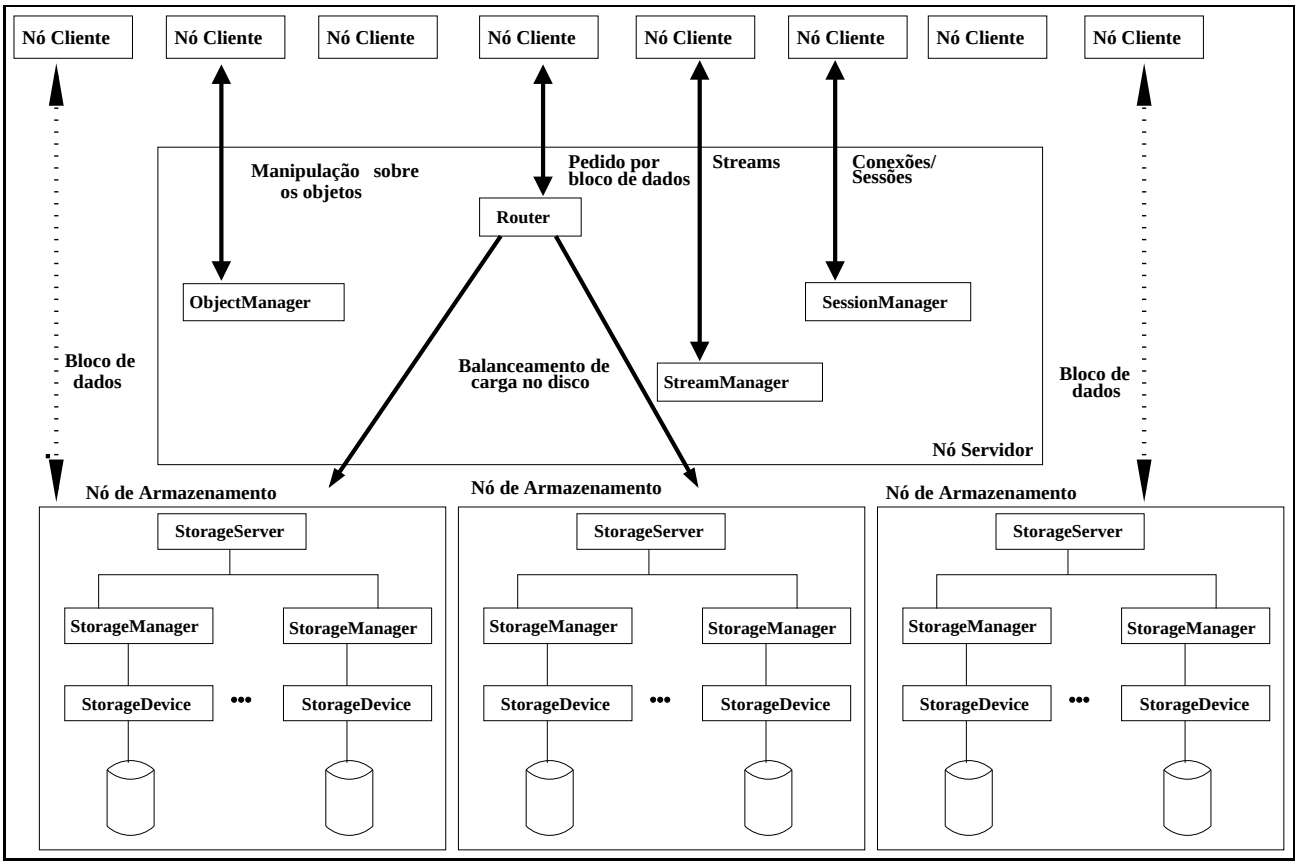

Figura 3.3: Componentes da Arquitetura do Servidor RIO

compõe o Servidor RIO.

3.3 Nó Servidor

No Servidor Multimídia RIO, diretórios, informações de usuários, arquivos de configuração, arquivos metadados dos objetos armazenados compõem o Nó Servidor. Sua implementação baseada numa arquitetura parametrizada, identifica os Nós de Armazenamento presentes, o mapeamento dos discos, o mapeamento dos objetos multimídia, e atributos que determinam o comportamento e desempenho do Servidor RIO.

Todos os tipos de mídia armazenados no Servidor RIO, são chamados de objetos e cada um desses objetos é dividido em blocos de mesmo tamanho e distribuídos randomicamente nos discos dos Nós de Armazenamento. Para cada objeto armazenado nos Nós de Armazenamento, existe um arquivo correspondente denominado metadado, que registra informações de seus atributos e do mapeamento dos seus blocos lógicos nos blocos físicos.

A cada pedido do cliente, o servidor, utilizando as informações contidas no metadado

do objeto solicitado, mapeia os blocos do objeto multimídia e envia cada requisição para o Nó de Armazenamento correspondente, responsável pelo processamento (leitura/escrita de blocos).

O Servidor RIO tem como principais módulos o *SessionManager*, o *ObjectManager*, o *StreamManager* e o *Router*.

O *SessionManager* (Gerenciador de Sessão) é responsável pelo disparo de uma *thread* e pela criação de um novo *socket* para cada novo cliente. Além disso ele controla a troca de informações entre os outros módulos e sincroniza suas execuções.

O *ObjectManager* (Gerenciador de Objetos) é responsável pela manutenção de todos os metadados dos objetos multimídia e fornecimento das informações dos metadados para o *StreamManager*.

O *StreamManager* (Gerenciador de Fluxos) gerencia a abertura de novos fluxos e dos fluxos em execução. As informações do objeto multimídia solicitado pelo cliente e a identificação do cliente solicitante são direcionado para o Roteador de forma a serem atendidas.

O *Router* (Roteador) gerencia os recebimentos de pedidos de leitura/escrita gerados pelo Gerenciador de Fluxos, seleciona os discos para o atendimento dos pedidos, envia os comandos para os Nós de Armazenamento selecionados e recebe as mensagens de controle dos Nós de Armazenamento, como mostra a Figura 3.4.

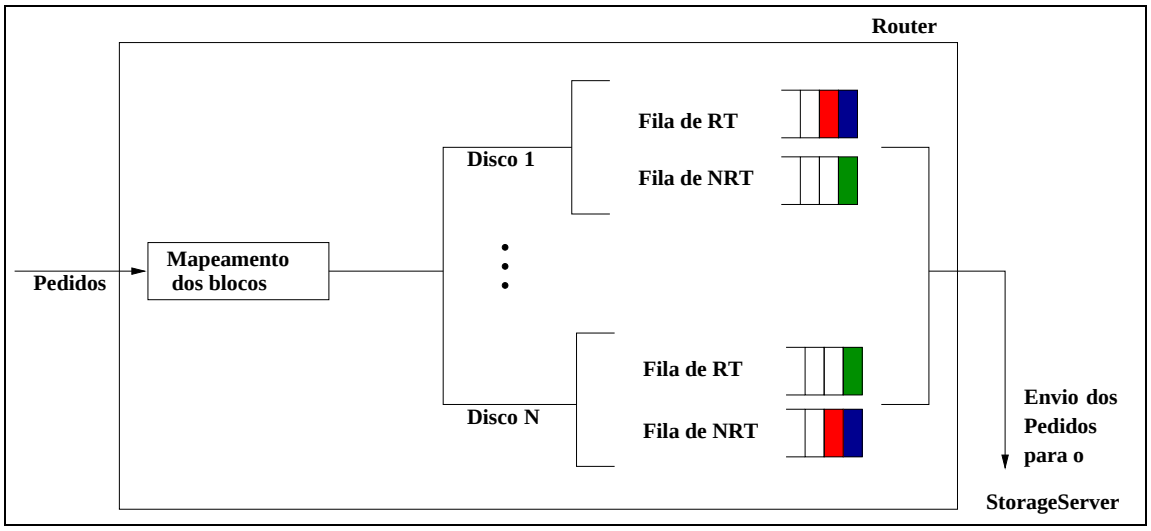

Figura 3.4: Esquema do Roteador do Servidor RIO

.

3.4 Nó de Armazenamento

Para cada Nó de Armazenamento (também denominado *Storage Nodes*) existe um *Storage Server* que gerencia todas as ativadade do Nó. Os componentes do *Storage Server* são : *RouterInterface*, *StorageManager*, *StorageDevice* e *ClientInterface* conforme Figura 3.5.

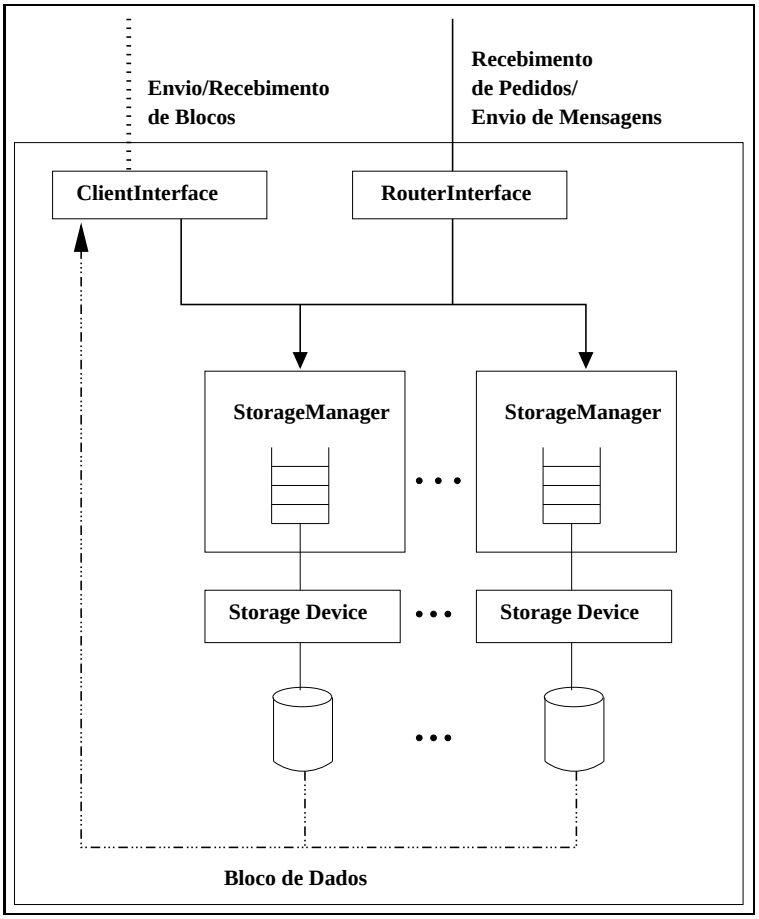

Figura 3.5: *Storage Server*

.O *RouterInteface* (Interface com o Roteador), é responsável pelo recebimento dos pedidos enviados pelo Server para transferência de blocos de dados localizados no Nó de Armazenamento.

O *StorageDevice* (Dispositivo de Armazenamento) de cada disco, gerencia as operações de E/S no respectivo disco, através de chamadas do sistema operacional.

Cada *StorageManager* (Gerenciador de Armazenamento) gerencia um *StorageDevice*, escalonando a fila de pedidos por blocos armazenados neste dispositivo, utilizando o algoritmo de escalonamento FIFO (*First In First Out*). A cada pedido está associado um buffer para o armazenamento do bloco de dados requisitado.

Os pedidos de acesso em cada disco do Servidor RIO são atendidos independentemente

dos seus outros discos, isto é, o acesso aos discos não é sincronizado.

Após a leitura de um bloco pelo *StorageDevice*, o *StorageManager* pede ao *ClientInterface* o envio do bloco de dados para o cliente. Se por outro lado, a operação é de escrita, o *StorageManager* recebe uma solicitação do *ClientInterface* para armazenar o bloco de dados recebido do cliente e o envia para o *StorageDevice* para a realização da escrita no disco.

O *ClientInterface* (Interface com o Cliente) gerencia o envio/recebimento dos blocos de dados a serem transmitidos para os clientes ou armazenados no Nó de Armazemanto a ele associado.

3.5 Nó dos Clientes

Os Nós clientes, são os componentes geradores das solicitações de objetos multimídia, cujo atendimento com o nível de QoS exigida pelas aplicações multimídia, é o principal objetivo do Servidor RIO. Para um processamento mais eficaz dos pedidos dos clientes [33], as solicitações dos clientes são identificadas e classificadas pelo *Server*, de modo que possam ser priorizada as tarefas de tempo real. Os processos clientes disponíveis no Nó Cliente no Servidor RIO são:

- Cliente *riosh*: o *riosh* permite o acesso do usuário a sua conta criada no servidor, através de uma validação de login e senha. O Cliente *riosh* uma vez autorizado pode inserir objetos multimídia, remover objetos multimídia, listar objetos, criar diretórios dentre outras tarefas previstas no *riosh*. Ele possui uma interface de linhas de comandos e outra gráfica;

- Cliente *riomtv*: este processo é utilizado para visualização dos vídeos através do aplicativo *mtvp* [34], sem as facilidades dos recursos de avançar, voltar, pausar, etc;

- Cliente *riommclient*: este processo é utilizado para visualização dos vídeos através do aplicativo *mplayer* e possui uma interface gráfica que permite o uso dos recursos de pausar, avançar e voltar, controlando a exibição do vídeo; e

- Cliente *riowin*: processo semelhante ao *riommclient* executando em sistema operacional Windows e utilizando o Windows *Media Player* para visualização dos vídeos.

3.6 Comunicação entre os Componentes do RIO

Para atender a QoS exigida pelas aplicações multimídia, é necessário também definir adequadamente a comunicação entre os componentes do Servidor RIO.

Na camada de transporte da Internet, os protocolos TCP e UDP são usados por aplicações na transmissão de mensagens. O TCP (*Transfer Control Protocol*) é um protocolo que garante a entrega dos pacotes sem perdas e de forma ordenada. Ele é orientado a conexão, visto que estabelece uma conexão antes do início da transmissão dos dados entre dois processos e que encerram ao término da comunicação. O UDP (*User Datagrama Protocol*) não cria conexões entre os processos, não oferece garantia para entrega ordenada dos pacotes e não possui mecanismos que detectem a perda dos pacotes. Por não existir estabelecimento de conexão, nem a retransmissão de informações perdidas, o UDP é mais adequado às aplicações multimídia do que o TCP. Nessas aplicações, perdas eventuais de pacotes podem ser ocultadas dos usuários.

A comunicação entre o Nó Servidor e os Nós Clientes, bem como entre o Nó Servidor e os Nós de Armazenamento é feita através do protocolo TCP, por meio de mensagens que se compõem de pedidos de blocos e informações de controle e autenticação. Já a comunicação entre os Nós de Armazenamento e os Nós Clientes, por onde fluem os dados solicitados, é feita em UDP.

3.7 Armazenamento

O armazenamento rápido e eficiente, bem como a recuperação dos blocos dos objetos multimídia nos Nós de Armazenamento, é uma função importante do Servidor RIO para garantir a QoS no fornecimento dos objetos multimídia aos Nós Clientes. Assim sendo, a escolha de técnicas usadas no seu sistema de armazenamento é fator decisivo para seu desempenho.

O Servidor RIO armazena os objetos multimídia em formato comprimido, dividindo-os em blocos do mesmo tamanho para evitar a fragmentação do espaço de armazenamento e otimizar o uso da vazão do conjunto de discos dos Nós de Armazenamento.

Os objetos multimídia são compostos por quadros onde cada quadro é uma fotografia instantânea dos bits exibidos na tela do computador. Para decodificar os quadros de um fluxo, o cliente tem de armazená-los na memória, o que implica em bufferização. Durante a exibição, os quadros são consumidos a uma taxa constante.

Em função dos requisitos rigorosos nas restrições de tempo, são usadas técnicas apresentadas em [35] para reduzir interrupções na exibição dos vídeos. Estas técnicas atrasam o ínicio da exibição (*playout delay*) do áudio/vídeo, para compensar os possíveis retardos introduzidos pela rede, além de possibilitar a eliminação do *jitter* (variação do atraso na entrega dos pacotes). O objeto multimídia somente começa a ser exibido depois que um buffer existente no cliente chamado *playout buffer* estiver completamente cheio com pacotes recebidos da rede. O dimensionamento do buffer, deve ser tal que não produza uma latência inicial[2] intolerável.

A implementação atual dos Nós de Armazenamento do Servidor RIO requer que todos os Nós tenham o mesmo número de discos com igual capacidade. Porém, através da compilação do código do RIO é possível ao servidor trabalhar com Nós de Armazenamento com quantidades diferentes de discos e com discos com capacidades distintas.

O Servidor Multimídia RIO utiliza a distribuição randômica com replicação. A gravação de um bloco de dados é feita mediante a alocação randômica de um disco do *array* de discos do servidor e uma vez escolhido o disco, é feita a alocação randômica da posição no disco selecionado. Após a gravação de um bloco do objeto multimídia em um Nó de Armazenamento, o próximo bloco do objeto tem que ser gravado em disco de outro Nó.

A alocação randômica usada no RIO resulta no escalonamento assíncrono dos [36] discos que garante melhor desempenho do que a política *round robin* usada no *Data Striping*. Com apenas 25% de replicação dos blocos, a vazão dos discos se aproxima da taxa máxima evidenciando um melhor balanceamento de carga.

Diferentemente do que foi projetado em [6], o Servidor RIO replica sempre todos os blocos de cada objeto multimídia. A gravação da réplica de cada bloco, respeita o mesmo critério da gravação dos blocos originais. A réplica de cada bloco não pode ser gravada em um disco que pertença ao mesmo Nó aonde foi gravado o bloco original. É possível ainda, existir mais de uma réplica para cada objeto multimídia.

O Servidor RIO identifica o número de réplicas através de um arquivo de configuração. Contraditoriamente, se o número de réplicas é igual a 1, isso significa que sómente há a gravação dos blocos originais sem réplicas. Somente quando o número de réplicas é maior do que 1, o RIO tem cópias dos blocos originais. O número de réplicas não pode exceder ao número de Nós presentes na configuração.

O Servidor RIO ao ser iniciado, se baseia nos arquivos de configuração para identificar

[2]intervalo de tempo entre a requisição do cliente e o ínicio da exibição do objeto multimídia

os Nós de Armazenamento presentes, numerar os discos válidos, criar as filas correspondente aos discos válidos, e determinar o número de réplicas que devem ser feitas de cada bloco, usando para isso o algoritmo de distribuição randômica. A gravação dos blocos do objeto é acompanhada da gravação do metadado correspondente, que possui as informações do objeto multimídia e as posições em que os blocos originais e suas réplicas estão armazenados nos discos.

A cada abertura de sessão pelo cliente para exibição de um vídeo, é disparado uma *thread* que gerencia as solicitações deste cliente. Com o objetivo de agilizar a preparação do objeto a ser disponibilizado para o cliente, o Servidor RIO carrega para a memória, através do *Object Manager*, o metadado do objeto requisitado. O Roteador, com base no metadado, direciona a solicitação de blocos para a menor das filas de requisições a disco existente. Uma vez que solicitação do bloco esteja em uma fila, ela é encaminhada pelo Roteador para o Nó de Armazenamento correspondente. O *Storage Server* do Nó de Armazenamento correspondente faz a leitura do bloco e o coloca no seu buffer. Posteriormente, o *Client Interface* do Nó encaminha a informação para o cliente identificado no corpo da solicitação, fracionando um bloco de 128 KB em 90 fragmentos.

3.8 Restrições

Apesar de toda a estrutura do Servidor Rio ter sido definida para atender aos rígidos padrões que as aplicações multimídia exigem, modificações dos seus arquivos de configuração em tempo de execução não foram implementadas na versão atual do servidor.

Na situação atual do Servidor RIO, se o número de réplicas for igual a 1, isto é, não existem cópias de blocos, na eventual falha de conexão de um Nó de Armazenamento, há falhas no fornecimento de informações para o cliente, já que blocos do objeto multimídia armazenados nos discos do Nó não podem ser acessados. Se o número de réplicas é maior do que um, ainda assim podem ocorrer falhas na exibição no cliente, pois o Roteador não sabe que o Nó de Armazenamento está inacessível e continua enviando pedidos de leitura de blocos para os discos do Nó em falha. Essas falhas são atenuadas à medida que o número de réplicas aumenta. Isto ocorre porque, aumentando o número de réplicas, em caso de falha de conexão de um determinado Nó de Armazenamento, o Roteador tem a sua disposição outros Nós onde a réplica pode ser obtida. Como é de se esperar, no caso de inserção de um novo Nó de Armazenamento, a exibição do vídeo no cliente não é afetada.

Contudo, tanto na remoção como na inserção de um Nó, é necessário a parada do Servidor RIO para posterior alteração dos arquivos de configuração. A nova configuração dos Nós de Armazenamento precisa ser atualizada para refletir uma remoção e/ou inserção. Pode ser necessário ainda modificar o número de réplicas de modo a compatibilizar esse número com o novo espaço de armazenamento.

Após a atualização dos arquivos de configuração do RIO para utilizar todos os discos, é necessária a remoção de todos os objetos multimídia, e após esta operação, uma nova inserção de todos esses objetos multimídia, para garantir uma redistribuição randômica no novo cenário de discos do servidor. O problema principal desse procedimento é o tempo necessário para que as operações de remoção e inserção sejam realizadas, deixando o RIO inoperante para clientes, por longo tempo.

Sendo o Servidor RIO um servidor *multithread*, cada cliente é atendido por uma *thread*. Durante o atendimento da requisição para exibição de um vídeo para um cliente, o Servidor carrega em memória o metadado correspondente do objeto solicitado. Como não existem atualmente mecanismos detectores de falhas de Nós de Armazenamento, a falha em um dos Nós causa a falha no acesso de alguns blocos de dados do objeto, visto que o Roteador está se baseando num metadado inconsistente, isto é, o metadado contém informações de blocos armazenados em discos que estão inacessíveis. O reínicio do servidor é necessário para que sejam reconstruídas as filas considerando apenas os discos válidos, isto é, os discos que pertencem aos Nós de Armazenamento que estão conectados ao Servidor RIO, conforme descrito no arquivo de configuração, atualizado pelo operador. Um outra conseqüencia da falha na conexão do Nó de Armazenamento é o *overflow* das filas dos discos, isto é, o crescimento das filas dos discos do Nó removido.

A técnica existente de detecção da falha de um Nó de Armazenamento através do *overflow* da fila de requisições dos seus discos, não é suficiente para a que o servidor RIO possa iniciar um procedimento de redistribuição dos objetos multimídia. É necessário inicialmente a atualização pelo RIO, dos arquivos de configuração para isolar logicamente os discos do Nó de Armazenamento em falha. Feita a atualização, duas possibilidades de implementação seriam viáveis (e em ambas as soluções haveria a perda de alguns blocos de dos objetos nos fluxos ativos):

- Na primeira solução seriam marcados como inválidas as filas para os discos que pertencem ao Nó de Armazenamento que foi removido; e
- Na segunda seriam re-identificados os discos dos Nós de Armazenamento remanes-

cente. Essa nova identificação implica na eliminação de todas as filas existentes e na criação de novas filas que consideriam apenas os discos acessíveis.

No primeiro caso, usar a marcação de filas como inválida, implica em criar um procedimento de crítica no Roteador para confirmar ou não uma determinada escolha de disco para direcionar o pedido do cliente. Desta maneira, a rotina existente para realizar a redistribuição randômica não poderia ser aplicada, porque ela não prevê filas de discos marcadas como inválida, ela simplesmente usa as filas criadas na partida do RIO. Além disso quando da inserção do objeto multimídia, o Servidor RIO, teria sua rotina de escolha randômica afetada, visto que a escolha aleatória teria de ser feita inúmeras vezes, já que se um disco inválido fosse selecionado, a rotina teria de ser ativada novamente.

No segundo caso, recriar as filas re-executando os vários processos responsáveis por esta tarefa com o RIO em funcionamento, implica na inicialização de variáveis globais usadas por outros módulos (afetando a execução do Servidor RIO de modo ainda não conhecido) e na eliminação de todas as filas. Além disso os metadados podem estar apontando blocos em discos não mais acessíveis, e teriam de ser atualizados, isto é, seria necessário corrigir os ponteiros dos blocos para discos que pertençam a Nós de Armazenamento que tenham sido removidos, apontando-os para réplicas armazenadas em Nós ativos. Já que o metadado precisa ser atualizado, o RIO teria de carregar em memória novamente o metadado e, dessa maneira, o roteador perderia o referencial do próximo bloco a ser solicitado para cada objeto multimídia em exibição, antes da detecção da modificação da configuração dos nós. Neste caso a rotina "*rioie*", da forma como foi implementada, poderia realizar a redistribuição, mas teria que se adequar a reinicialização automática do Servidor RIO.

Capítulo 4

Redistribuição de Blocos em Tempo Real no RIO

A rotina de redistribuição de blocos dos objetos multimídia em tempo de execução implementada, visa garantir a rápida restauração dos serviços do Servidor RIO aos clientes, após a mudança da configuração dos Nós de Armazenamento pelo operador. A solução para o problema da remoção e inserção de um Nó de Armazenamento, leva em conta a minimização do tempo de interrupção do fornecimento das atividades do RIO na entrega de objetos multimídia. Os fatores considerados nessa solução devem garantir a consistência dos metadados dos objetos multimídia e a redistribuição gradual dos blocos desses objetos reduzindo assim a probabilidade de falhas no fornecimento de blocos aos clientes.

4.1 Implementação da Rotina de Redistribuição de Blocos

Após a intervenção do operador, para atualização dos arquivos de configuração no que tange a identificação dos Nós de Armazenamento existentes e ao número de réplicas dos blocos, a rotina implementada neste trabalho, faz a redistribuição dos blocos dos objetos multimídia do Servidor RIO. A distribuição é realizada durante a operação normal do RIO, isto é, durante o atendimento dos clientes. A rotina foi denominada "*rioie*", como referência ao RIO e a **i**mportação e **e**xportação dos objetos multimídia dos Nós de Armazenamento.

A rotina "*rioie*" foi implementada na linguagem C++ (gcc version 3.4.3) utilizando o sistema operacional Linux Mandriva 10.2. Sua execução consiste das seguintes etapas:

1. Para cada objeto multimídia:

   1.1. Salva o metadado;

2. Se a operação é de remoção de Nó, então:

   2.1. A rotina aguarda autorização para realizar a atualização dos metadados (a autorização pode ser dada quando o operador tiver certeza que a falha na conexão com o Nó, não é transiente);

   2.2. Se a autorização do operador foi dada, então:

      2.2.1. Para cada objeto multimídia:

         2.2.1.1. Se um metadado indica que um bloco (cópia ou original) se encontra em um Nó de Armazenamento removido, então o metadado é atualizado apontando para o bloco réplica que está gravado em um Nó ativo (visto que o Servidor RIO não atualiza os metadados, caso tenha havido uma modificação na configuração dos Nós de Armazenamento);

3. Feita a atualização dos metadados o RIO passa a atender as solicitações dos clientes e a rotina aguarda autorização do operador para realizar redistribuição randômica dos blocos dos objetoa multimídia para remoção ou para inserção de Nó de Armazenamento;

   3.1. Se a autorização do operador foi dada, então:

      3.1.1. Para cada objeto multimídia:

         3.1.1.1. Torna o objeto multimídia indisponível;

         3.1.1.2. Faz a cópia do objeto para uma área temporária;

         3.1.1.3. Faz a remoção do objeto dos Nós de Armazenamento;

         3.1.1.4. Insere (randomicamente) os blocos do objeto nos Nós de Armazenamento a partir da área temporária e cria seu novo metadado;

         3.1.1.5. Torna disponível o objeto multimídia com o correspondente metadado;

         3.1.1.6. Armazena estatísticas de tempo e de quantidade de blocos movimentados;

4. Ao fim da redistribuição:

   4.1. Atualiza o arquivo de configuração (mantendo apenas o registro dos Nós de Armazenamento ativos);

   4.2. Grava arquivos com os dados estatísticos coletados;

4.3. Apagado os metadados salvos no início da operação.

Além da implementação da "*rioie*" foi necessária a modificação das rotinas do Servidor RIO:

- *ObjectManager* gerenciamento e manutenção dos metadados dos objetos multimídia;
- *ReadConfiguration* que lê a configuração dos Nós nos arquivos de configuração; e
- *RioError.cpp* que trata as mensagens de erro.

Foi necessário ainda modificar as estrutras de dados usadas pelas rotinas *ReadConfiguration RioError.cpp* que são:

- *RioError.h*; e
- *ObjectHeader*.

Assumimos na implementação do "*rioie*", que a operação de redistribuição de objetos multimídia nos Nós de Armazenamento deve ser autorizada pelo operador, pois a redistribuição automática pelo RIO em caso de uma falha de conexão transiente de um dos Nós poderia resultar na eliminação indevida, desse Nó do Servidor RIO. Da mesma forma, a mudança no número de réplicas exige também a intervenção do operador. A rotina de redistribuição randômica "*rioie*" só é executada por usuário privilegiado previamente cadastrado no Servidor RIO (usuário *root*).

No caso da remoção de um Nó de Armazenamento, não é possível realizar a redistribuição randômica dos objetos multimídia nos discos dos Nós de Armazenamento, quando o número de réplicas é igual a um. A redistribuição não é possível porque não há como recuperar cópias dos blocos perdidos em qualquer outro Nó de Armazenamento, uma vez que número de réplicas igual a um (vide Capítulo 3), significa que somente existem blocos originais dos objetos multimídia no *Storage Server*.

Existem dois arquivos de configuração no Servidor RIO e ambos são necessários para a execução da "*rioie*". O primeiro arquivo de configuração, denominado "disk.cfg" (Figura 4.1), identifica para o servidor os Nós de Armazenamento existentes.

O segundo arquivo de configuração denominado "system.cfg" Figura 4.2, contém parâmetros que regulam o funcionamento e influênciam o desempenho do Servidor RIO,

```
local host
node ped3.ic.uff.br
node ped4.ic.uff.br
```

Figura 4.1: Exemplo de disk.cfg

```
# system.cfg -- Server SystemManager configuration file
# TotalRate and NRTReservedRate are in units of Kbits/sec

# Parameters to new cac, server side buffers and collect measures:
# NetworkRate in units of Mbits/sec
# CollectMeasures, UseServerSideBuffers and UseNewCAC must be 'on' or 'off'
# MaxIntervalEmpty in msec, must be >= 0
# NumberOfEmptyTimes must be >= 0
# NumberOfBuffersToEachClient should be between 0 and 10
# because memory capacity: each storage must allocate
# nBuffers = NumberOfBuffersToEachClient + 1 * MaxClients + 10
# Don't forget to update the parameter -b at each Storage to the appropriated
# value, according the NumberOfBuffersToEachClient being used here!

# GenerateLogs <0|1>
# This option controls log generation. Default is 0 (no logs created)

# MaxDisks control not exactly the number of RIO's disks, but what's the
# numeric limit to the disks's numeration. The RIO server gives each storage 4
# numbers to assign to its disks. So he knows, for instance, that disk 5 is
# always with storage 2. A commom error when trying to use the redundance is
# to put 5 storages with one disk each, but let the MaxDisks with the default
# value (10). By doing that the first disk will be disk 1, the second one, 5
# and so on until the fourth with 13 and the fifth with 17. So the server will
# not recognize those as valid disks for replication, even if the Router adds
# them to the Server during startup.

    BlockSize       131072
    MaxDisks        100
    MaxSessions     300
    MaxStreams      300
    MaxOpenObjects      300
    MaxReplications     8
    UseReplications     2
    MaxDataRequests     1500
    MaxPendingRequests 100
    FileRoot        FileRoot
    MetaRoot        MetaRoot
    UserRoot        UserRoot
    TotalRate       100000
    NRTReservedRate     1500
    NumberOfBuffersForEachClient    5
    BurstSizeOfEachClient           7
    NetworkRate                   100
    CollectMeasures               off
    UseServerSideBuffers           on
    UseNewCAC                     off
    EstimatedTimeParameter      0.125
    NumberOfEmptyTimes            765
    MaxIntervalEmpty             2000
    GenerateLogs                    0
```

Figura 4.2: Exemplo de system.cfg

entre eles o número de réplicas providas pela rotina de inserção e remoção de objetos multimídia.

Para a implementação da rotina de redistribuição randômica, foi acrescido no arquivo de configuração "disk.cfg", ao final de cada linha um atributo que identifica o estado do Nó de Armazenamento. Os atributos necessários criados são:

- A - Nó de Armazenamento ativo ;
- R - Nó de Armazenamento removido ;
- I - Nó de Armazenamento inserido.

Desta maneira, quando da inserção ou remoção de um Nó de Armazenamento do RIO, o operador do sistema deve modificar o estado desse Nó de modo a refletir sua situação atual. A Figura 4.3 exemplifica como fica o novo formato do "disk.cfg", mostrado anteriormente na Figura 4.1. Estes estados permitem que a rotina "*rioie*" conheça a situação de cada Nó de Armazenamento, habilitando então o usuário *root* autorizar a atualização dos metadados.

A atualização dos metados é o primeiro passo da execução do "*rioie*". Ele é feito substituindo-se no metadado de cada objeto, o ponteiro dos blocos que apontam para discos de Nós de Armazenamento com indicação de Nó removido, por qualquer ponteiro de uma de suas réplicas (que obrigatoriamente estão gravadas em outros Nós de Armazenamento). Esta atualização causa um desbalanceamento de carga momentâneo, visto que não foi realizado através de alocação randômica, mas permite o acesso aos objetos armazenados no Servidor RIO, durante a redistribuição randômica dos blocos de objetos multimída.

```
local host            A
node ped3.ic.uff.br   R
node ped4.ic.uff.br   I
```

Figura 4.3: Novo disk.cfg

Ao ser modificada a configuração do Nós de Armazenamento do Servidor RIO, a reinicialização deve levar em consideração o estado de cada Nó indicado no arquivo "disk.cfg". Desta maneira, no caso de remoção de um Nó de Armazenamento, a modificação da rotina *ReadConfiguration* impede a identificação de discos e a criação de filas (contendo solicitações de acesso) para discos de Nós que estejam com indicação de removido. Permite

também, no caso da inserção de um Nó, a identificação de seus discos e a criação das respectivas filas.

A Figura 4.4 mostra no destaque a modificação no código fonte do Servidor RIO para tratar o novo layout do arquivo de configuração "disk.cfg".

A execução do RIO utiliza parâmetros que norteiam sua execução. Do mesmo modo a execução da rotina "*rioie*" utiliza parâmetros para executar as ações necessárias indicadas pelo operador. Na Tabela 4.1 a primeira coluna contém o nome da variável usada no código fonte que define a ação a ser executada pela "*rioie*". Na segunda coluna estão indicados os valores atribuídos no código fonte às variáveis da primeira coluna, quando um parâmetro da terceira coluna é fornecido pelo operador, e finalmente na quarta coluna temos a descrição de cada ação implementada no "*rioie*".

Tabela 4.1: Opções das linhas de comando da rotina *rioie*

Variável	**C**	**P**	**Significado**
IE_OPT_INVALID	0		Opção inválida
IE_OPT_HELP	1	-h	Obtém ajuda
IE_OPT_EXECUTE	2	-e	Executa o Export-Import
IE_OPT_CLEAR	3	-c	Limpa a ultima execução
IE_OPT_STATUS	4	-s	Retorna em um arquivo o estado da execução
IE_OPT_LOCK	5	-l	Bloqueia os metafiles
IE_OPT_UNLOCK	6	-u	Desbloqueia os metafiles
IE_OPT_NEWMAP	7	-m	Constrói um novo mapeamento para os discos
IE_OPT_NPEXEC	8	-f	Executa as opções 7 e 2 em seqüência
IE_OPT_VERIFY	9	-v	Verifica se há algo a fazer no mapeamento dos discos
IE_OPT_REALTIME	10	-p	Acompanha em tempo real a execução da tarefa

As ações de "Constrói um novo mapeamento da execução" e "Executa o Export-Import" dos objetos multimídia estão integradas no Servidor RIO e autorizadas pelo operador. As outras opções são executadas em outra console, em paralelo com a execução do RIO, para acompanhar a evolução da redistribuição de blocos.

As mensagens para o usuário emitidas pela rotina "*rioie*" estão descritas na Tabela 4.2, enquanto as mensagens de erro estão descritas na Tabela 4.3. Nas tabelas a primeira coluna contém a identificação usada no código fonte do *string* associado ao texto da segunda coluna. Nas duas tabelas, na segunda coluna, os caracteres %s são preenchidos por parâmetros determinados pela rotina "*rioie*" em tempo de execução.

O script "runserv" Figura 4.5 ativa os Nós de Armazenamento e a execução do Servidor RIO. Esse mesmo script é usado para fazer a sincronização da execução da rotina "*rioie*"

```
int DiskMgr::ReadConfiguration()
{
    int rc = 0;
    char *nodename = 0;
/// Autor : Sandoval  -  05/05/06
/// Inserção de variavel que registrará o status do Nó de armazenamento
    char *nodestate = 0;
    const char CONFIGNAME[] = "disk.cfg";
    cout << endl <<"   Reading DiskMgr configuration " << CONFIGNAME <<" ... ";
    token tok;
//    tok.setcerr(&cout);
    if( tok.openfile( CONFIGNAME ))
        return 1;
    int word;
    while(( word = tok.parseline( mykeys )) >= 0 )
    {
        switch( word )
        {
            case k_disk:
                tok.emsg("not yet implemented");
                break;
            case k_node:
                if( tok.getstrnew( &nodename ))
                {
                    tok.emsg("missing storage node hostname");
                    break;
                }
/// Autor : Sandoval  -  05/05/06
/// leitura do proximo token - tok.next()
/// captura do string que registra a situacao do Nó - tok.getstrnew(&nodestate)
/// se nodestate contiver a letra 'R', significa Nó removido
                tok.next();
                 if( tok.getstrnew( &nodestate ))
                {
                    tok.emsg("missing states node hostname");
                    break;
                }
                 if ((*nodestate) != 'R')
                   rc |= BuildNode(nodename);
//                       rc |= BuildNode(nodename);
                   break;
            default:
                tok.emsg("{default} invalid word");
                continue;
        }
        tok.ckend();
    }
    tok.closefile();
    if( rc == 0 )
    {
        cout << " OK. " << endl;
    }

    // ---------------------------------------------------------------------

    return rc;
}
```

Figura 4.4: Sub-rotina *ReadConfiguration* do Servidor RIO

Tabela 4.2: Mensagens do sistema para o usuário

Identificação	**Significado**
IE_MESSAGE_01	Nenhuma modificação nos discos foi detectada.
IE_MESSAGE_02	A remoção de %s disco(s) foi detectada.
IE_MESSAGE_03	Deve-se efetuar o processo de ajuste dos mapas dos arquivos para a nova configuração dos discos (Sim/Não) ?
IE_MESSAGE_04	Foi detectada a inclusão de novo(s) disco(s) no sistema.
IE_MESSAGE_05	Lembre-se de que sem esta ação, o sistema provavelmente não entrará em operação corretamente !!
IE_MESSAGE_06	Caso a resposta a seguinte pergunta for negativa lembre-se de executá-lo manualmente para que os discos sejam rebalanceados !!!! (comando : %s -e)
IE_MESSAGE_07	Deve-se executar o processo de importação/exportação em seguida ao remapeamento dos meta-arquivos (Sim/Não) ?
IE_MESSAGE_08	Deseja iniciar o processo de rebalanceamento de carga dos arquivos do sistema agora (Sim/Não) ?
IE_MESSAGE_09	%s :
IE_MESSAGE_10	Versão (%s)
IE_MESSAGE_11	Uso: %s < -h|-e|-c|-s arquivo-de-saida|-l|-u-m|-f|-v|-p segundos > Opções:
IE_MESSAGE_12	-h : Apresenta este texto de ajuda e a versão do sistema.
IE_MESSAGE_13	-e : Executa a operação e exportação e importação dos objetos do servidor.
IE_MESSAGE_14	-c : Limpeza dos dados de exportação e importação do servidor.
IE_MESSAGE_15	-s : Gera um relatório no arquivo-de-saída com o estado atual do processo de exportação e importação do servidor.
IE_MESSAGE_16	-l : Efetua um lock lógico nos meta-files do servidor.
IE_MESSAGE_17	-u : Efetua um unlock lógico nos meta-files do servidor.
IE_MESSAGE_18	-m : Atualiza as informações dos meta-files para propiciar a operação do servidor enquanto o processo de importação e exportação dos objetos está em curso.
IE_MESSAGE_19	-f : executa as opções -m e -e em seqüência.
IE_MESSAGE_20	-v : Verifica se ocorreram mudanças no mapa dos discos e inicia as operações de manutenção necessárias através de interação com o operador do sistema.
IE_MESSAGE_21	-p : Acompanha, em tempo real, a execução da tarefa. segundos: opcional intervalo de refresh das informações ([30]<->300).
IE_MESSAGE_22	-t : Para o uso em rotinas de testes apenas.

Tabela 4.3: Mensagens de erro do sistema para o usuário

Identificação	**Significado**
IE_MSG_ERR_01	O arquivo de origem da cópia (%s) não foi encontrado.
IE_MSG_ERR_02	Não foi possível criar o arquivo destino da cópia (%s).
IE_MSG_ERR_03	Um erro foi detectado durante a escrita no arquivo destino (%s)
IE_MSG_ERR_04	O diretório do executável (%s) não foi encontrado. Favor colocá-lo no PATH do usuário.
IE_MSG_ERR_05	O arquivo, %s, que contém os dados para a conexão com o servidor não foi encontrado.
IE_MSG_ERR_06	Não pude criar o arquivo de controle da execução concorrente (%s)
IE_MSG_ERR_07	Não consegui ler o arquivo temporário (%s).
IE_MSG_ERR_08	Não foi possível criar o arquivo temporário para a apresentação dos dados (%s)
IE_MSG_ERR_09	O arquivo de configuração do programa não foi encontrado (%s)
IE_MSG_ERR_10	O meta-arquivo %s não foi encontrado.
IE_MSG_ERR_11	Erro durante a leitura do meta-arquivo %s
IE_MSG_ERR_12	Erro durante a escrita no meta-arquivo %s.
IE_MSG_ERR_13	Favor executar antes a opção: %s -v
IE_MSG_ERR_14	O arquivo de configuração dos discos não foi encontrado (%s)
IE_MSG_ERR_15	Erro durante a leitura do arquivo de configuração do programa (%s).
IE_MSG_ERR_16	Erro durante a leitura do arquivo de configuração do programa.
IE_MSG_ERR_17	Erro durante a escrita no arquivo de configuração do programa.
IE_MSG_ERR_18	O objeto %s não pode ser removido.
IE_MSG_ERR_19	O objeto %s não existe.
IE_MSG_ERR_20	Permissão negada para remover o objeto %s.
IE_MSG_ERR_21	Erro durante o download do objeto.
IE_MSG_ERR_22	Erro durante o upload do objeto.
IE_MSG_ERR_23	A conexão com o servidor na máquina %s falhou.
IE_MSG_ERR_24	Versão inválida.
IE_MSG_ERR_25	O nome do usuário é inválido.
IE_MSG_ERR_26	O servidor não está respondendo.
IE_MSG_ERR_27	O número máximo de sessões foi alcançado.
IE_MSG_ERR_28	Erro inesperado.
IE_MSG_ERR_29	O tamanho do bloco utilizado pelo servidor não pode ser lido.
IE_MSG_ERR_30	Permissão negada para o objeto %s.
IE_MSG_ERR_31	Erro inesperado para o objeto %s.
IE_MSG_ERR_32	Este passo de preparação é necessário para o correto cumprimento desta tarefa.
IE_MSG_ERR_33	Opção inválida para o programa.
IE_MSG_ERR_34	Esta opção deve ser executada de forma exclusiva. Já existe um programa em execução.
IE_MSG_ERR_35	Erro na leitura do arquivo de configuração dos discos.
IE_MSG_ERR_36	Erro na criação do arquivo de configuração dos discos (%s). Favor retornar o backup deste arquivo diskcfg.bak.
IE_MSG_ERR_37	Nenhuma atividade do programa foi detectada.

com a execução do reinício do atendimento de solicitações dos clientes pelo Servidor RIO. Este script vai garantir que a rotina "*rioie*" detecte a mudança de configuração dos Nós de Armazenamento antes do Servidor RIO iniciar sua execução, e em caso afirmativo ative os procedimentos necessários para a atualização dos metadados dos objetos multimídia.

```
export RIOIE_PATH=/home/vod/RIOServer
PATH=$PATH:/home/vod/RIOServer
export PATH
$RIOIE_PATH/rioie -v
./RIOServer 2>RIOServer.log &
```

Figura 4.5: Script runserv

.

4.1.1 Execução da Rotina de Redistribuição dos Blocos

Após o operador do Servidor RIO detectar a falha de conexão em algum Nó de Armazenamento ou ao desejar remover ou inserir um Nó de Armazenamanto, o Servidor RIO deve ser interrompido pelo operador e as alterações adequadas nos arquivos de configuração devem ser realizadas (como foi visto na Secção 4.1).

A rotina "*rioie*" detecta através do arquivo de configuração, que houve modificação na configuração dos Nós de Armazenamento. Em conseqüência desta detecção, o "*rioie*" faz um backup e a atualização de cada metadado de objetos multimídia armazenados no Servidor RIO. Em seguida, o Servidor RIO é reíniciado.

Durante a execução da rotina "*rioie*", os estados de cada um dos metadados são gravados no arquivo "ie_lst.dat". Na Tabela 4.4 temos os diferentes estados que podem ser assumidos pelos metadados dos objetos multimídia durante a redistribuição randômica. A primeira coluna contém o nome da variável usada no código fonte que define estado do metadado. Na segunda coluna estão indicados os valores atribuídos no código fonte às variáveis da primeira coluna, e finalmente na terceira coluna temos a descrição do estado do metadado. É possível ao operador acompanhar a evolução da atualização dos metadados, em tempo real, por meio da exibição do conteúdo do "ie_lst.dat" na console do operador.

No caso de remoção de um Nó de Armazenamento os metadados estão inconsistentes no reinício do Servidor RIO, e por isso a rotina de redistribuição randômica atualiza todos os metadados para que o RIO possa iniciar o atendimento dos clientes. Esta atualização é feita substituindo-se no metadado de cada objeto, os ponteiro para os blocos com indicação

Tabela 4.4: Tabela dos estados possíveis para os objetos multimídia

Identificação	**Valor**	**Significado**
IE_STATE_WAI	0	Remapeamento não iniciado
IE_STATE_MNR	1	Remapeamento não necessário
IE_STATE_MPR	2	Remapeamento necessário
IE_STATE_MAP	3	Remapeamento executado
IE_STATE_DWL	4	Efetuando download
IE_STATE_DWD	5	Download efetuado
IE_STATE_DET	6	Removendo
IE_STATE_DEL	7	Removido
IE_STATE_UPL	8	Efetuando upload
IE_STATE_DON	9	Terminado

de discos que pertençam a Nós que foram removidos, por qualquer ponteiro de uma de suas réplicas, que estão gravadas em outros Nós de Armazenamento. A atualização dos metadados causa um desbalanceamento de carga momentâneo, visto que não foi realizado através de alocação randômica, mas permite o roteamento correto para todos os objetos armazenados no Servidor RIO.

Para que a redistribuição de um objeto multimídia seja realizada, é necessário que seja feito inicialmente antes uma cópia de segurança do objeto. Para tanto o "*rioie*" implementa o processo de exportação do objeto (isto é, o *download*). O *download* faz a cópia do objeto multimídia para uma área de trabalho em disco do Servidor RIO. Em seguida, o "*rioie*" realiza a remoção de todos os blocos do objeto multimídia dos Nós de Armazenamento do Servidor RIO, eliminando também seu correspondente metadado. Finalmente, o "*rioie*", a partir da cópia de segurança realizada pelo *download*, faz a importação do objeto multimídia (isto é, o *upload*), que insere nos Nós de Armazenamento os blocos do objeto multimídia, utilizando para isso o algoritmo de distribuição randômica do Servidor RIO, criando também seu correspondente metadado. A exportação, a importação e a remoção do objeto multimídia utilizam as funções do "*rioshell*". O "*rioshell*" é um módulo do RIO responsável pelas tarefas de inclsão e remoção de objetos dos Nós de Armazenamento.

Ao remover um objeto multimídia a rotina "*rioie*" precisa resolver o problema do conflito no acesso concorrente do metadado, já que podem ocorrer solicitações de clientes requisitando a exibição do objeto multimídia. Para resolver este problema foi criado uma variável na estrutura do *header* do metadado (Figura 4.6) denominada "lock". A rotina "*rioie*" ao acessar um metadado para efetuar a remoção, atribui imediatamente o valor

um à variável "lock" evitando que esse metadado seja lido do disco pelo *ObjectManager*. O *ObjectManager* testa essa variável quando há uma solicitação para exibição de um vídeo, impedindo seu acesso ao metadado no disco, enquanto a redistribuição não estiver completada.

```
struct ObjectHeader
{
    char            Signature[8];
    short           Type;
    time_t          LastWrite;
    int             nRep;
    unsigned long   BlockSize;
    RioBlock        nBlocks;
    RioObjectSize   Size;
    unsigned char   lock;     // variavel usada para lock e unlock dos metafiles
    char            pad[27];  // pad to 64 byte boundary
};
```

Figura 4.6: Estrutura do *Header* do metadado dos objetos multimídia

.O bloqueio do acesso ao metadado de um objeto por meio da variável *lock* não resolve o problema dos objetos que já estão em exibição, já que esse metadado já foi lido anteriormente do disco para memória e está sendo utilizado pelo Roteador. Quando a rotina "*rioie*" inicia a remoção de um objeto em exibição, pode haver a interrupção desse vídeo pois o objeto pode ter seus blocos removidos pelo "*rioie*". Em outras palavras, o metadado lido pelo Roteador pode estar apontando para blocos já removidos.

Para minimizar o problema foi introduzido então um retardo (função *sleep*(nn)) na rotina "*rioie*" antes da operação de remoção para reduzir a possibilidade de interrupção de exibição do vídeo em andamento. O dimensionamento do retardo depende de estudos estatísticos baseados em acompanhamento da execução com massa real em um intervalo de tempo longo o suficiente para que se possa estabelecer um valor médio que minimize ao máximo a possibilidade de interrupção de um vídeo.

É importante notar que a inserção de Nó de Armazenamento, exige a execução prévia do script "*format*" para criação e formatação dos discos do novo Nó de Armazenamento, bem como a alteração do script "*rioserverd*" para inserir a linha com a ativação do novo Nó de Armazenamento. No caso da remoção do Nó de Armazenamento, deve ser retirada a linha correspondente ao Nó de Armazenamento que foi removido, no script "*rioserverd*".

Existem duas diferentes situações onde a redistribuição dos blocos de objetos multimídia no RIO é interrompida: (1) parada da execução do "*rioie*" momentânea, e (2) desistência definitiva do processo de redistribuição em curso (o operador se arrepende de ter autorizado a redistribuição). Assim ações diferentes precisam ser executadas em cada caso:

1. Reativar a rotina e deixar que execute até o fim (é obrigatório não modificar os arquivos de configuração do Servidor RIO usados pela rotina "*rioie*" antes da interrupção);

2. Restaurar os arquivos de configuração do Servidor RIO usados antes da autorização da redistribuição, para que o backup feito do metadado seja consistente. Neste ponto o "*rioie*" é ativado com o parametro " -c" (Tabela 4.1). Este comando lê o arquivo "ie_lst.dat" (Tabela 4.4) que tem as informações dos estados dos metadados, e realiza a recuperação da distribuição anterior a autorização do operador, fazendo para cada objeto multimídia do RIO:

   2.1. Se for igual a 0, 1 ou 2, não há nada a fazer pois a redistribuição não foi iniciada para objetos com esse estado;

   2.2. Se for igual a 3 o backup do metadado é usado para recuperar o seu estado anterior;

   2.3. Se for igual a 4 ou 5 o objeto multimídia é removido da área de trabalho (não é necessário restaurar seus blocos nos Nós), pois apenas o *download* do objeto foi feito;

   2.4. Se for igual a 6, o objeto multimídia está em remoção, e deve ser inteiramente removido dos Nós, em seguida é feita a importação do objeto a partir da área de trabalho;

   2.5. Se for igual a 7, o objeto multimídia já foi removido dos Nós e é feita a importação da área de trabalho;

   2.6. Se for igual a 8, o objeto multimídia é removido dos Nós (redistribuição cancelada) e é efetuada a importação da área de trabalho, visto que neste caso estava sendo efetuado um *upload*;

   2.7. Se for igual 9, o objeto multimídia que está na área de trabalho atualmente é removido, e em seguida é feita a exportação do objeto a ser restaurado e é feita sua remoção e por último, é feita a importação deste objeto obedecendo a redistribuição randômica.

Ao final da execução da rotina "*rioie*", é feita a atualização do arquivo de configuração "disk.cfg". A linha que contém um identificador de Nó de Armazenamento removido é excluída do arquivo. A linha que contém um identificador de Nó de Armazenamento que tenha sido inserido terá seu atributo de estado alterado para "A".

A rotina "*rioie*" contabiliza ainda os tempos de exportação, remoção e importação de cada objeto multimídia, e ao final da sua execução, contabiliza as quantidades e tempos globais de todo o processo, gravando os resultados em arquivos de log. Como a redistribuição dos blocos dos objetos multimídia é gradual e ocorre em paralelo com a execução do Servidor RIO, o balanceamento de carga de todos os discos dos Nós de Armazenamento só é alcançando no final da execução da rotina "*rioie*".

O pseudo-código da rotina "*rioie*" está no Anexo A, e o seu código fonte encontra-se no Anexo B.

Capítulo 5

Experimentos

O objetivo dos experimentos realizados é mostrar que a redistribuição dos blocos dos objetos multimídia é feita concorrentemente com o atendimento das solicitações dos cliente, reduzindo o tempo de parada necessário para a remoção e inserção de Nós de Armazenamento no RIO.

5.1 Ambiente

Para realização dos testes, utilizamos uma rede de computadores (Figura 5.1) gerenciada pelo sistema operacional Linux distribuição Mandriva 10.2. Uma das máquinas, é um Pentium IV 2.4 Ghz com 512 MB de memória e um disco de 40 GB, onde o Servidor RIO e um cliente RIO são executados. Duas outras máquinas, Pentium IV 2.4 Ghz com 256 MB de memória e disco de 40 GB cada uma, executam clientes RIO. As máquinas se interligam através de placas de rede 10/100 Ethernet.

Na máquina em que é executado o Servidor RIO, foi criado um Nó de Armazenamento. Em cada uma das outras duas máquinas, também foram criados Nós de Armazenamento, totalizando três Nós de Armazenamento. Cada Nó de Armazenamento foi configurado com discos de mesmo tamanho. A máquina em que é executado o servidor foi denominada PED1 e os discos do Nó de Armazenamento criado nesta máquina foram denominados respectivamente discoPED11 e discoPED12. As outras duas máquinas e seus respectivos discos foram denominados como: máquina PED3 com os discos discoPED31 e discoPED32; e máquina PED4 com os disco discoPED41 e discoPED42.

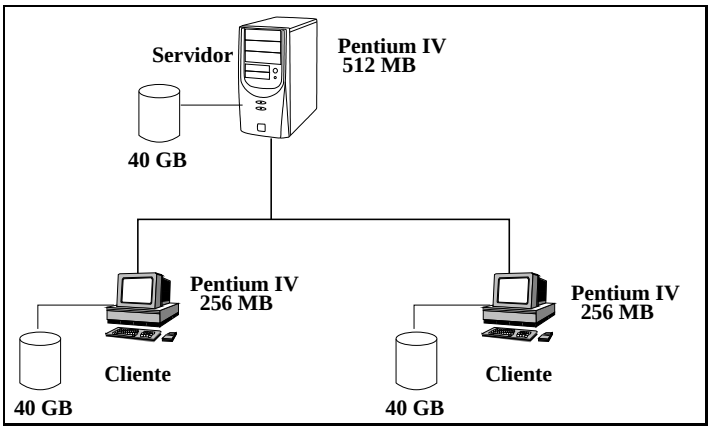

Figura 5.1: Ambiente de testes da rotina *rioie*

5.2 Testes

Para os testes, foi criado um usuário no RIO que solicita através do processo cliente a exibição de vídeo, a inserção de vídeo, a remoção de vídeo, a listagem de vídeos existente e vizualização da taxa de ocupação dos discos dos Nós de Armazenamento (funções presentes no RIO).

É importante ressaltar, que os teste visam verificar a eficácia da rotina. O desempenho da rotina *rioie* é um fator secundário, visto que esta rotina visa manter o desempenho do Servidor RIO compatível com as exigências requeridas para manter a QoS em aplicações multimídia.

Através do processo "riommclient" é possível a exibição do vídeo. Para a manutenção dos vídeos nos Nós de Armazenamento é utilizado o processo "riosh -t" do Servidor RIO, através de uma interface texto que permite a entrada de comandos para a realização desta manutenção conforme mostrado na Tabela 5.1.

Tabela 5.1: Comandos do processo riosh

Comando	**Descrição da Tarefa**
cp	permite inserir vídeo nos Nós de Armazenamento do Servidor RIO
rm	permite remover vídeo dos Nós de Armazenamento do Servidor RIO
ls	permite listar os vídeos de um cliente gravados nos Nós de Armazenamento do Servidor RIO
df	permite visualizar a taxa de ocupação dos discos nos Nós de Armazenamento do Servidor RIO

Para assegurar que a redistribuição dos blocos dos objetos multimídia entre os Nós de Armazenamento após a execução da rotina "*rioie*" obedece aos parâmetros contidos nos

arquivos de configuração, foram utilizados os comandos de console do processo "riosh -t". Por meio do comando "df" do processo "riosh -t", é possível ver a taxa (percentual) de ocupação dos discos dos Nós de Armazenamento pelos vídeos. Assim, após a remoção de um Nó de Armazenamento e eventual mudança no número de réplicas, após a redistribuição feita pela rotina "*rioie*" nos Nós de Armazenamento, podemos conferir executando mais uma vez o comando "df" do processo "riosh -t", se a taxa resultante confirma a redistribuição esperada.

Em outras palavras, o espaço ocupado por um conjunto de vídeos utilizando um determinado número de réplicas, nos Nós de Armazenamento do Servidor RIO, implica numa taxa de ocupação conhecida nos Nós de Armazenamento. Após a mudança de configuração dos Nós de Armazenamento e eventual mudança do número de réplicas, podemos calcular a nova taxa de ocupação dos vídeos sobre o novo cenário. Assim sendo, após a realização da redistribuição, o comando "df" do processo "riosh -t", confirma se a nova taxa de ocupação esperada foi realmente respeitada pela rotina "*rioie*".

Analogamente podemos utilizar o mesmo procedimento para aferir se a inserção de Nó de Armazenamento, foi realizada corretamente pela rotina "*rioie*".

Como a rotina "*rioie*" também apura os tempos de exportação, importação e atualização dos metadados (remapeamento), contabilizando e gravando esses tempos em um arquivo log, é também possível observar os tempos consumidos no tratamento de cada vídeo, bem como o tempo global de toda a execução de redistribuição.

Foram realizadas três baterias de testes: na primeira foi utilizada uma configuração de Nós de Armazenamento com dois discos de 40MB em cada nó, resultando num espaço total de armazenamento de 240 MB; na segunda bateria de testes foi utilizada uma configuração de Nós de Armazenamento com dois discos de 2GB em cada nó, totalizando um espaço de 12 GB; e na terceira bateria foi mantida a configuração dos Nós de Armazenamento. Na tabela 5.2 temos o resumo das configurações que foram utilizadas nos experimentos da rotina "*rioie*".

Tabela 5.2: Configurações utilizadas nos experimentos

Baterias	**No. de Nós**	**No. de discos por Nó**	**Tamanho dos discos (MB)**	**No. de réplicas**
Bateria 01	3	2	40	2 e 1
Bateria 02	3	2	2000	2
Bateria 03	3	2	2000	2

5.2.1 Bateria 01

Na primeira bateria de testes, foram usadas duas vídeo-aulas, ambas com 57,5 MB de tamanho, totalizando 115 MB. Foram usados três Nós de Armazenamento com dois discos de 40MB cada um, totalizando 240 MB de espaço total. Foi fixado um número de réplica igual a dois. A preparação do RIO para inclusão das vídeo-aulas exige inicialmente a formatação dos discos. Essa formatação é realizada por uma rotina do Servidor RIO seguindo parâmetros contidos em arquivo de configuração específico. A Figura 5.2 e Figura 5.3 ilustram a seqüência de ações realizadas pela rotina de formatação, mostrando os parâmetros usados.

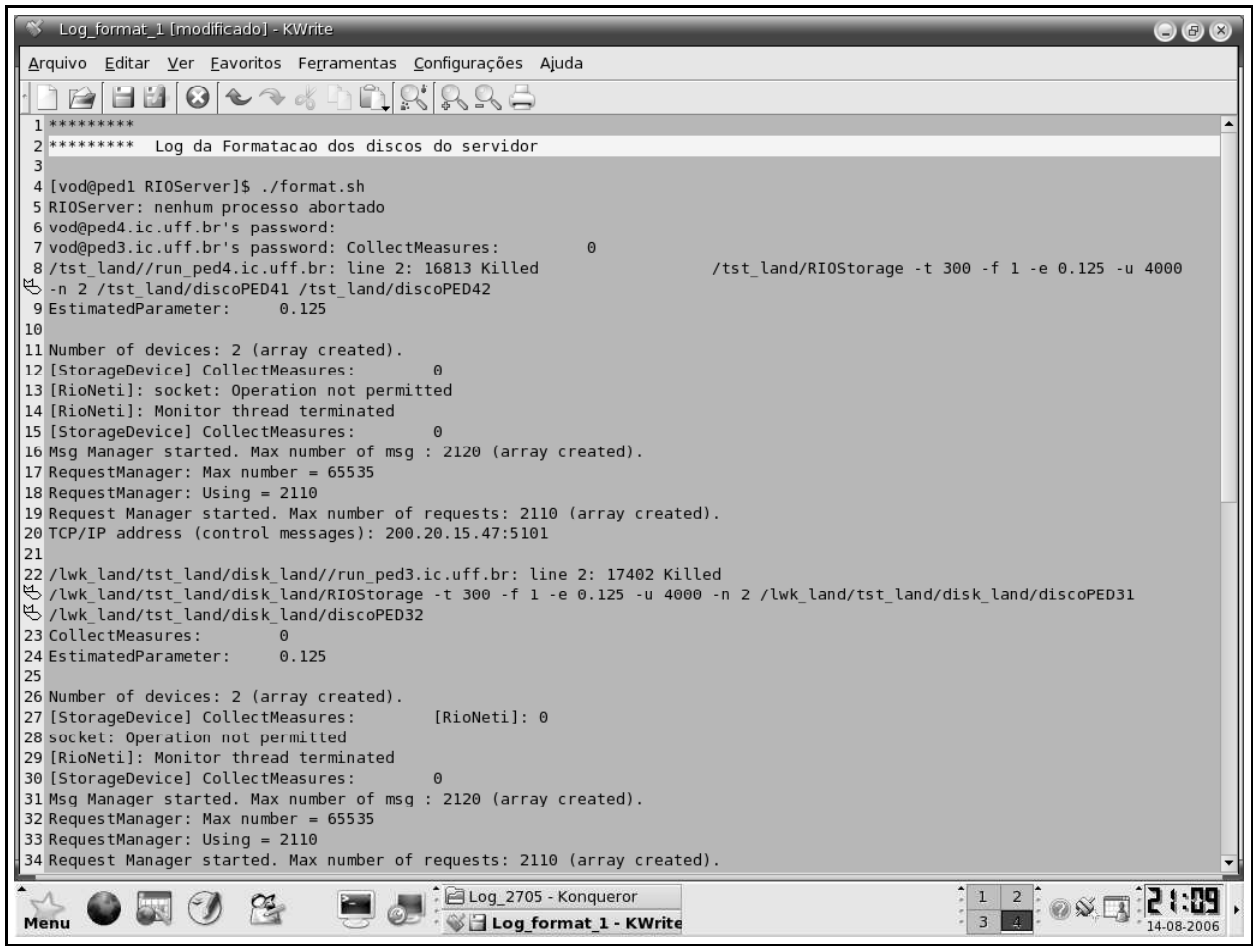

Figura 5.2: Tela inicial de formatação

Na Figura 5.2, observamos os discos dos Nós de Armazenamento que estão sendo formatados - PED41 e PED42, PED31 e PED32, bem como os parâmetros que o RIOStorage utiliza para formatá-los.

Figura 5.3: Tela inicial de formatação (cont..)

Na Figura 5.3, observamos a finalização da formatação, exibindo os parâmetros definidos no arquivo de configuração "system.cfg", que determinam o funcionamento e desempenho do Servidor RIO, tais como o número de réplicas. Ao final o sistema exibe o número de discos presentes no Servidor RIO.

Concluida a formatação dos discos, o operador executa o Servidor RIO que exibe na tela a seqüência de ações como ilustrado nas Figuras 5.4, 5.5 e 5.6.

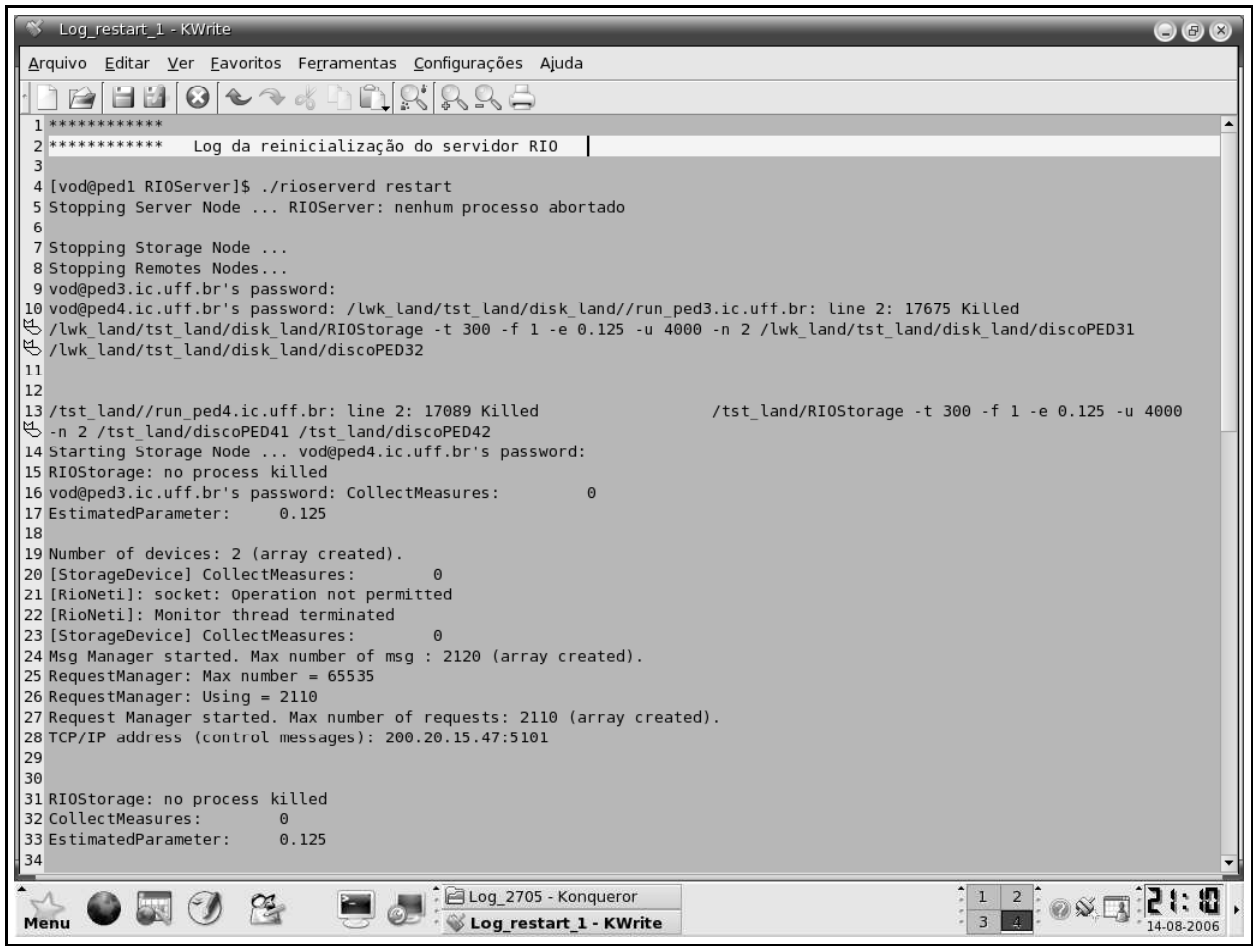

Figura 5.4: Inicialização do Servidor RIO

Na Figura 5.4, observamos o procedimento de iniciar o Servidor RIO, identificando os discos que estão sendo conectados ao Servidor RIO, (PED31, PED32, PED41 e ped42).

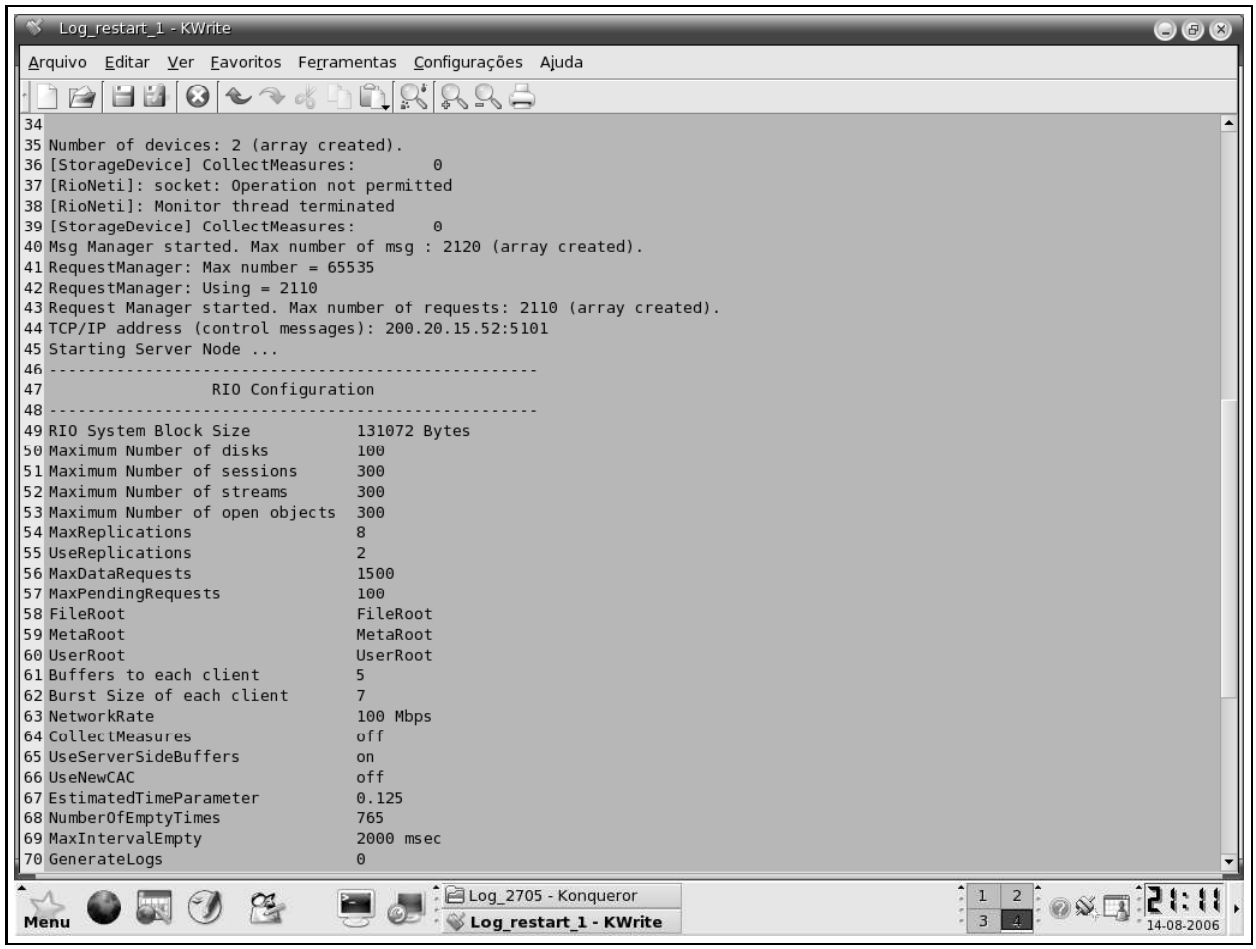

Figura 5.5: Inicialização do Servidor RIO (cont..)

Na Figura 5.5, observamos o procedimento de iniciar o Servidor RIO, exibindo os parâmetros de configuração do Servidor RIO, que determinam o funcionamento do servidor.

Figura 5.6: Finalização do procedimento de inicialização do Servidor RIO

Na Figura 5.6, observamos a finalização do procedimento de iniciar o Servidor RIO, exibindo o número de discos conectados ao servidor, e iniciando os módulos necessários a execução do Servidor RIO.

Dando prosseguimento a preparação dos testes, através do processo "riosh", com o comando "df", obtivemos a taxa de ocupação dos discos vazios, como visto na Figura 5.7. Copiamos os vídeos, através do comando "cp" do mesmo processo "riosh", e em seguida observamos novamente a taxa de ocupação após a cópia dos vídeos (Figura 5.8), verificando então uma taxa média de ocupação de aproximadamente 96% em cada Nó de Armazenamento.

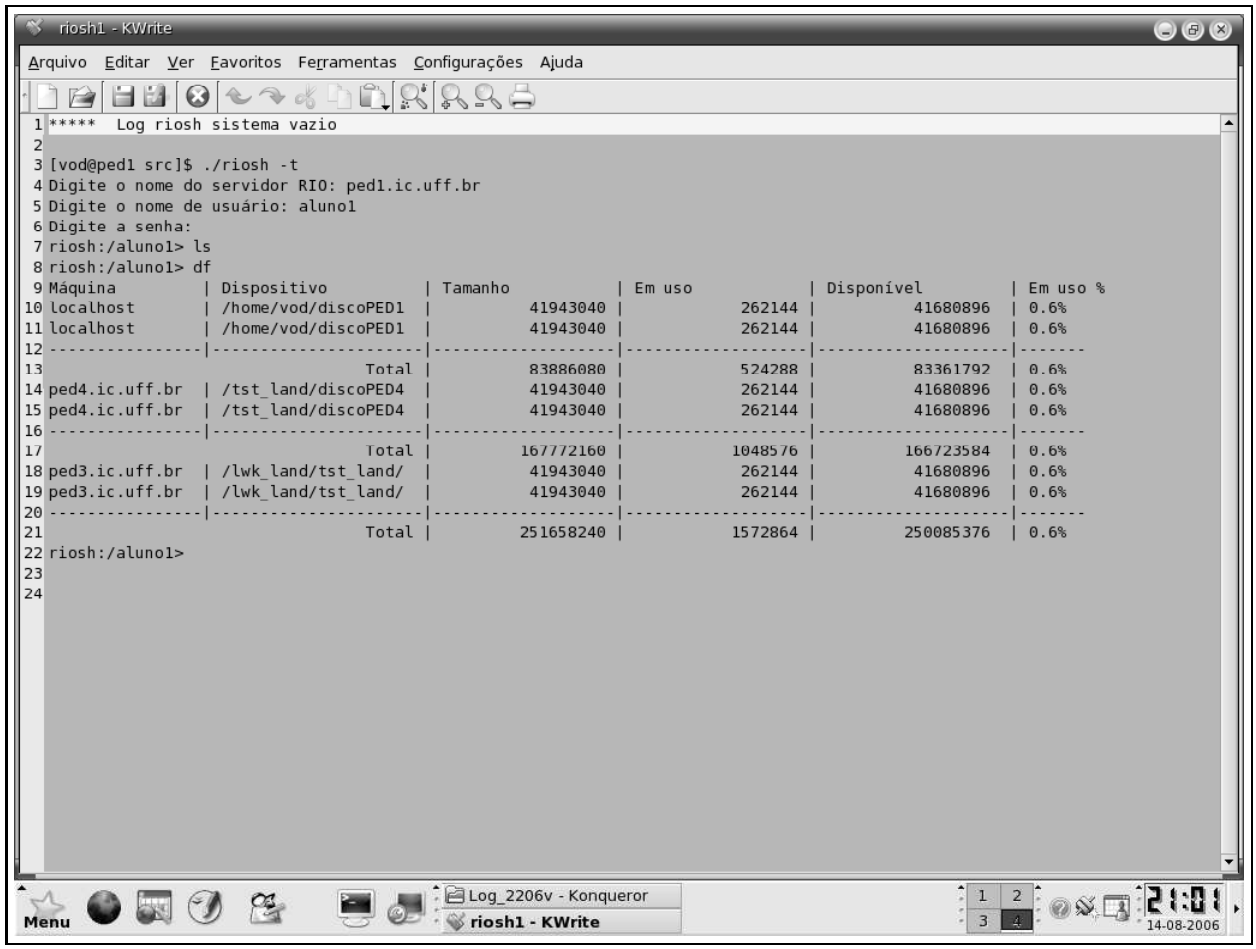

Figura 5.7: Taxa de ocupação dos discos dos Nós de Armazenamento quando vazio

Na Figura 5.7, observamos a taxa de ocupação do conjunto de Nós de Armazenamento, quando não existe nenhum vídeo armazenado.

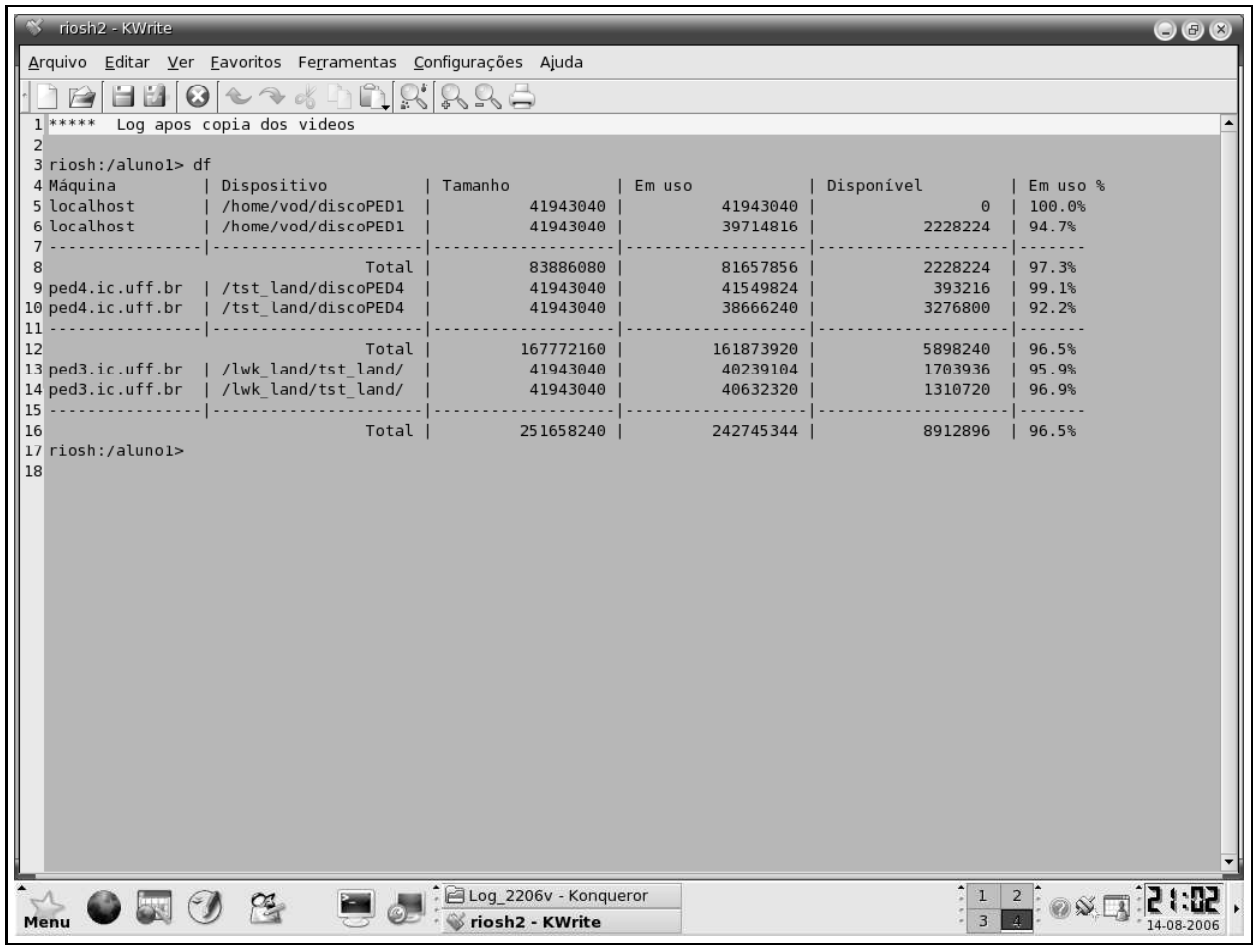

Figura 5.8: Taxa de ocupação após inserção das duas vídeo-aulas

Na Figura 5.8, observamos a taxa de ocupação do conjunto de Nós de Armazenamento, após a inserção de duas vídeo-aulas com aproximadamente 96,7%.

Preparado o ambiente para testes, realizamos dois experimentos:

- remoção de um Nó de Armazenamento; e
- inserção do Nó de Armazenamento.

Na remoção de um Nó de Armazenamento, simulamos uma falha de conexão ou remoção do Nó de Armazenamento, retirando o cabo de rede da máquinas PED3. Os vídeos que estavam sendo assistidos nos clientes RIO nas demais máquinas, apresentaram algumas irregularidades na sua exibição, já que os blocos armazenados no nó removido, não conseguem ser recuperados.

Em seguida, foram interrompidas propositadamente pelo operador, as exibições de vídeos das máquinas ativas, e o Servidor RIO também foi interrompido para que fossem feitas as alterações necessárias nos arquivos de configuração. O número de réplicas foi modificado para um porque na nova configuração o espaço disponível não comportaria os objetos armazenados se mantido o número de réplicas anterior. Coube ao operador a detectar essa impossibilidade e a corespondente alteração do parâmetro.

O operador acrescenta então o atributo "R" no arquivo "disk.cfg", na linha correspondente ao Nó de Armazenamento que foi removido. No *script* "rioserverd", foi retirada a linha de conexão via "ssh" correspondente ao Nó de Armazenamento que foi removido.

Quando o Servidor RIO foi reiniciado, a rotina "*rioie*", detectou o registro da remoção do Nó de Armazenamento e após autorização via console iniciou a atualização dos metadados, que permite ao Servidor RIO prover a exibição dos vídeos solicitados pelos clientes, enquanto a rotina "*rioie*" realiza gradualmente a redistribuição randômica de cada objeto multimídia pelos discos dos Nós de Armazenamento do novo cenário, conforme visto nas figuras Figura 5.9, Figura 5.10 e Figura 5.11. Através do "riosh", podemos observar a ausência do nó removido, antes mesmo de fazer a redistribuição, conforme Figura 5.12.

O tempo médio que um cliente fica em espera para assistir a um determinado vídeo, é de no máximo o tempo necessário para a reinserção do vídeo (através do procedimento de redistribuição dos blocos do vídeo por todos os Nós de Armazenamento da configuração atual). Caso o vídeo desejado não esteja sendo redistribuído o acesso é imediato. A rotina *rioie* redistribui um vídeo apenas de cada vez.

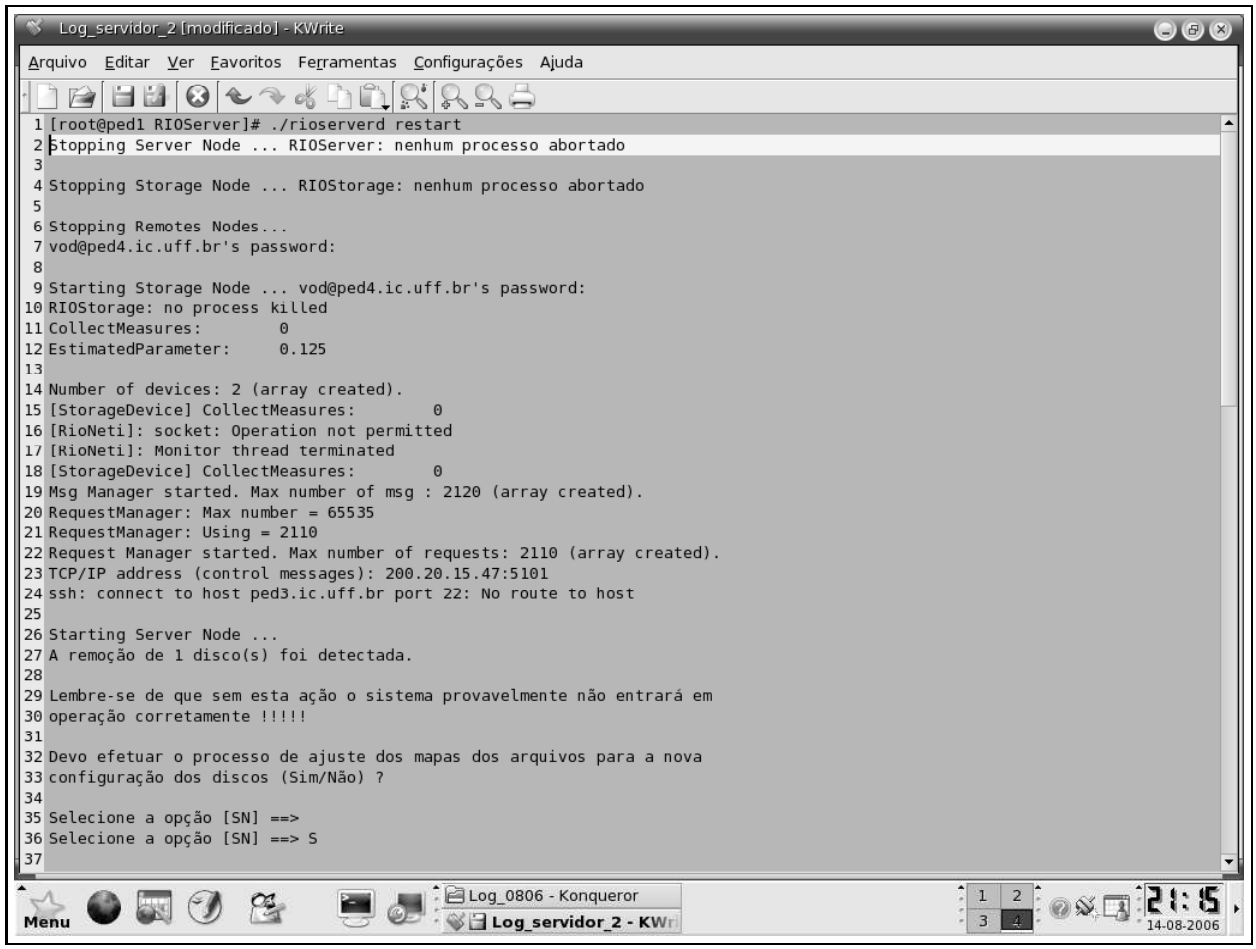

Figura 5.9: Reinício do Servidor RIO com detecção da remoção do Nó de Armazenamento

Na Figura 5.9 observamos a rotina "*rioie*", detectando a remoção de um Nó de armazenamento, e solicitando do operador, autorização para permitir atualizar os metadados, através do processo de ajuste dos mapas.

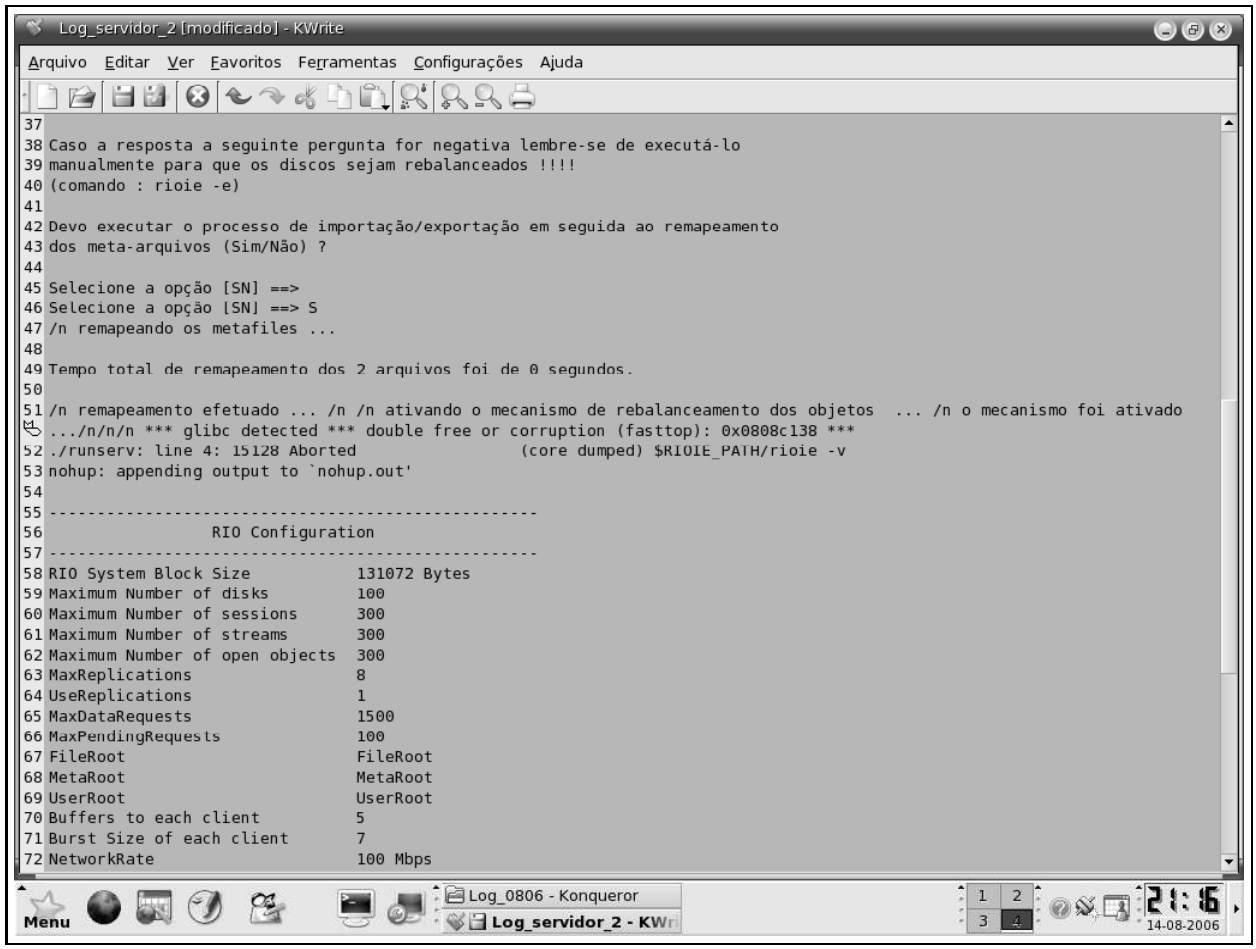

Figura 5.10: Reinício do Servidor RIO com detecção da remoção do Nó de Armazenamento (cont..)

Na Figura 5.10 observamos a rotina "*rioie*", solicitando do operador, autorização para iniciar a redistribuição randômica dos blocos dos objetos multimídia na nova configuração de Nós de Armazenamento e exibindo os parâmetros de configuração do Servidor RIO, ao reiniciar o Servidor RIO.

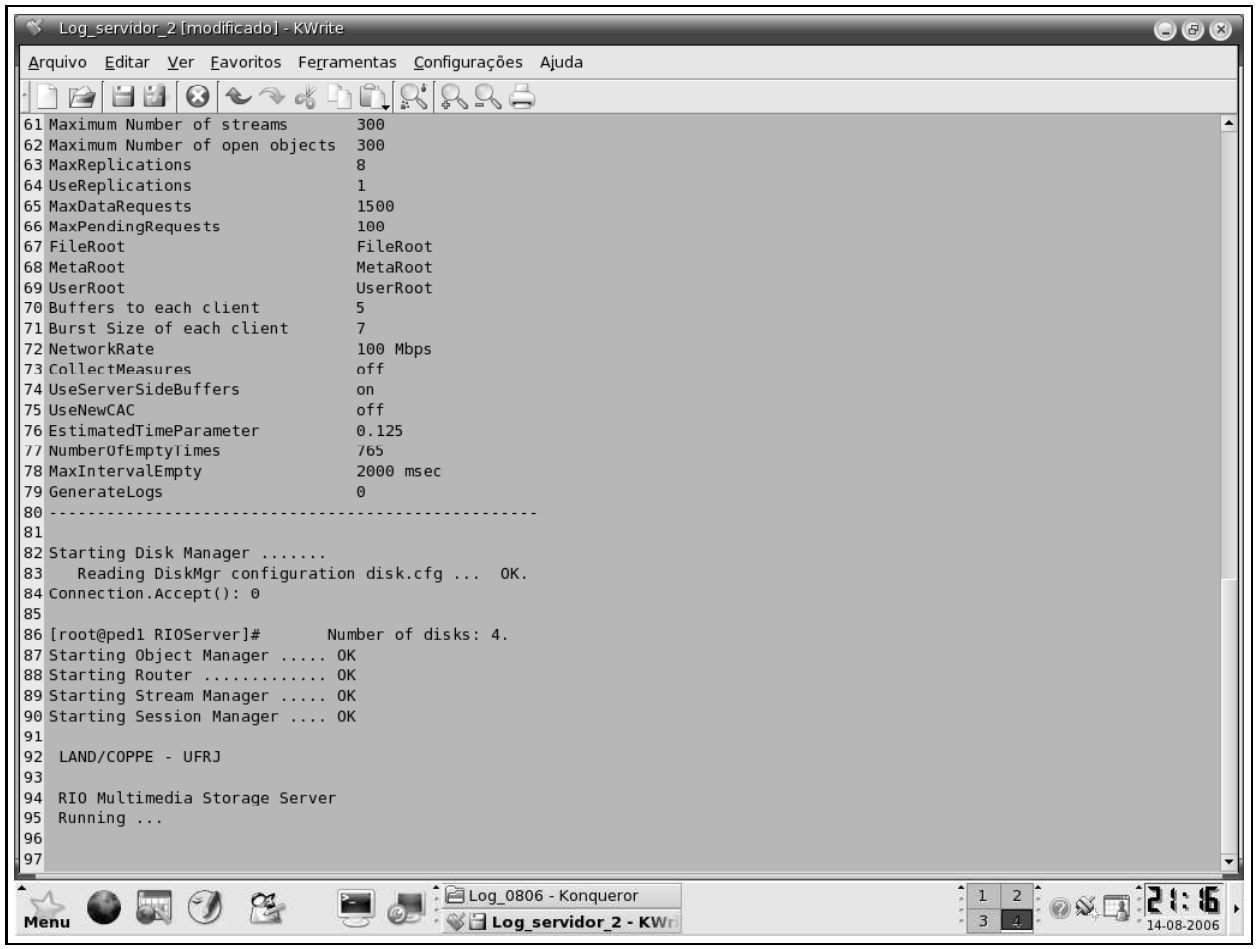

Figura 5.11: Reinício do Servidor RIO com detecção da remoção do Nó de Armazenamento (cont..)

Na Figura 5.11 observamos a finalização do procedimento de reiniciar o Servidor RIO, exibindo o número de discos disponíveis após a mudança de configuração dos Nós de Armazenamento e iniciando os módulos necessários a execução do Servidor RIO. Em paralelo a execução do Servidor RIO, a rotina "*rioie*", realiza a gradual redistribuição randômica dos blocos dos objeto multimídia nos Nós de Armazenamento da nova configuração.

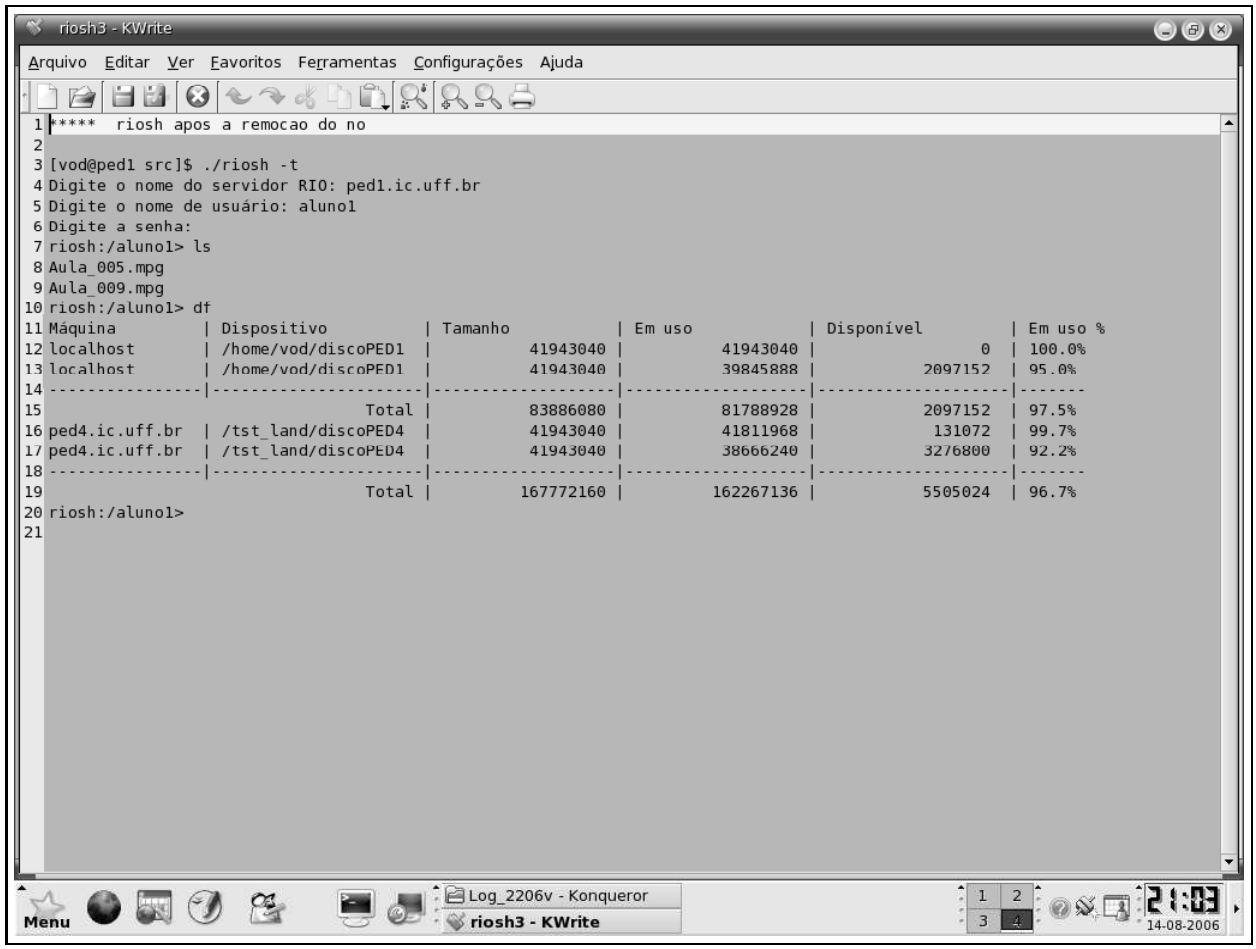

Figura 5.12: Taxa de ocupação após remoção de um Nó de Armazenamento

Na Figura 5.12 observamos que o processo "riosh" não contabiliza o espaço de armazenamento dos discos do Nó de Armazenamento da máquina PED3, que foi removido, isto é, os discos discoPED31 e discoPED32, não aparecem mais na contabilização do processo "riosh".

Após a execução da rotina "*rioie*" foi verificado uma taxa média de ocupação de aproximadamente 72% em cada disco dos Nós de Armazenamento, uma vez que foi gravado um conjunto de vídeos de 115MB de tamanho num espaço de armazenamento de 160MB. Este resultado comprova a correta redistribuição dos objetos multimídia respeitando o algoritmo randômico no Servidor RIO, conforme visto na Figura 5.13.

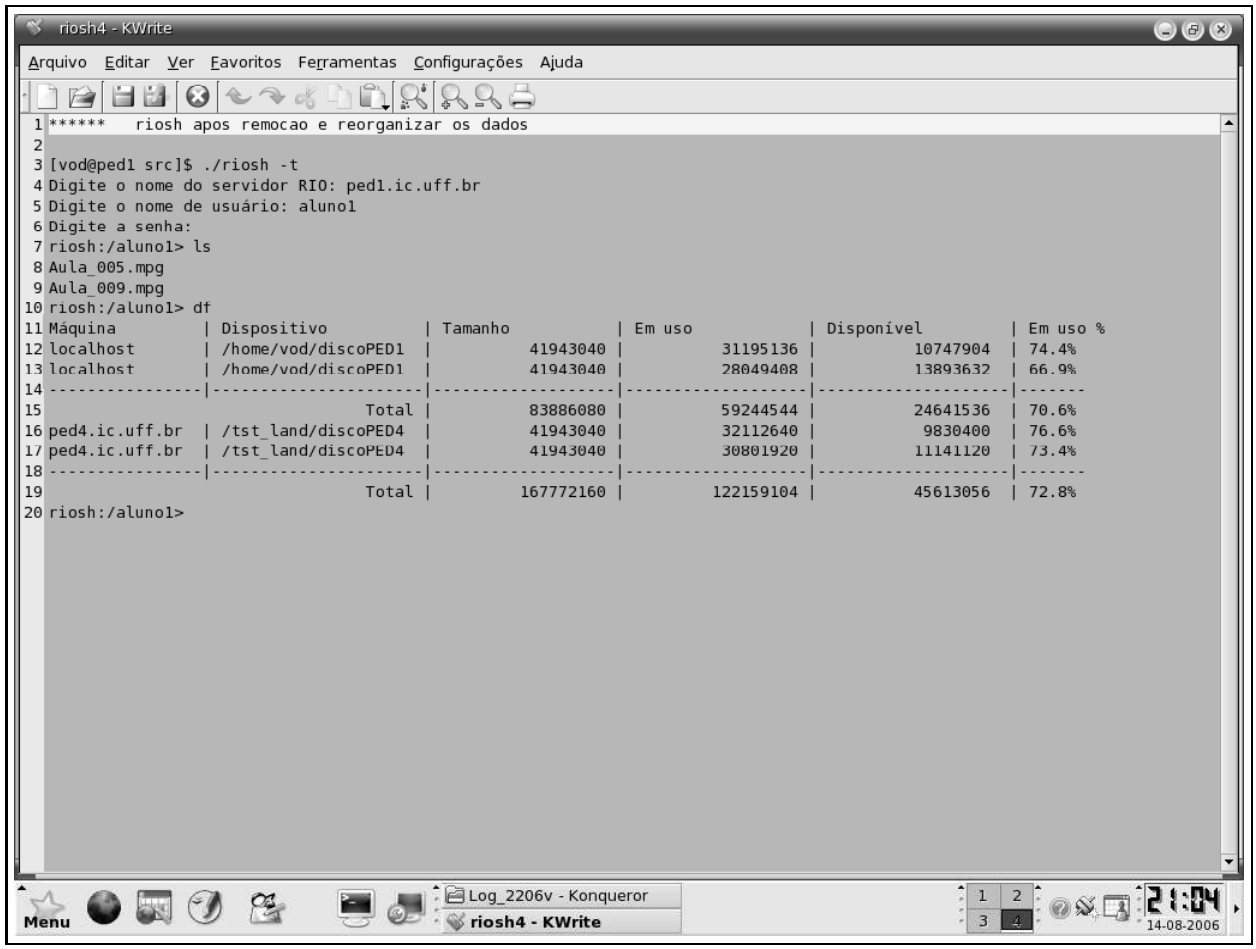

Figura 5.13: Taxa de ocupação após o "*rioie*" realizar a redistribuição

Na inserção do Nó de Armazenamento utilizando o cenário resultante da experiência anterior, foi reconectada a máquina PED3. De modo análogo à remoção do Nó de Armazenamento, o Servidor RIO foi interrompido para fazer as alterações necessária nos arquivos de configuração.

No arquivo "disk.cfg", foi acrescentada uma linha identificando o Nó de Armazenamento que está sendo inserido e adicionado o atributo "I". No *script* "rioserverd" foi acrescentada a linha de conexão via "ssh" correspondente ao Nó de Armazenamento que foi inserido. Quanto ao número de réplicas, mediante avaliação do espaço no novo cenário, este valor pode ser alterado ou não. Assim, optamos por manter o número de réplicas.

No Nó de Armazenamento inserido, foi feita a recriação dos discos. Uma vez reiniciado o Servidor RIO, a rotina "*rioie*" detectou o registro da inserção do Nó de Armazenamento e após autorização via console, iniciou a redistribuição gradual dos objetos multimídias pelos discos dos Nós de Armazenamento, enquanto o Servidor RIO atendia as requisições de exibição dos vídeos dos clientes conforme visto nas figuras Figura 5.14, Figura 5.15 e Figura 5.16. É importante observar que na inserção do Nó de Armazenamento não foi necessário o procedimento de atualização dos metadados, já que os metadados não possuem nenhum bloco de dados inconsistente.

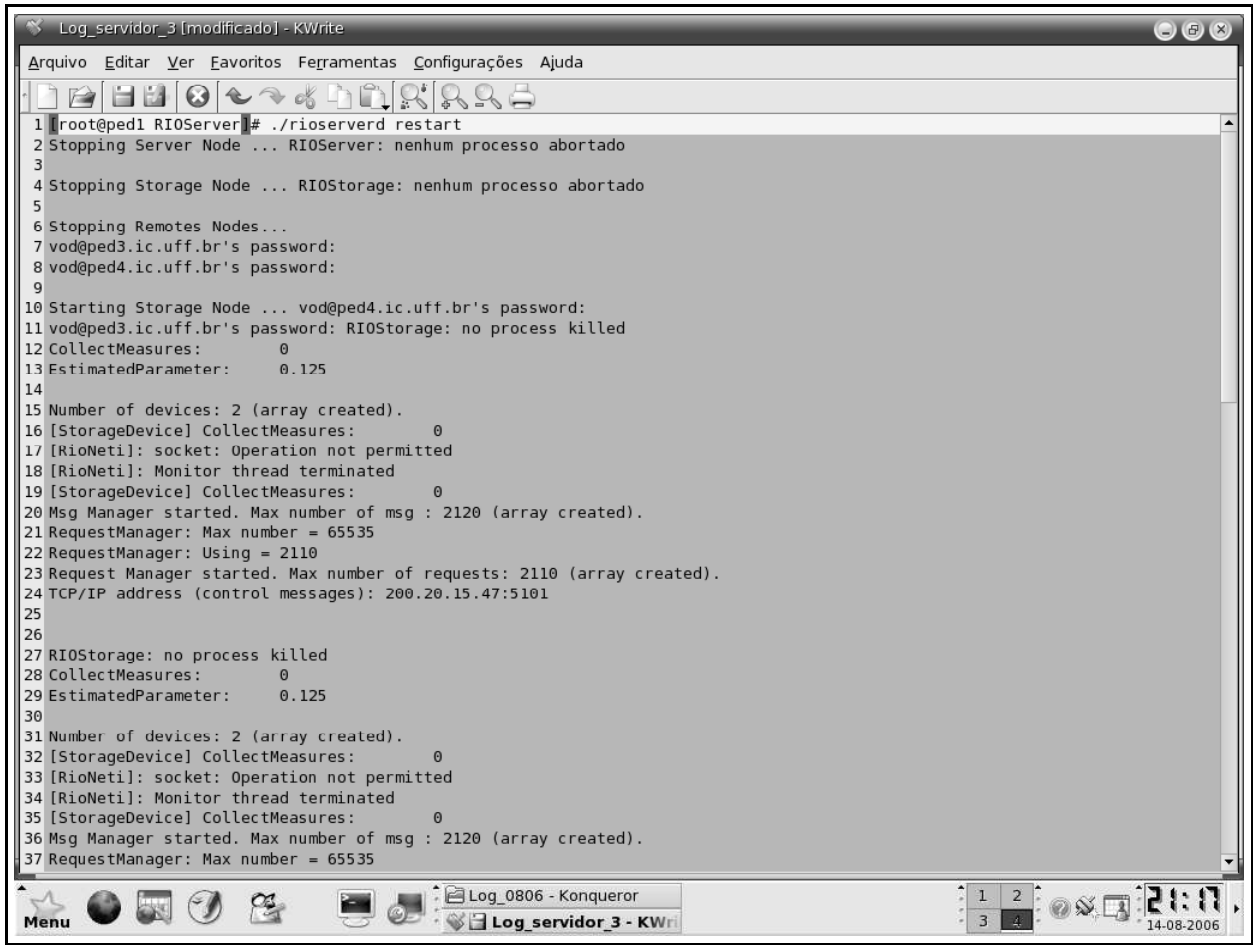

Figura 5.14: Reinício do Servidor RIO após a inserção do Nó de Armazenamento

Na Figura 5.14, observamos o reinicio do Servidor RIO, após ter sido inserido um novo Nó de Armazenamento.

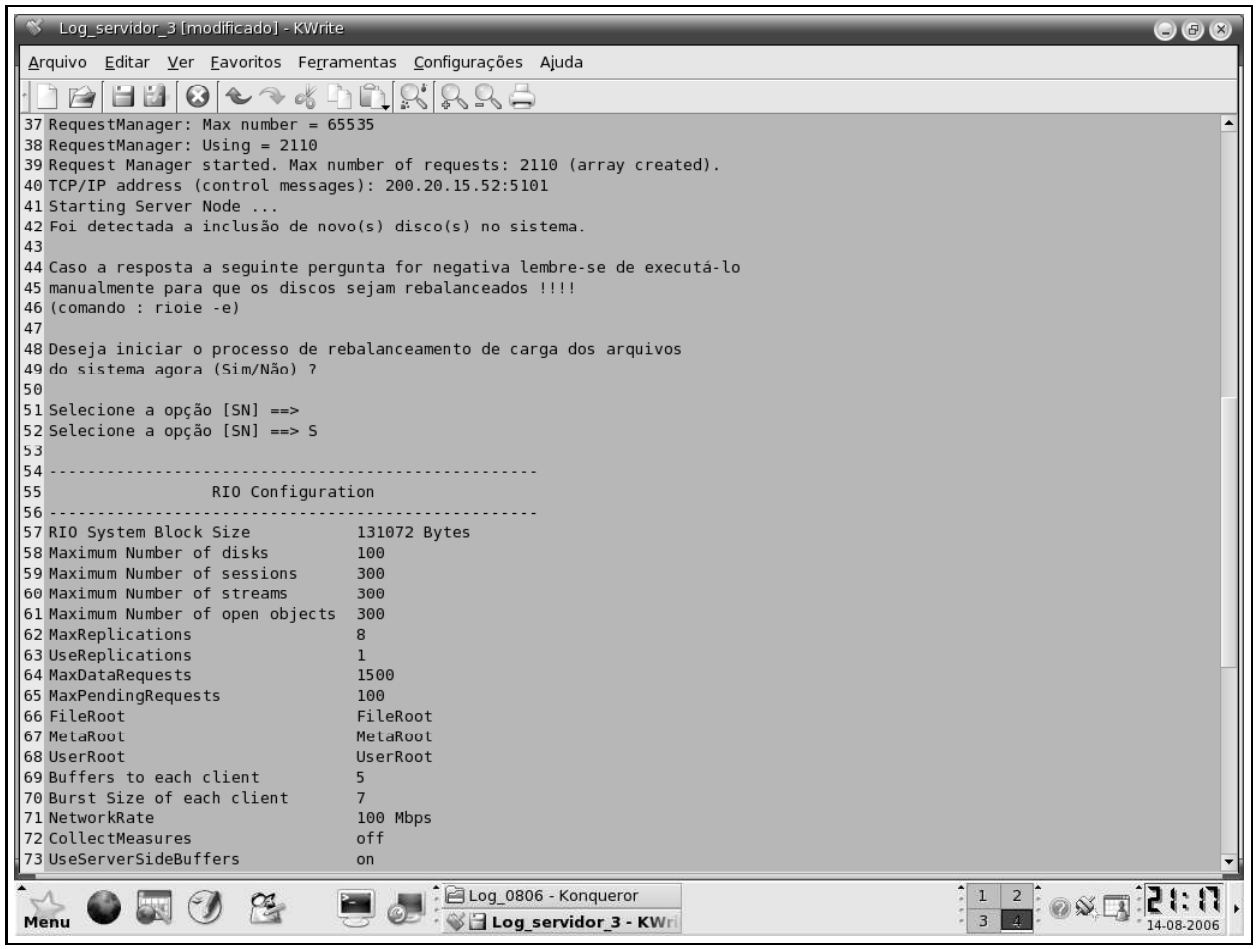

Figura 5.15: Reinício do Servidor RIO com detecção da inserção do Nó de Armazenamento (cont..)

Na Figura 5.15, observamos a rotina "*rioie*" detectando a inserção de um Nó de Armazenamento e solicitando do operador autorização para iniciar a redistribuição randômica dos blocos dos objetos multimídia nos Nós de Armazenamento na nova configuração.

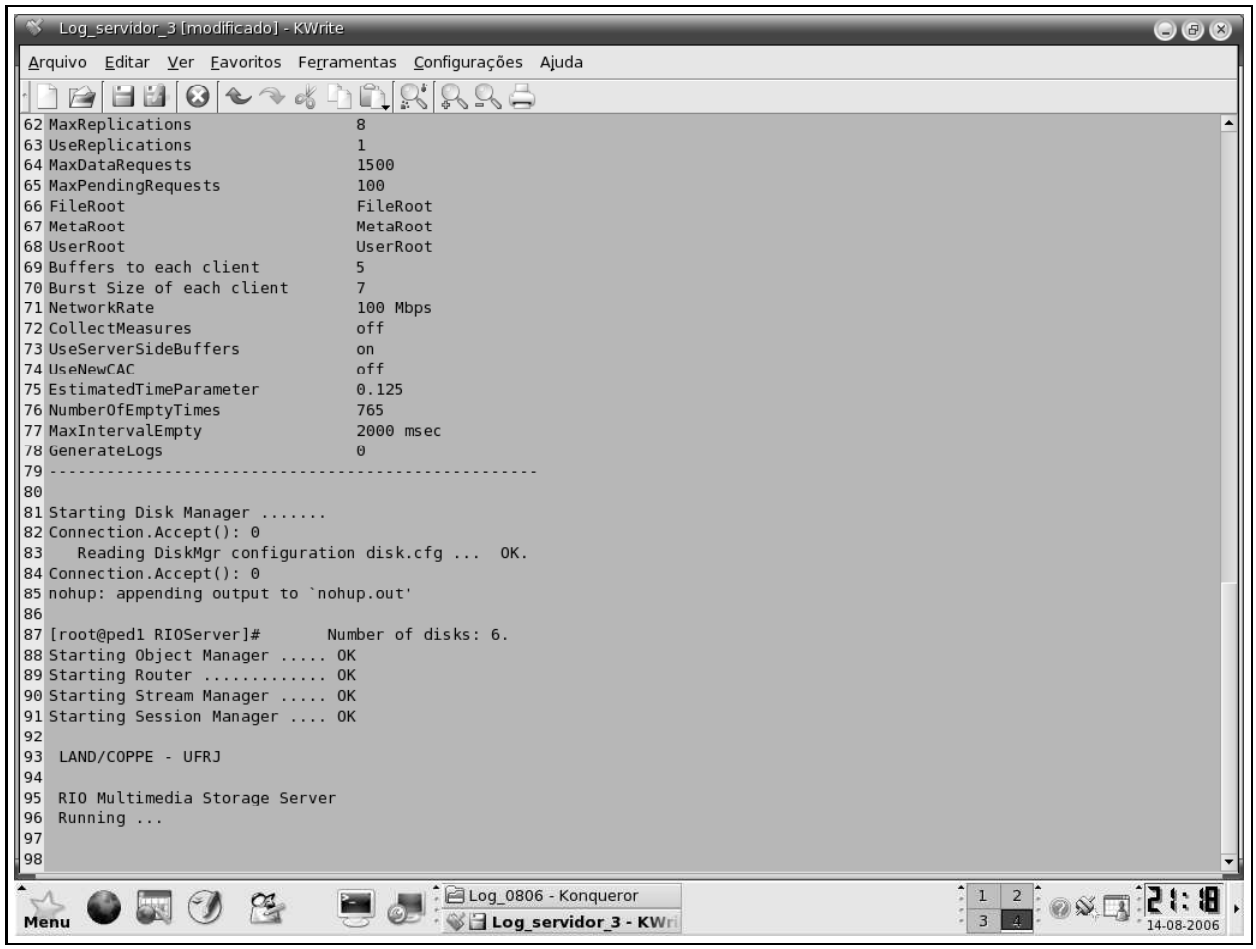

Figura 5.16: Finalização do procedimento de reínicio do Servidor RIO na inserção de um Nó de Armazenamento

Na Figura 5.16, observamos a finalização do procedimento de reiniciar o Servidor RIO, após a rotina "*rioie*" ter detectado a inserção de um Nó de Armazenamento, exibindo os parâmetros definidos nos arquivos de configuração, destacando o número de discos conectados ao servidor e iniciando os módulos necessários a execução do Servidor RIO.

Após a execução da rotina "*rioie*", foi verificado uma taxa média de ocupação de aproximadamente 48% em cada disco dos Nós de Armazenamento, uma vez que foi redistribuido um conjunto de vídeos de 115MB de tamanho num espaço de armazenamento de 240MB conforme Figura 5.17.

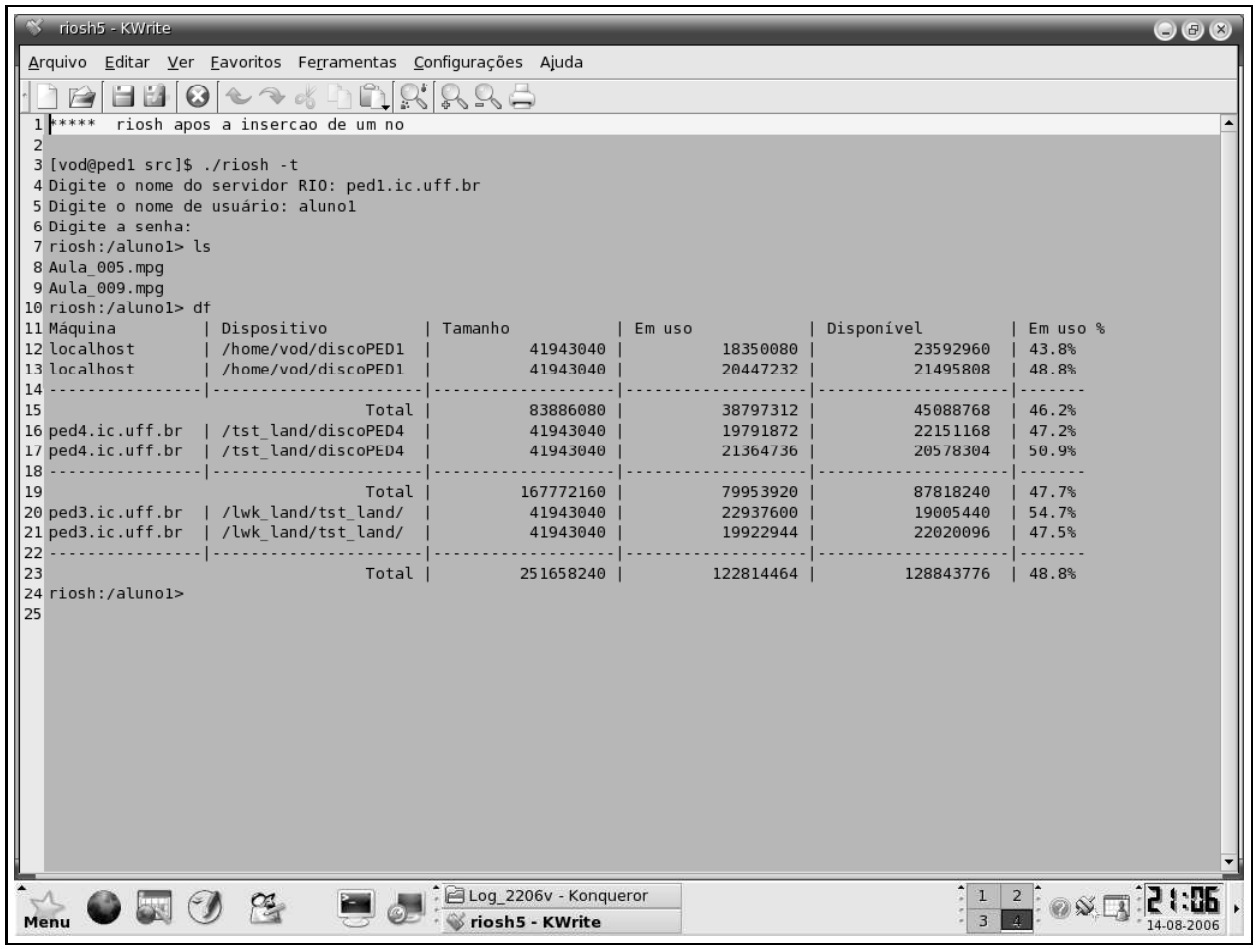

Figura 5.17: Taxa de ocupação dos discos após o "*rioie*" realizar a redistribuição, após inserção de nó

A seguir temos os tempos apurados das tarefas de redistribuição para cada objeto realizado pela rotina "*rioie*" durante a remoção de um Nó de Armazenamento conforme pode ser visto nas Tabela 5.3 e 5.4. Em seguida a apuração global de toda execução da rotina "*rioie*" pode ser vista na Tabela 5.5.

Todos os tempos apurados, foram obtidos em em segundos com precisão de milésimos de segundos. O significado do cabeçalho de cada coluna nas tabelas de apuração são descritos a seguir:

- Estado: qual o estado do objeto ao final da redistribuição;
- Usuário: nome do usuário que é o dono do objeto que foi redistribuído;
- Objeto: nome do objeto;

- Tamanho: tamanho do objeto em bytes;
- Download: número de blocos copiados para a área de backup;
- Download Time: tempo apurado para a realização do Download;
- Upload Time: tempo apurado para a realização do Upload;
- Remova_time: tempo apurado para a remoção do objeto dos Nós de Armazenamento;
- Map_time: tempo apurado para atualizar o metadado do objeto em redistribuição;
- Núm.Réplicas: número de réplicas utilizadas no objeto;
- Núm.Blocos: número de blocos movimentados do objeto durante a redistribuição;
- Bloqueado: informa se o metadado está bloqueado(1) ou não(0) ao final da redistribuição;

Nas tabelas de apurações globais, nas respectivas colunas, temos apenas o somatório de todos os valores apurados para todos os objetos multimídia, após a redistribuição.

Tabela 5.3: Apurações dos tempos da redistribuição na remoção do Nó de Armazenamento

Estado	Usuário	Objeto	Tamanho	Downloaded	Download Time	Upload Time
Terminado	aluno1	Aula_005.mpg	60241080	60241080	4,975	6,995
Terminado	aluno1	Aula_009.mpg	60241080	60241080	6,342	6,460

Tabela 5.4: Apurações dos tempos(2) da redistribuição na remoção do Nó de Armazenamento

Objeto	Removal Time	Map Time	Núm.Réplicas	Núm.Blocos	Bloqueado(0/1)
Aula_005.mpg	0,078	0,001	2	460	0
Aula_009.mpg	0,096	0,001	2	460	0

Tabela 5.5: Apurações global da redistribuição na remoção do Nó de Armazenamento

Tamanho da base	Upload Time	Download Time	Removal Time	Map Time
120482160	13,455	11,317	0,174	0,002

A seguir temos o tempo apurado (segundos) das tarefas de redistribuição para cada objeto realizado pela rotina "*rioie*" durante a inserção de um Nó de Armazenamento. (Tabelas 5.6 e 5.7). Em seguida a apuração global de toda execução da rotina "*rioie*" (Tabela 5.8).

Tabela 5.6: Apurações dos tempos da redistribuição na inserção do Nó de Armazenamento

Estado	Usuário	Objeto	Tamanho	Downloaded	Download Time	Upload Time
Terminado	aluno1	Aula_005.mpg	60241080	60241080	4,610	6,511
Terminado	aluno1	Aula_009.mpg	60241080	60241080	4,613	6,827

Tabela 5.7: Apurações dos tempos (2) da redistribuição na inserção do Nó de Armazenamento

Objeto	Removal Time	Map Time	Núm.Réplicas	Núm.Blocos	Bloqueado(0/1)
Aula_005.mpg	0,343	0,000	1	460	0
Aula_009.mpg	0,034	0,000	1	460	0

Tabela 5.8: Apurações global da redistribuição na inserção do Nó de Armazenamento

Tamanho da base	Upload Time	Download Time	Removal Time	Map Time
120482160	13,338	9,223	0,377	0,000

5.2.2 Bateria 02

Foi utilizada nesta bateria,uma configuração de Nós de Armazenamento com dois discos de 2GB em cada nó, totalizando um espaço de 12 GB. Foram usadas as seguintes vídeos-aulas: Aula_012 (320 MB), Aula_017 (508,7 MB), Aula_022 (474,1 MB) e Aula_023 (459 MB), totalizando 1761.8 MB ou 1,76GB de tamanho. Foi fixado um número de réplicas igual a dois. Analogamente, foram feitos testes semelhantes, com esta nova base de dados, obtendo os resultados contidos nas Figuras 5.18 a 5.32.

São apresentados a seguir os testes de remoção e inserçao de um Nó de Armazenamento, realizado na terceira bateria de testes.

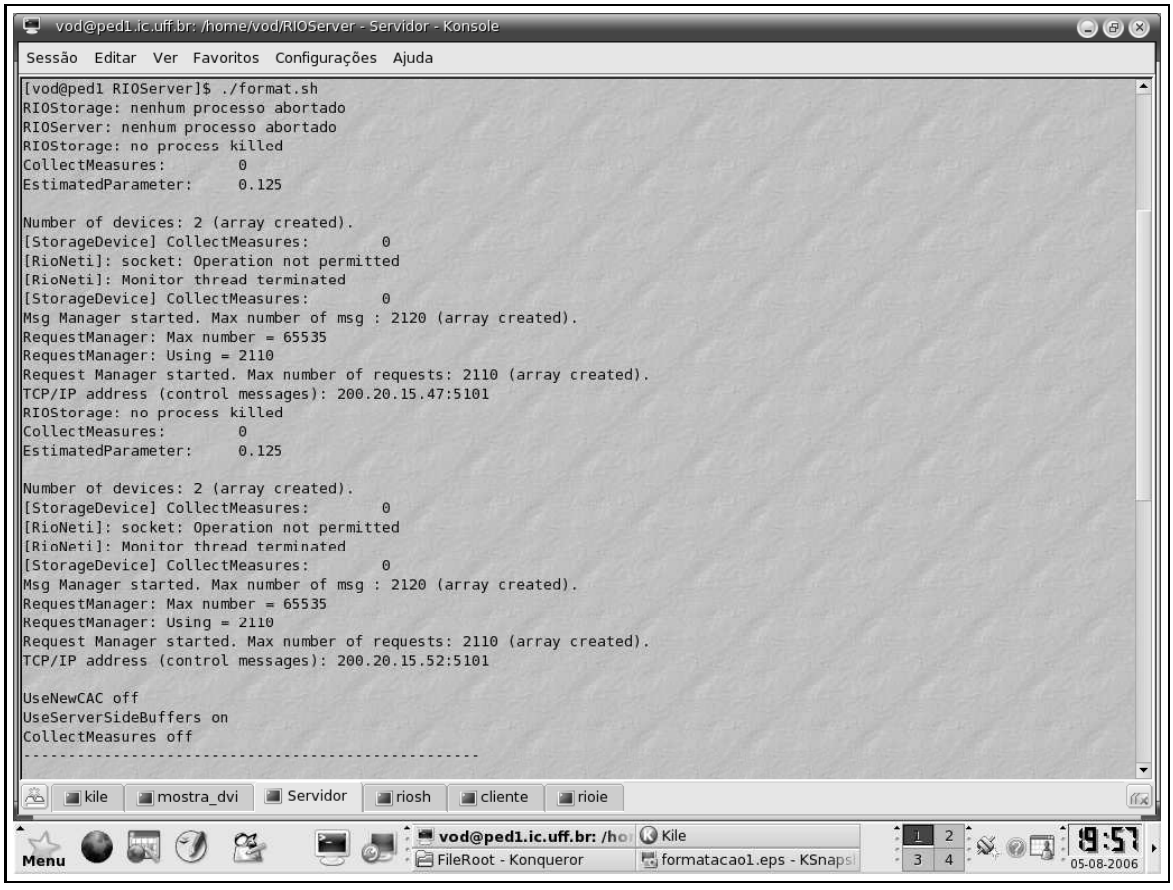

Figura 5.18: Início da Formatação dos discos dos Nós de Armazenamento

Podemos observar na Figura 5.18 o ínicio da formatação dos discos dos Nós de Armazenamento do Servidor RIO, identificando as máquinas conectadas ao servidor através do seu endereço IP.

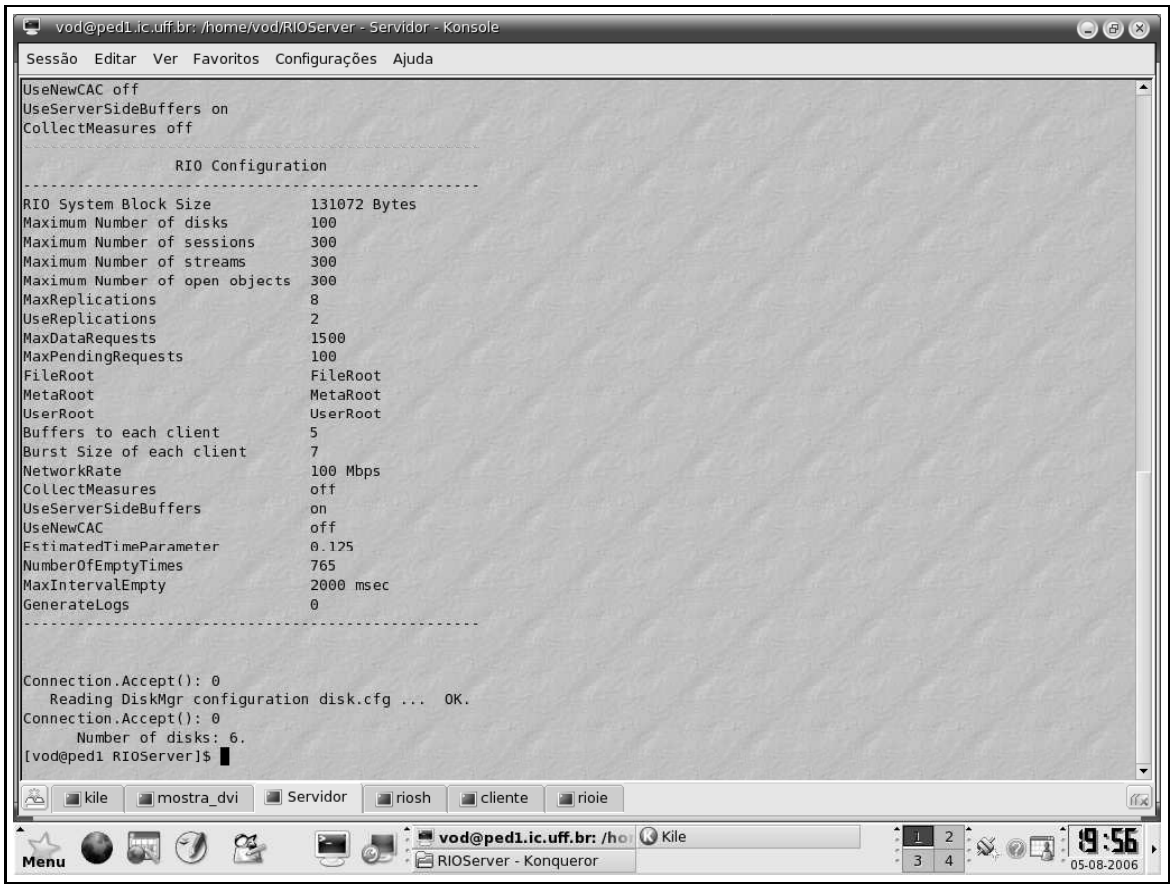

Figura 5.19: Finalização da Formatação dos discos do Nó de Armazenamento

Na Figura 5.19 observamos a finalização da formatação, exibindo os parâmetros definidos no arquivo de configuração "system.cfg", bem como identificando quantos discos estão conectados ao servidor.

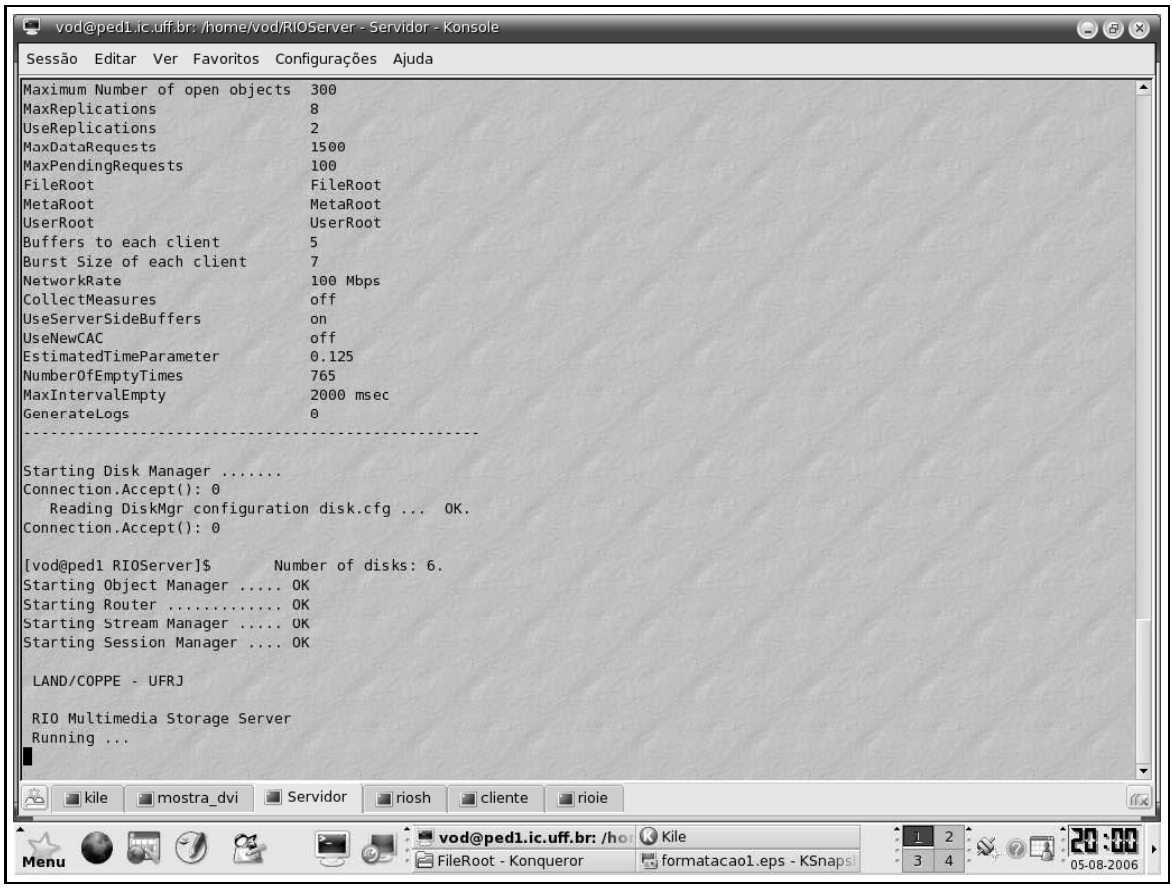

Figura 5.20: Inicialização do Servidor RIO

Na Figura 5.20 observamos a finalização do processo de reínicio do Servidor RIO exibindo o número de discos disponíveis e iniciando os módulos necessários a execução do Servidor RIO.

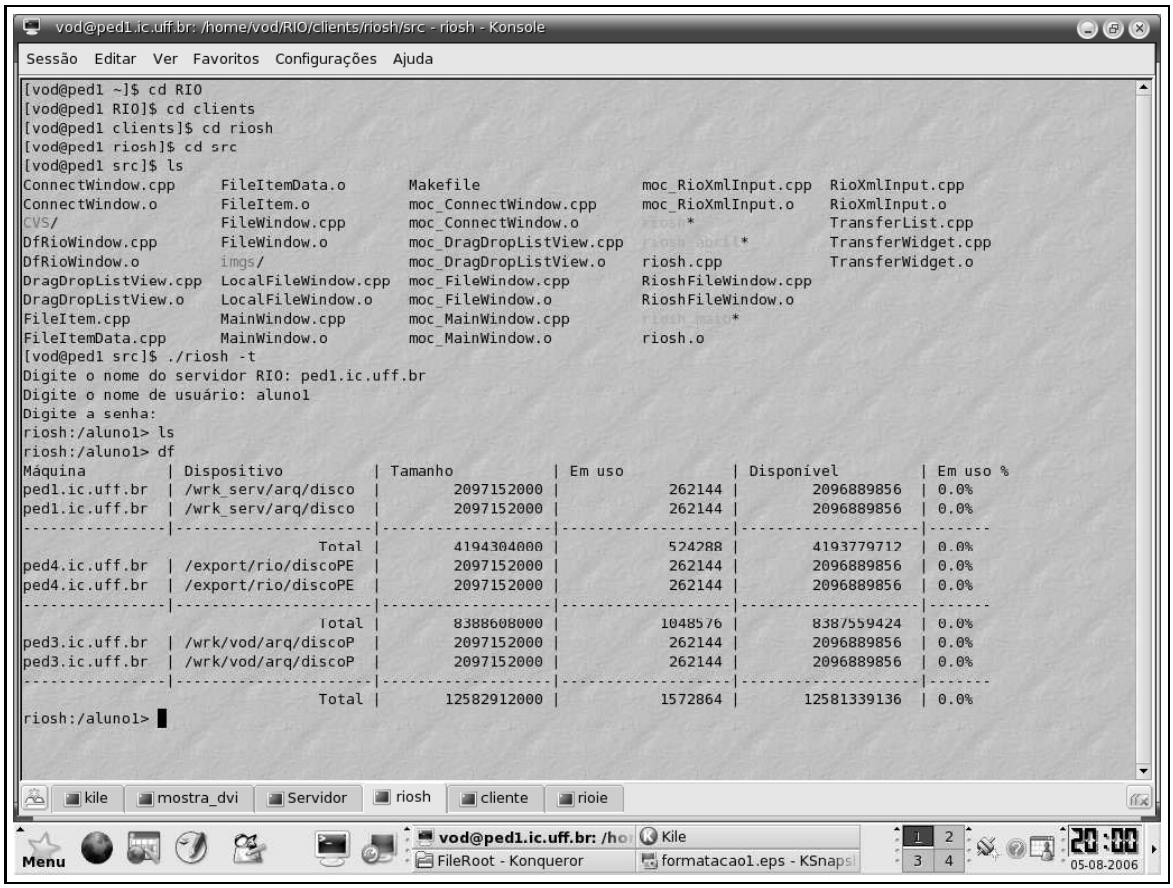

Figura 5.21: Taxa de ocupação do conjunto de discos do Nó de Armazenamento após formatação

Na Figura 5.21 é mostrada a taxa de ocupação do conjunto de discos dos Nós de Armazenamento, quando nenhum vídeo foi gravado.

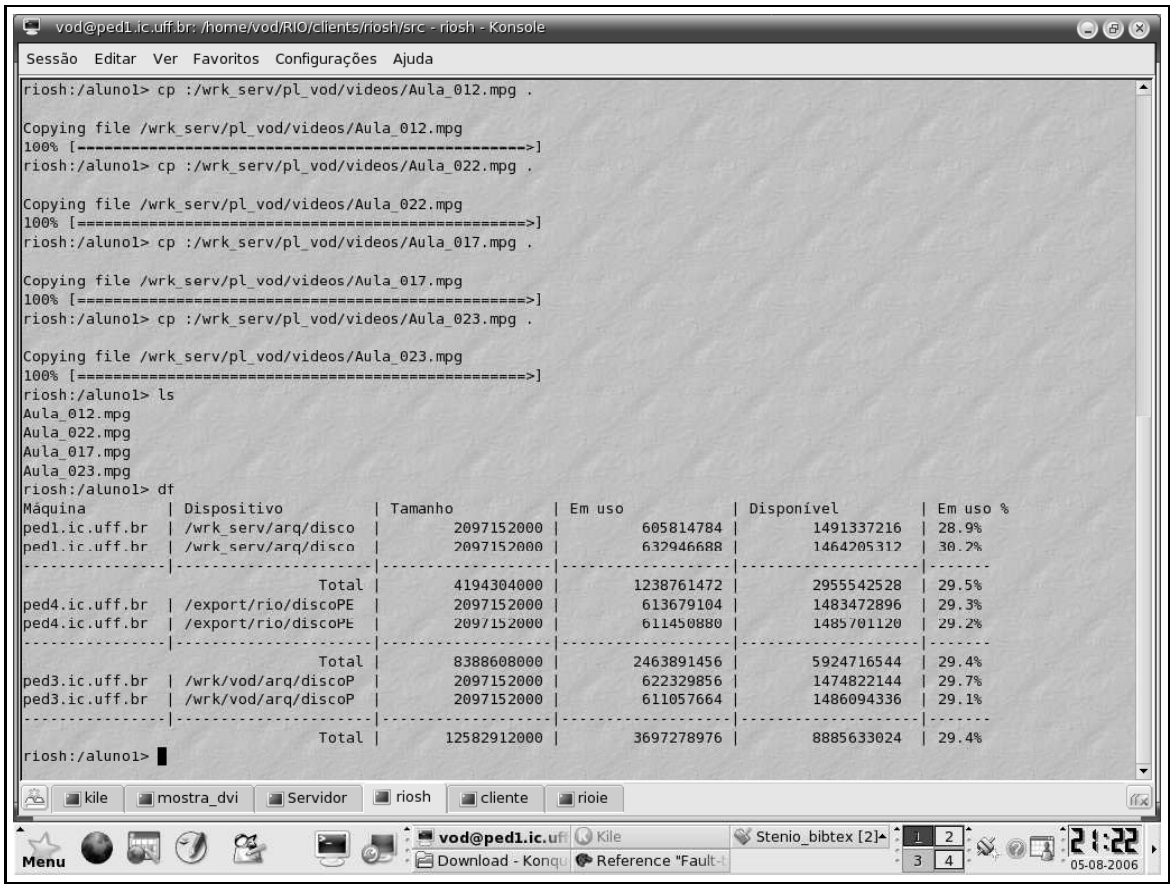

Figura 5.22: Taxa de ocupação do conjunto de discos do Nó de Armazenamento após inserção de vídeos

Na Figura 5.22 é mostrado a taxa de ocupação do conjunto de discos dos Nós de Armazenamento, após a inserção das quatro vídeo-aulas definidas para a terceira bateria. Foi verificado nesta Figura, uma taxa média de ocupação de aproximadamente 29,4% para o conjunto de Nós de Armazenamento.

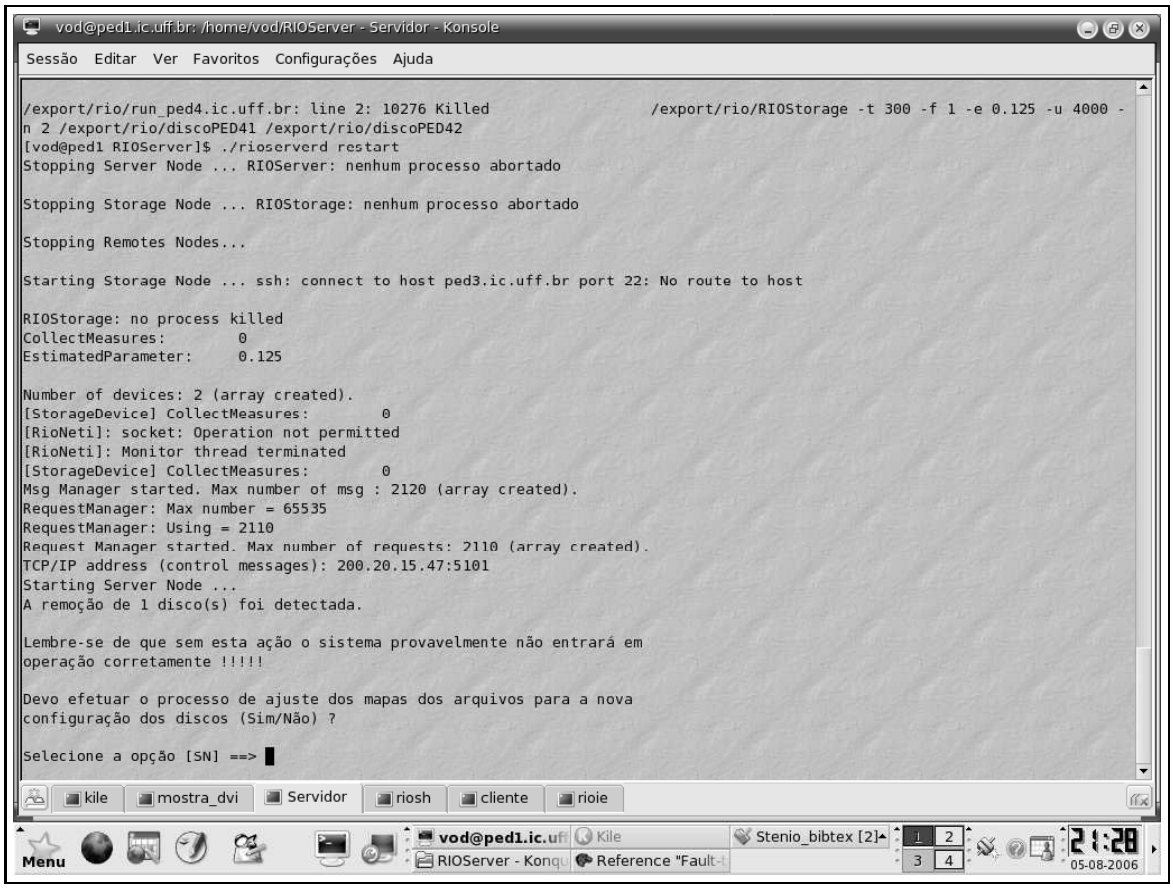

Figura 5.23: Rotina "*rioie*" detecta a remoção de um Nó de Armazenamento

Na Figura 5.23, a rotina "*rioie*" detecta a remoção de um Nó de Armazenamento e solicita autorização do operador para atualizar os metadados dos objetos multimídia armazenados nos Nós de Armazenamento.

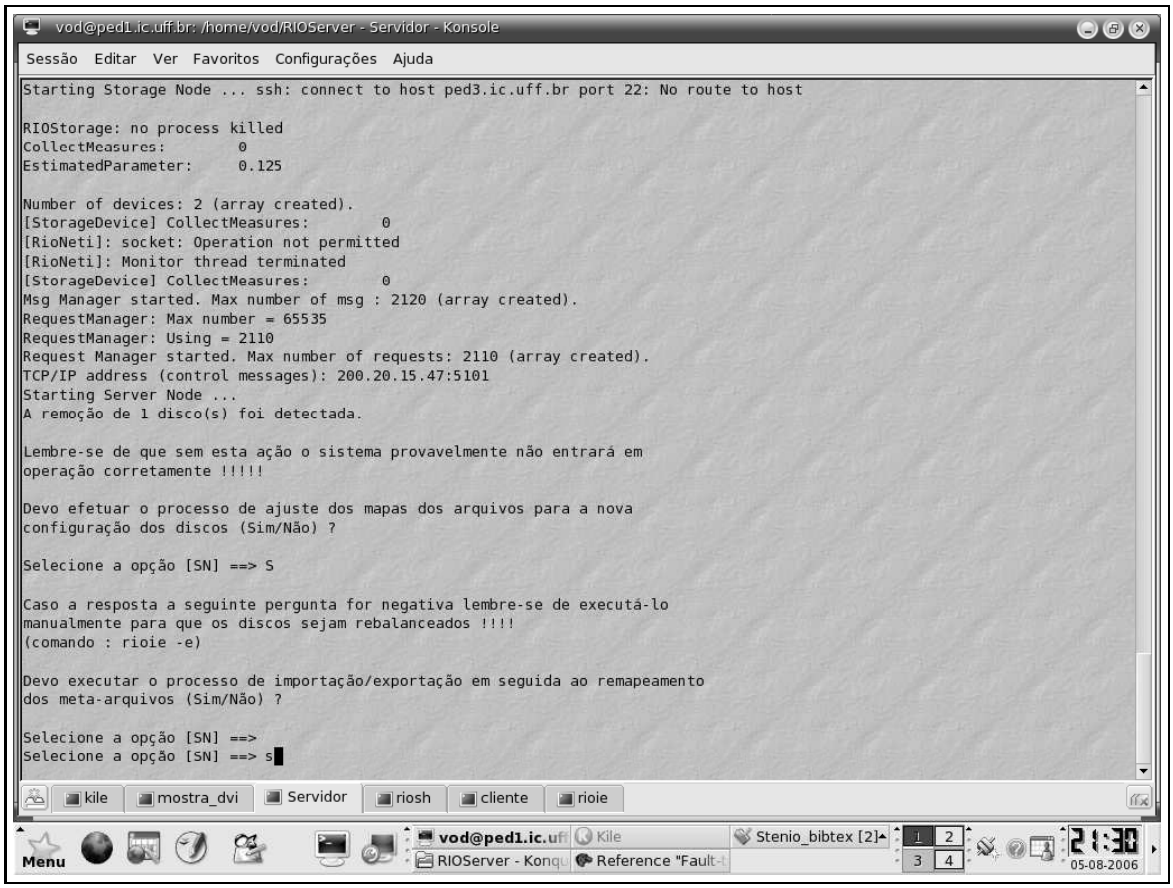

Figura 5.24: Solicitação para início da redistribuição

Na Figura 5.24, a rotina "*rioie*" solicita autorização do operador para iniciar a redistribuição randômica dos blocos dos objetos multimídia nos Nós de Armazenamento na nova configuração dos nós.

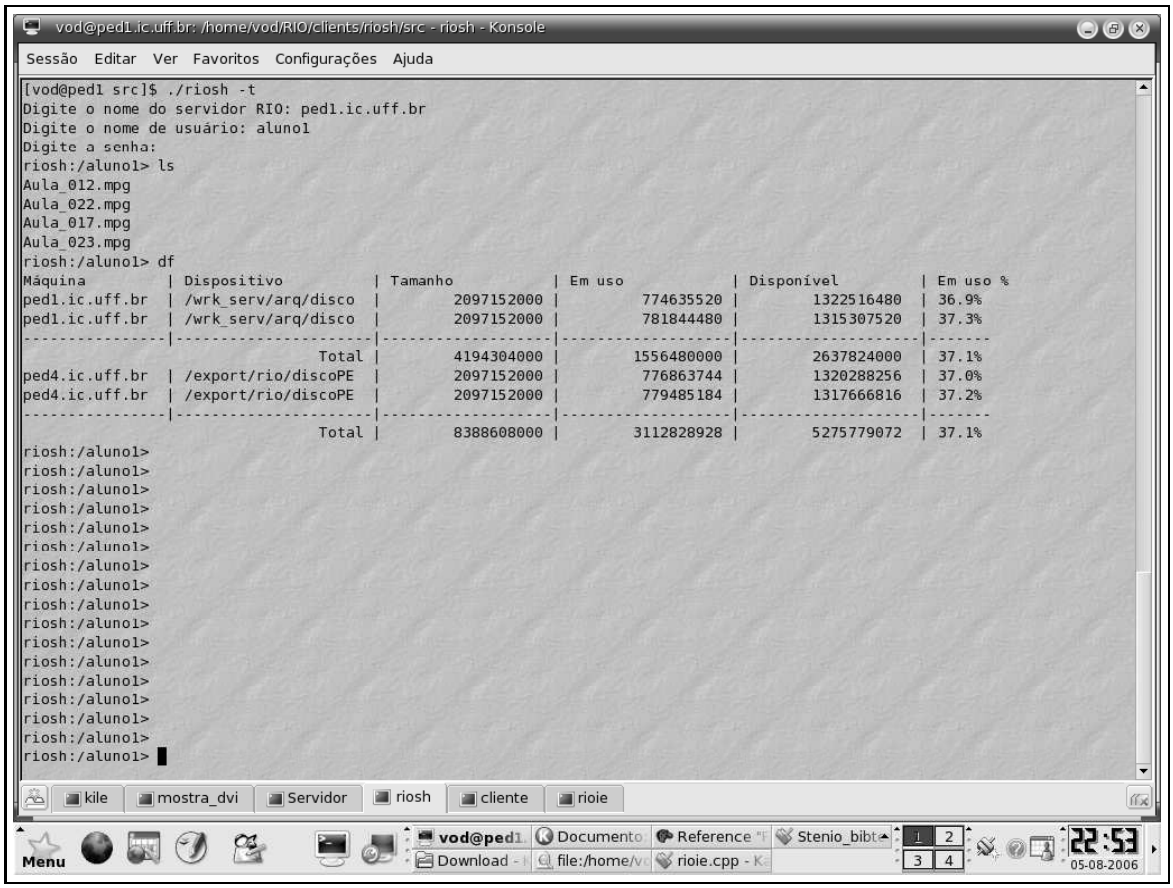

Figura 5.25: Taxa de ocupação do conjunto de discos do Nó de Armazenamento após a redistribuição

Após a redistribuição ter sido completada, podemos observar a taxa de ocupação dos discos pela Figura 5.25 no conjunto de Nós de Armazenamento que ficou em 37,1% aproximandamente.

De modo análogo, ao que foi realizado na primeira bateria, na inserção do Nó de Armazenamento foi utilizado o cenário resultante da experiência anterior, e foi reconectada a máquina PED3. O Servidor RIO foi interrompido para fazer as alterações necessárias nos arquivos de configuração. Em seguida, após a rotina "*rioie*" ter detectado a modificação da configuração dos Nós de Armazenamento e mediante autorização concedida pelo operador, foi iniciada a redistribuição dos blocos dos objetos multimídia nos Nós de Armazenamento e reiniciado o Servidor RIO.

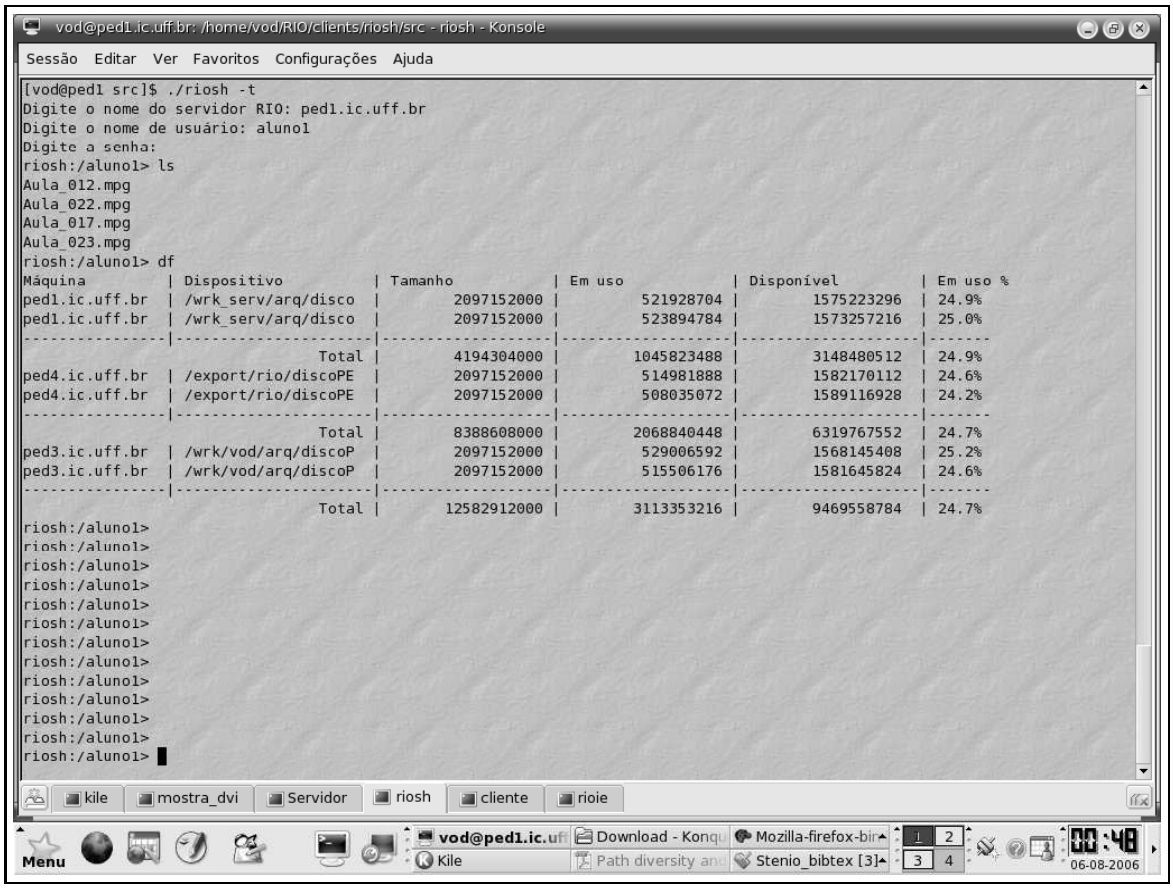

Figura 5.26: Taxa de ocupação do conjunto de discos do Nó de Armazenamento após a redistribuição

Após a redistribuição podemos verificar na Figura 5.26 que a taxa de ocupação do conjunto de discos dos Nós de Armazenamento do Servidor RIO, após inserção do Nó de Armazenamento ficou em aproximadamente 24,7%.

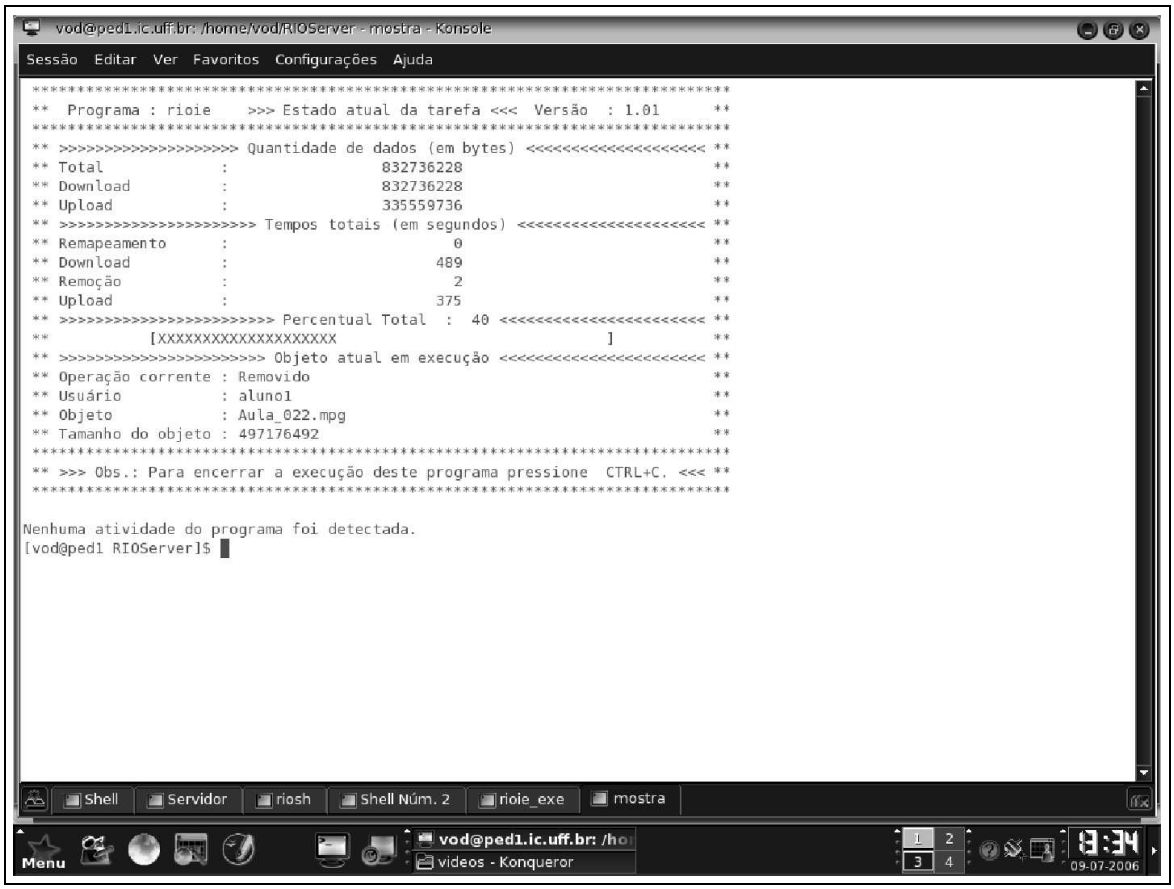

Figura 5.27: teste console 2

Na Figura 5.27 temos um exemplo de acompanhamento em tempo real do que a rotina "*rioie*" está realizando nos objetos multimídia armazenados nos Nós de Armazenamento, atualizando as informações na tela a cada 30 segundos.

A seguir temos o tempo apurado (segundos) das tarefas de redistribuição para cada objeto realizado pela rotina "*rioie*" durante a remoção de um Nó de Armazenamento. (Tabelas 5.9 e 5.10). Em seguida a apuração global de toda execução da rotina "*rioie*" (Tabela 5.11).

Tabela 5.9: Apurações dos tempos da redistribuição na remoção do Nó de Armazenamento

Estado	Usuário	Objeto	Tamanho	Downloaded	Download Time	Upload Time
Terminado	aluno1	Aula_012.mpg	335559736	335559736	73,755	88,397
Terminado	aluno1	Aula_017.mpg	533364736	533364736	121,842	143,322
Terminado	aluno1	Aula_022.mpg	497176492	497176492	113,863	132,953
Terminado	aluno1	Aula_023.mpg	481298620	481298620	111,372	131,128

Tabela 5.10: Apurações dos tempos da redistribuição na remoção do Nó de Armazenamento

Objeto	Removal Time	Map Time	Núm.Réplicas	Núm.Blocos	Bloqueado(0/1)
Aula_012.mpg	1,628	0,003	2	2561	0
Aula_017.mpg	1,710	0,003	2	4070	0
Aula_022.mpg	1,749	0,003	2	3794	0
Aula_023.mpg	1,744	0,003	2	3673	0

Tabela 5.11: Apurações global da redistribuição na remoção do Nó de Armazenamento

Tam. da base	Upl Time	Down Time	Rem Time	Map Time
1847399584	495,800	420,832	6,831	0,012

A seguir temos o tempo apurado (segundos) das tarefas de redistribuição para cada objeto realizado pela rotina "*rioie*" durante a inserção de um Nó de Armazenamento. (Tabelas 5.12 e 5.13). Em seguida a apuração global de toda execução da rotina "*rioie*" (Tabela 5.14).

Tabela 5.12: Apurações dos tempos da redistribuição na inserção do Nó de Armazenamento

Estado	Usuário	Objeto	Tamanho	Downloaded	Download Time	Upload Time
Terminado	aluno1	Aula_012.mpg	335559736	335559736	76,533	100,810
Terminado	aluno1	Aula_017.mpg	533364736	533364736	119,239	145,997
Terminado	aluno1	Aula_022.mpg	497176492	497176492	113,615	134,180
Terminado	aluno1	Aula_023.mpg	481298620	481298620	108,886	127,213

Tabela 5.13: Apuraçãos dos tempos da redistribuição na inserção do Nó de Armazenamento

Objeto	Removal Time	Map Time	Núm.Réplicas	Núm.Blocos	Bloqueado(0/1)
Aula_012.mpg	0,310	0,000	2	2561	0
Aula_017.mpg	0,244	0,000	2	4070	0
Aula_022.mpg	0,240	0,000	2	3794	0
Aula_023.mpg	0,245	0,000	2	3673	0

Tabela 5.14: Apuraçãos global da redistribuição na inserção do Nó de Armazenamento

Tam. da base	Upl Time	Down Time	Rem Time	Map Time
1847399584	508,200	418,273	1,039	0,000

5.2.3 Bateria 03

A ultima bateria, usou uma configuração de Nós de Armazenamento com dois discos de 2GB em cada nó, totalizando um espaço de 12 GB. A intenção dos testes desta bateria, foi avaliar o impacto nos tempos da redistribuição realizada pela rotina "*rioie*" enquanto vários clientes assistiam vídeos em outras unidades. Foram simuladas 5 (cinco) sessões de clientes no servidor como pode ser observado na Figura 5.28 e 5 (cinco) sessões clientes na máquina PED4. Foram usadas as seguintes vídeos-aulas: Aula_030 (57,5 MB), Aula_032 (67,0 MB), Aula_033 (57,5 MB) e Aula_034 (67,0 MB), Aula_042 (84,5 MB), Aula_044 (84,5 MB), Aula_046 (84,5 MB) e Aula_048 (84,5 MB), totalizando 587,0 MB. Foi fixado um número de réplicas igual a dois. Analogamente, foram feitos testes semelhantes, com esta nova base de dados, obtendo os resultados que podem ser observados nas Figuras 5.29 e 5.30

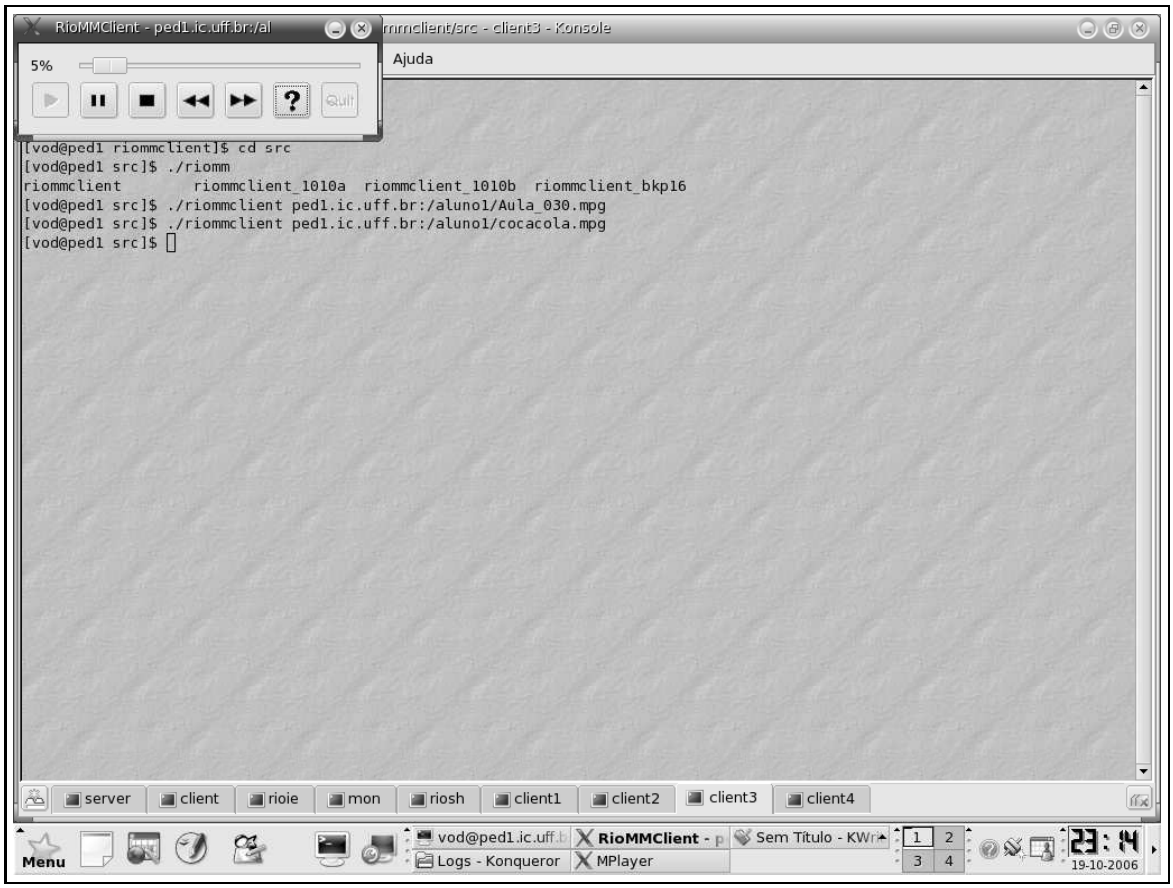

Figura 5.28: Execução de exibição de vídeos durante a rotina de redistribuição

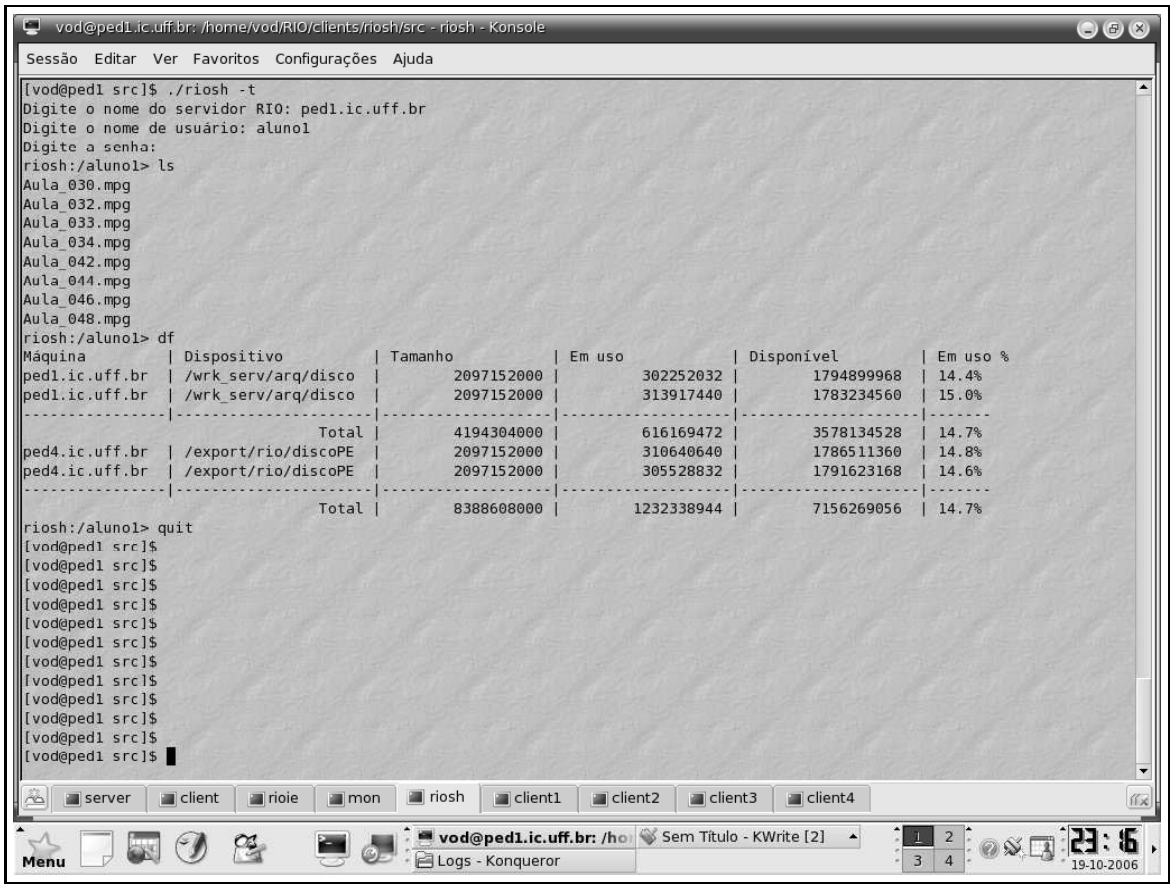

Figura 5.29: Taxa de ocupação dos objetos na remoção do Nó de Armazenamento após a redistribuição

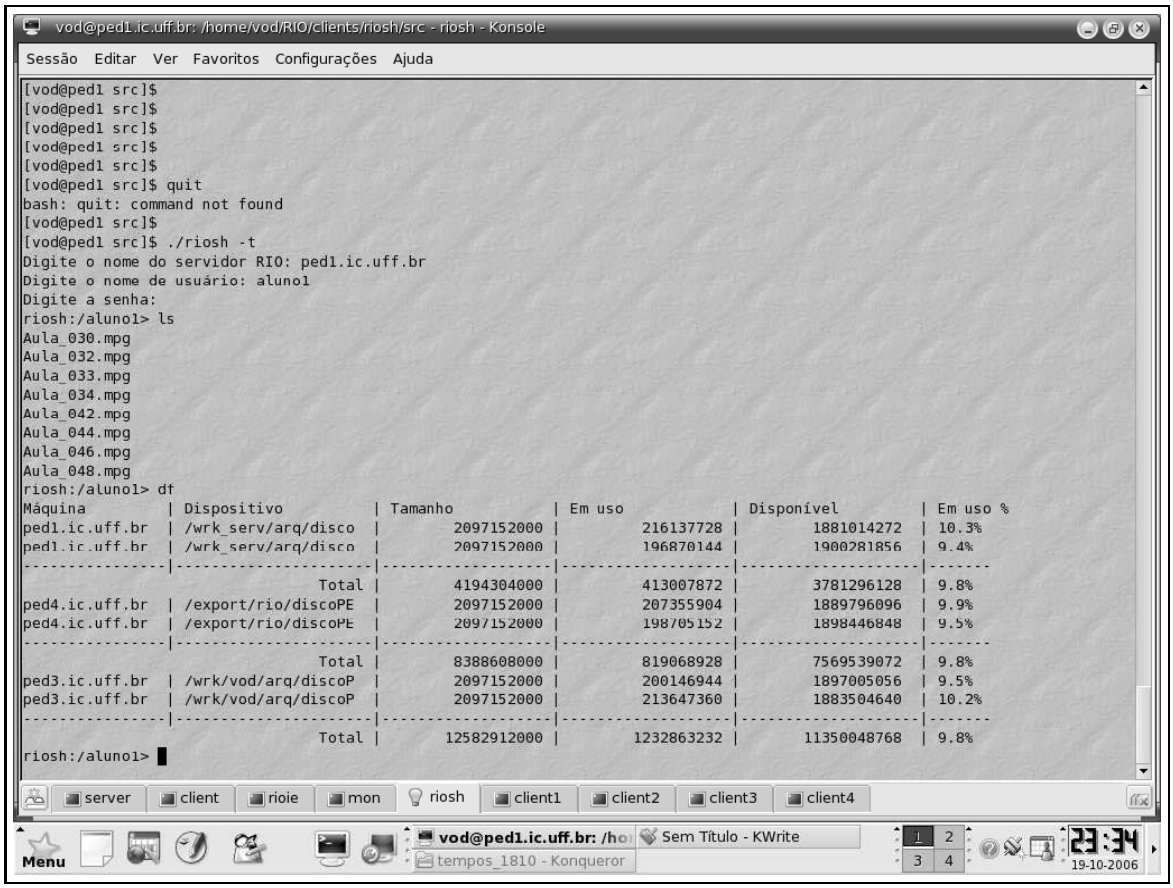

Figura 5.30: Taxa de ocupação dos objetos na inserção do Nó de Armazenamento após a redistribuição

Os tempos apurados (segundos com precisão de milésimos de segundos) das tarefas de redistribuição para cada objeto realizado pela rotina "*rioie*" durante a remoção de um Nó de Armazenamento conforme pode ser visto nas Tabela 5.15 e 5.16. Em seguida a apuração global de toda execução da rotina "*rioie*" pode ser vista na Tabela 5.17.

Tabela 5.15: Apurações dos tempos da redistribuição na remoção do Nó de Armazenamento

Estado	Usuário	Objeto	Tamanho	Downloaded	Download Time	Upload Time
Terminado	aluno1	Aula_030.mpg	60241080	60241080	14,463	16,173
Terminado	aluno1	Aula_032.mpg	70221146	70221146	16,294	17,912
Terminado	aluno1	Aula_033.mpg	60241080	60241080	14,997	14,833
Terminado	aluno1	Aula_034.mpg	70221146	70221146	16,664	16,718
Terminado	aluno1	Aula_042.mpg	88567286	88567286	21,944	22,102
Terminado	aluno1	Aula_044.mpg	88567286	88567286	22,454	20,829
Terminado	aluno1	Aula_046.mpg	88567286	88567286	22,304	21,088
Terminado	aluno1	Aula_048.mpg	88567286	88567286	23,469	20,667

A seguir temos os tempos apurados das tarefas de redistribuição para cada objeto realizado pela rotina "*rioie*" durante a inserção de um Nó de Armazenamento. (Tabelas 5.18 e 5.19). Em seguida a apuração global de toda execução da rotina "*rioie*" (Tabela 5.20).

Tabela 5.16: Apurações dos tempos(2) da redistribuição na remoção do Nó de Armazenamento

Objeto	Removal Time	Map Time	Núm.Réplicas	Núm.Blocos	Bloqueado(0/1)
Aula_030.mpg	2,081	0,003	2	460	0
Aula_032.mpg	0,092	0,003	2	536	0
Aula_033.mpg	0,082	0,003	2	460	0
Aula_034.mpg	0,110	0,002	2	536	0
Aula_042.mpg	0,120	0,002	2	676	0
Aula_044.mpg	0,115	0,003	2	676	0
Aula_046.mpg	0,126	0,003	2	676	0
Aula_048.mpg	0,116	0,003	2	676	0

Tabela 5.17: Apurações global da redistribuição na remoção do Nó de Armazenamento

Tamanho da base	Upload Time	Download Time	Removal Time	Map Time
615193596	150,322	152,589	2,842	0,022

Tabela 5.18: Apurações dos tempos da redistribuição na inserção do Nó de Armazenamento

Estado	Usuário	Objeto	Tamanho	Downloaded	Download Time	Upload Time
Terminado	aluno1	Aula_030.mpg	60241080	60241080	13,485	13,976
Terminado	aluno1	Aula_032.mpg	70221146	70221146	15,721	15,240
Terminado	aluno1	Aula_033.mpg	60241080	60241080	15,258	14,330
Terminado	aluno1	Aula_034.mpg	70221146	70221146	16,648	16,760
Terminado	aluno1	Aula_042.mpg	88567286	88567286	20,705	21,281
Terminado	aluno1	Aula_044.mpg	88567286	88567286	21,854	21,196
Terminado	aluno1	Aula_046.mpg	88567286	88567286	22,412	21,365
Terminado	aluno1	Aula_048.mpg	88567286	88567286	22,156	21,662

Tabela 5.19: Apurações dos tempos(2) da redistribuição na inserção do Nó de Armazenamento

Objeto	Removal Time	Map Time	Núm.Réplicas	Núm.Blocos	Bloqueado(0/1)
Aula_030.mpg	0,057	0,000	2	460	0
Aula_032.mpg	0,063	0,000	2	536	0
Aula_033.mpg	0,048	0,000	2	460	0
Aula_034.mpg	1,068	0,000	2	536	0
Aula_042.mpg	0,076	0,000	2	676	0
Aula_044.mpg	0,061	0,000	2	676	0
Aula_046.mpg	0,093	0,000	2	676	0
Aula_048.mpg	1,069	0,000	2	676	0

Tabela 5.20: Apurações global da redistribuição na inserção do Nó de Armazenamento

Tamanho da base	Upload Time	Download Time	Removal Time	Map Time
615193596	145,810	148,239	2,535	0,000

Capítulo 6

Conclusão

O fornecimento de serviços multimídia, exige servidores com características, que atendam aos requesitos rigorosos de limites para *jitter*, perdas de dados e retardo para garantir a QoS necessária às aplicações. Apesar de todos os avanços tecnológicos, a Internet ainda não garante todos os requisitos para que uma aplicação multimídia funcione com QoS exigida, de modo que são criadas técnicas para suprir estas necessidades.

Em sistemas de tempo real, a unidade de armazenamento é um ponto crítico do projeto. O seu perfeito dimensionamento não é o único fator a ser considerado. Outros fatores têm influência no desempenho e na confiabilidade do servidor multimídia: os algoritmos de distribuição dos dados nas unidades de armazenamento influênciam o tempo de armazenamento e de acesso; e os mecanismos de tolerância a falhas e as políticas de redundância de dados atenuam o impacto das falhas da unidade de armazenamento.

Como visto no Capítulo 2, são desenvolvidas várias soluções para atender as diversas situações que criam obstáculos para o perfeito funcionamento das aplicações multimídia através da Internet.

Um típico sistema multimídia distribuído é composto de um Nó Servidor, vários Nós de Armazenamento e vários Nós Clientes. Os discos presentes nos Nós de Armazenamento compõem o espaço de armazenamento do servidor multimídia. O disco do servidor ajuda a compor o espaço de armazenamento do servidor multimídia. O algoritmo de distribuição dos objetos multimídia no espaço de armazenamento do servidor multimídia, pode variar entre os modelos *Data Striping* e *Random Data Allocation*. O objeto multimídia a ser gravado, é dividido em blocos de mesmo tamanho e este tamanho corresponde ao que foi definido nos discos que compõe o espaço de armazenamento do servidor multimídia. Com a finalidade de aumentar o desempenho e a tolerância a falhas, podem ser acrescentados mecanismos de redundância para armazenamento dos objetos multimídia, isto é, podem

ser gravados várias cópias do mesmo bloco para cada objeto multimídia, em unidades de discos diferentes. Podem ser implementadas, cópias de alguns blocos para cada objeto multimídia , ou cópias de todos os blocos para cada objeto multimídia.

O Servidor Multimídia RIO, é um servidor multimídia distribuído que permite acesso simultâneo aos seus Nós de Armazenamento. O algoritmo de distribuíção utilizado é o *Random Data Allocation* (alocação randômica de dados). O Servidor RIO, implementa o mecanismo de redundância, distribuindo cópias de todos os blocos do objeto multimídia nos Nós de Armazenamento. No RIO, o mecanismo de redundância é definido como réplica. Quando o número de réplicas é igual a um, significa que somente foram gravados blocos originais do objeto multimídia nos Nós de Armazenamento. Quando o número de réplicas, por exemplo for igual a três, significa que para cada bloco original do objeto multimídia, existem outras duas cópias do mesmo bloco gravadas em Nós de Armazenamento diferentes. A existência de redundância, isto é de réplicas de objetos multimídia no Servidor RIO, ameniza o impacto no acesso aos vídeos pelos clientes, das eventuais falhas de Nós de Armazenamento, visto que oferece ao servidor Nós de Armazenamento alternativos para o acesso ao blocos de vídeos.

Atualmente uma eventual remoção ou falha de conexão de um Nó de Armazenamento pode causar falhas ou interrupção na exibição dos objetos multimídia nos Nós Clientes, dependendo do número de réplicas definido nos arquivos de configuração do Servidor RIO, visto que um maior número de réplicas, diminuirá a probabilidade de roteamento para o Nó de Armazenamento no qual foi detectada a falha. Por outro lado, a inserção de Nós de Armazenamento com a finalidade de aumentar o espaço de armazenamento não é detectada pelo Servidor RIO. É necessária então a parada do RIO. Após a parada do servidor são atualizados os arquivos de configuração e é feita a remoção de todos os objetos multimídia. Após esta operação, para garantir uma redistribuição randômica no novo cenário de discos do servidor, é feita a inserção de todos esses objetos multimídia removidos, deixando os clientes sem atendimento durante todo esse processo.

A solução adotada neste trabalho para redistribuir os objetos multimídia, quando ocorre uma modificação na configuração dos Nós de Armazenamento é composta por três procedimentos básicos. A interrupção do Servidor RIO, a atualização *off-line* dos arquivos de configuração, e a reinicialização do Servidor RIO em paralelo com uma rotina de redistribuição, que gradativamente redistribui randomicamente os objetos multimídia no novo cenário de Nós de Armazenamento, ao mesmo tempo que os clientes são atendidos.

A rotina de redistribuição randômica denominada "rioie" é uma rotina parametrizada

que permite o acompanhamento da evolução da tarefa de redistribuiçãso dos blocos dos objetos multimídia em tempo real, registro em arquivos logs do número de blocos movimentados, bem como da apuração do tempo gasto para realizar a redistribuição randômica dos objetos multimídia nos Nós de Armazenamento. Realizamos experimentos para validar a correta distribuição dos blocos dos objetos multmídia nos Nós de Armazenamento bem como os tempos consumidos para execução de toda a tarefa.

Através desta solução, é possível após breve interrupção do Servidor RIO, retomar o fornecimento dos objetos multimídia pelo Servidor RIO aos seus clientes. Como a prioridade maior é a continuídade dos serviços do Servidor RIO, a tarefa de redistribuição dos objetos multimídia nos Nós de Armazenamento, tem seu impacto atenuado no desempenho do Servidor RIO, uma vez que a rotina "rioie" redistribui um objeto multimídia de cada vez.

6.1 Trabalhos Futuros

Como sugestão para otimização da redistribuição, é importante o estudo da mudança de estrutura do Servidor RIO, para que se tenha uma área temporária para armazenamento de objetos multimídia que estejam sendo redistribuídos com possibilidade de várias réplicas. A idéia se baseia em se criar uma área constituída de Nós de Armazenamentos reservados (Figura 6.1) somente para o tratamento da remoção de um objeto multimídia. O procedimento a ser adotado quando a rotina "*rioie*" fosse tratar a redistribuição de um determinado objeto multimídia seria o seguinte:

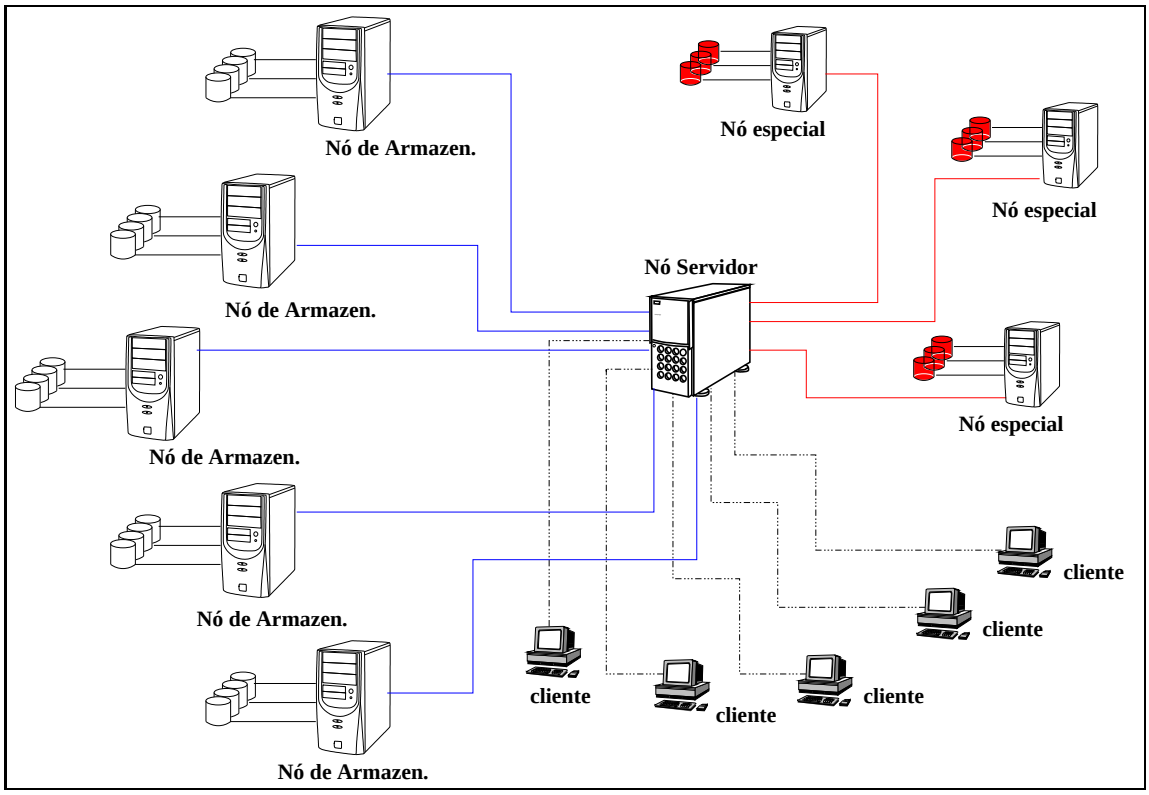

Figura 6.1: Estrutura com Nós Reservados (especiais)

1. Além da cópia de segurança do objeto multimídia feita pela "*rioie*", utilizando o algoritmo de distribuição randômica do RIO, a "*rioie*" iria inserir este objeto multimídia nestes dois Nós de Armazenamento reservados, usando uma versão modificada da rotina de inserir objetos multimídia do Servidor RIO;

2. Indicar em uma variável "X" que este objeto multimídia está nestes dois Nós de Armazenamento reservados;

3. Então a rotina "*rioie*" prossegue as suas tarefas de remoção e inserção do objeto multimídia nos Nós de Armazenamento do Servidor RIO;

4. Nesse intervalo, quando o Servidor RIO, recebesse um solicitação de exibição de um vídeo, seria analisado a condição do objeto multimídia:

   4.1. Se o objeto multimídia requisitado for o mesmo contido na variável "X" o Roteador irá direcionar as solicitações de blocos para estes Nós de Armazenamentos reservados, usando um versão modificada do Roteador do Servidor RIO, conforme visto na Figura 6.3;

   4.2. Senão, o Roteador irá direcionar as solicitações de blocos para os Nós de Armazenamentos do Servidor RIO, como normalmente é feito conforme visto na

Figura 6.2;

5. Ao final da inserção do objeto multimídia nos Nós de Armazenamento do Servidor RIO, a rotina "*rioie*" limparia o valor da variável "X" ;

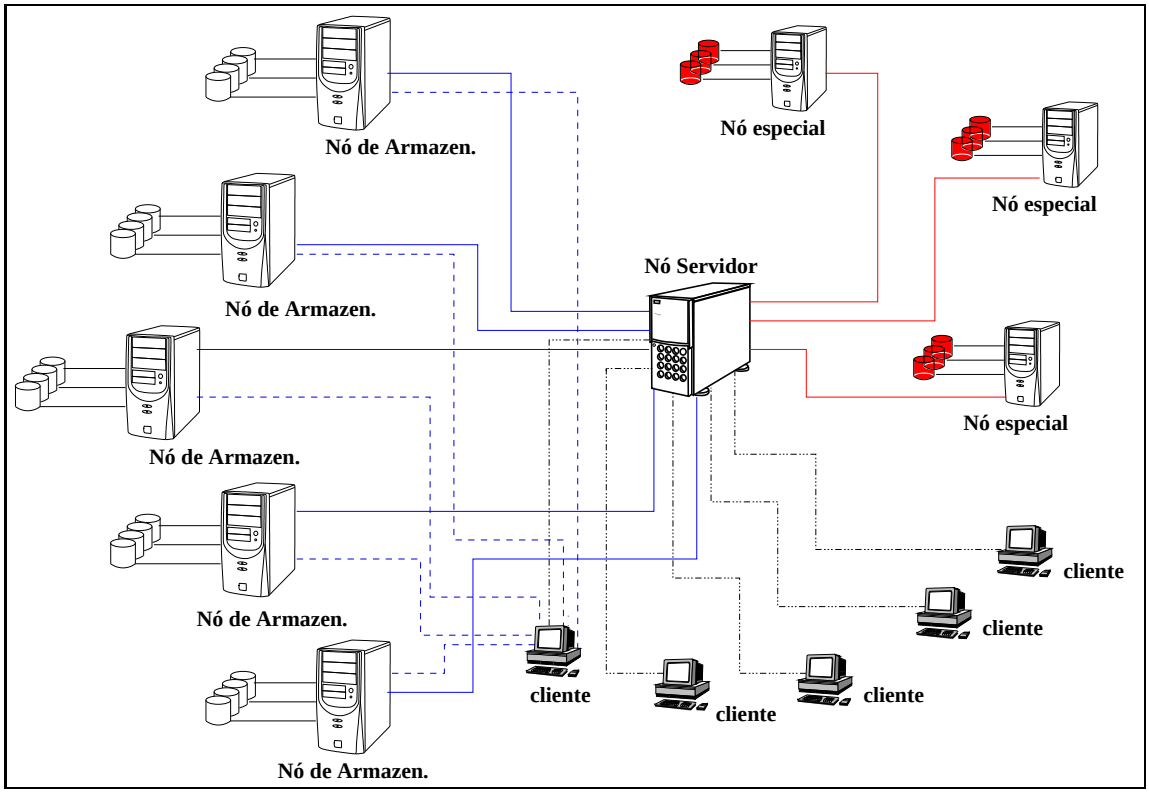

Figura 6.2: Acesso aos Nós de Armazenamento convencionais

Dessa maneira, como exposto acima, a remoção de um objeto multimídia, não afetaria de modo algum ao funcionamento do Servidor RIO.

Para a detecção automática da modificação de configuração dos Nós de Armazenamento, pode ser idealizado um módulo no servidor RIO, que monitore todas as conexões dos Nós de Armazenamento quanto as possíveis falhas e também possua uma interface de comunicação para aceitar a inserção de novo Nó de Armazenamento, identificando-o para o Servidor RIO, para que se possa fazer uma transição das solicitações em andamento para a nova configuração e depois realizar a redistribuição do objetos multimídia nos Nós de Armazenamento, sem que se tenha de interromper o Servidor RIO.

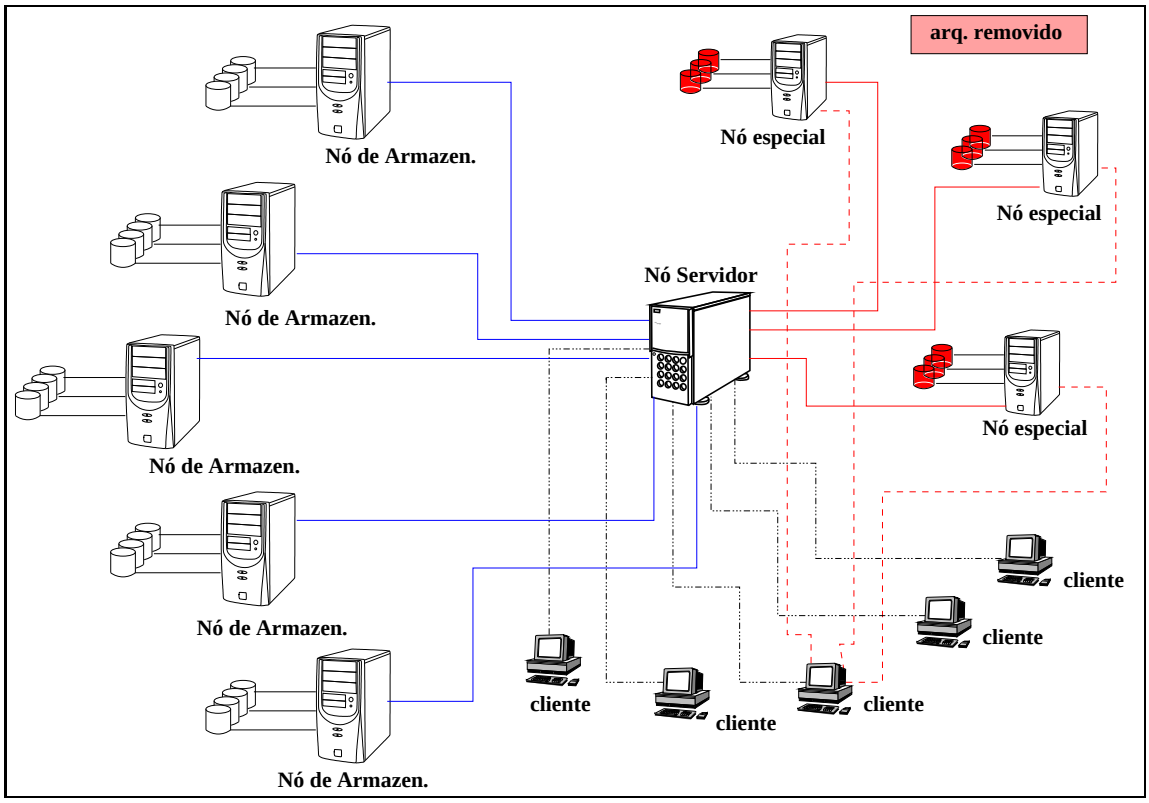

Figura 6.3: Acesso aos Nós de Armazenamento reservado (especiais)

E finalmente, avaliar a possibilidade de melhorar o desempenho da rotina "*rioie*" e medir o impacto causado no Servidor RIO em funcionamento.

APÊNDICE A – Pseudo-código da rotina "rioie"

Neste anexo o pseudo-código da rotina "*rioie*" e algumas das tabelas já mostradas na Seção 4.1 nas quais estão definidos parâmetros, variáveis, mensagens de erro, etc, necessários ao leitor para o entendimento do pseudo-código hora mostrado.

Como vimos no Capítulo 4, a execução do RIO utiliza parâmetros que norteiam sua execução. Do mesmo modo a execução da rotina "*rioie*" utiliza parâmetros para executar as ações necessárias indicadas pelo operador. Na Tabela A.1 a primeira coluna contém o nome da variável usada no código fonte que define a ação a ser executada pela "*rioie*". Na segunda coluna estão indicados os valores atribuídos no código fonte às variáveis da primeira coluna, quando um parâmetro da terceira coluna é fornecido pelo operador, e finalmente na quarta coluna temos a descrição de cada ação implementada no "*rioie*" .

Tabela A.1: Opções das linhas de comando da rotina *rioie*

Variável	**C**	**P**	**Significado**
IE_OPT_INVALID	0		Opção inválida
IE_OPT_HELP	1	-h	Obtém ajuda
IE_OPT_EXECUTE	2	-e	Executa o Export-Import
IE_OPT_CLEAR	3	-c	Limpa a ultima execução
IE_OPT_STATUS	4	-s	Retorna em um arquivo o estado da execução
IE_OPT_LOCK	5	-l	Bloqueia os metafiles
IE_OPT_UNLOCK	6	-u	Desbloqueia os metafiles
IE_OPT_NEWMAP	7	-m	Constrói um novo mapeamento para os discos
IE_OPT_NPEXEC	8	-f	Executa as opções 7 e 2 em seqüência
IE_OPT_VERIFY	9	-v	Verifica se há algo a fazer no mapeamento dos discos
IE_OPT_REALTIME	10	-p	Acompanha em tempo real a execução da tarefa

As mensagens para o usuário emitidas pela rotina "*rioie*" estão descritas na Tabela A.2, enquanto as mensagens de erro estão descritas na Tabela A.3. Nas tabelas a primeira coluna contém a identificação usado no código fonte do *string* associado ao texto da segunda coluna. Nas duas tabelas, na segunda coluna, os caracteres %s são preenchidos por parâmetros determinados pela rotina "*rioie*" em tempo de execução.

Tabela A.2: Mensagens do sistema para o usuário

Identificação	**Significado**
IE_MESSAGE_01	Nenhuma modificação nos discos foi detectada.
IE_MESSAGE_02	A remoção de %s disco(s) foi detectada.
IE_MESSAGE_03	Deve-se efetuar o processo de ajuste dos mapas dos arquivos para a nova configuração dos discos (Sim/Não) ?
IE_MESSAGE_04	Foi detectada a inclusão de novo(s) disco(s) no sistema.
IE_MESSAGE_05	Lembre-se de que sem esta ação, o sistema provavelmente não entrará em operação corretamente !!
IE_MESSAGE_06	Caso a resposta a seguinte pergunta for negativa lembre-se de executá-lo manualmente para que os discos sejam rebalanceados !!!! (comando : %s -e)
IE_MESSAGE_07	Deve-se executar o processo de importação/exportação em seguida ao remapeamento dos meta-arquivos (Sim/Não) ?
IE_MESSAGE_08	Deseja iniciar o processo de rebalanceamento de carga dos arquivos do sistema agora (Sim/Não) ?
IE_MESSAGE_09	%s :
IE_MESSAGE_10	Versão (%s)
IE_MESSAGE_11	Uso: %s < -h|-e|-c|-s arquivo-de-saida|-l|-u-m|-f|-v|-p segundos > Opções:
IE_MESSAGE_12	-h : Apresenta este texto de ajuda e a versão do sistema.
IE_MESSAGE_13	-e : Executa a operação e exportação e importação dos objetos do servidor.
IE_MESSAGE_14	-c : Limpeza dos dados de exportação e importação do servidor.
IE_MESSAGE_15	-s : Gera um relatório no arquivo-de-saída com o estado atual do processo de exportação e importação do servidor.
IE_MESSAGE_16	-l : Efetua um lock lógico nos meta-files do servidor.
IE_MESSAGE_17	-u : Efetua um unlock lógico nos meta-files do servidor.
IE_MESSAGE_18	-m : Atualiza as informações dos meta-files para propiciar a operação do servidor enquanto o processo de importação e exportação dos objetos está em curso.
IE_MESSAGE_19	-f : executa as opções -m e -e em seqüência.
IE_MESSAGE_20	-v : Verifica se ocorreram mudanças no mapa dos discos e inicia as operações de manutenção necessárias através de interação com o operador do sistema.
IE_MESSAGE_21	-p : Acompanha, em tempo real, a execução da tarefa. segundos: opcional intervalo de refresh das informações ([30]<->300).
IE_MESSAGE_22	-t : Para o uso em rotinas de testes apenas.

Tabela A.3: Mensagens de erro do sistema para o usuário

Identificação	**Significado**
IE_MSG_ERR_01	O arquivo de origem da cópia (%s) não foi encontrado.
IE_MSG_ERR_02	Não foi possível criar o arquivo destino da cópia (%s).
IE_MSG_ERR_03	Um erro foi detectado durante a escrita no arquivo destino (%s)
IE_MSG_ERR_04	O diretório do executável (%s) não foi encontrado. Favor colocá-lo no PATH do usuário.
IE_MSG_ERR_05	O arquivo, %s, que contém os dados para a conexão com o servidor não foi encontrado.
IE_MSG_ERR_06	Não pude criar o arquivo de controle da execução concorrente (%s)
IE_MSG_ERR_07	Não consegui ler o arquivo temporário (%s).
IE_MSG_ERR_08	Não foi possível criar o arquivo temporário para a apresentação dos dados (%s)
IE_MSG_ERR_09	O arquivo de configuração do programa não foi encontrado (%s)
IE_MSG_ERR_10	O meta-arquivo %s não foi encontrado.
IE_MSG_ERR_11	Erro durante a leitura do meta-arquivo %s
IE_MSG_ERR_12	Erro durante a escrita no meta-arquivo %s.
IE_MSG_ERR_13	Favor executar antes a opção: %s -v
IE_MSG_ERR_14	O arquivo de configuração dos discos não foi encontrado (%s)
IE_MSG_ERR_15	Erro durante a leitura do arquivo de configuração do programa (%s).
IE_MSG_ERR_16	Erro durante a leitura do arquivo de configuração do programa.
IE_MSG_ERR_17	Erro durante a escrita no arquivo de configuração do programa.
IE_MSG_ERR_18	O objeto %s não pode ser removido.
IE_MSG_ERR_19	O objeto %s não existe.
IE_MSG_ERR_20	Permissão negada para remover o objeto %s.
IE_MSG_ERR_21	Erro durante o download do objeto.
IE_MSG_ERR_22	Erro durante o upload do objeto.
IE_MSG_ERR_23	A conexão com o servidor na máquina %s falhou.
IE_MSG_ERR_24	Versão inválida.
IE_MSG_ERR_25	O nome do usuário é inválido.
IE_MSG_ERR_26	O servidor não está respondendo.
IE_MSG_ERR_27	O número máximo de sessões foi alcançado.
IE_MSG_ERR_28	Erro inesperado.
IE_MSG_ERR_29	O tamanho do bloco utilizado pelo servidor não pode ser lido.
IE_MSG_ERR_30	Permissão negada para o objeto %s.
IE_MSG_ERR_31	Erro inesperado para o objeto %s.
IE_MSG_ERR_32	Este passo de preparação é necessário para o correto cumprimento desta tarefa.
IE_MSG_ERR_33	Opção inválida para o programa.
IE_MSG_ERR_34	Esta opção deve ser executada de forma exclusiva. Já existe um programa em execução.
IE_MSG_ERR_35	Erro na leitura do arquivo de configuração dos discos.
IE_MSG_ERR_36	Erro na criação do arquivo de configuração dos discos (%s). Favor retornar o backup deste arquivo diskcfg.bak.
IE_MSG_ERR_37	Nenhuma atividade do programa foi detectada.

Na Tabela A.4 temos os diferentes estados que podem ser assumidos pelos metadados dos objetos multimídia durante a redistribuição randômica. A primeira coluna contém o nome da variável usada no código fonte que define estado do metadado. Na segunda coluna estão indicados os valores atribuídos no código fonte às variáveis da primeira coluna, e finalmente na terceira coluna temos a descrição do estado do metadado.

Tabela A.4: Tabela dos estados possíveis para os objetos multimídia

Identificação	**Valor**	**Significado**
IE_STATE_WAI	0	Remapeamento não iniciado
IE_STATE_MNR	1	Remapeamento não necessário
IE_STATE_MPR	2	Remapeamento necessário
IE_STATE_MAP	3	Remapeamento executado
IE_STATE_DWL	4	Efetuando download
IE_STATE_DWD	5	Download efetuado
IE_STATE_DET	6	Removendo
IE_STATE_DEL	7	Removido
IE_STATE_UPL	8	Efetuando upload
IE_STATE_DON	9	Terminado

A rotina-principal (Figura A.1) é o ponto de partida da rotina "*rioie*". Com a captura dos argumentos passados ao programa pela linha de comando fornecida pelo script é feita a inicialização dos caminhos que identificam os diretórios dos arquivos de trabalho que são utilizados pela rotina. Em seguida, por meio do parâmetro contido no script é feita a seleção da sub-rotina a ser ativada (o parâmetro pode também ser fornecido pelo operador por meio de linha de comando). A seguir temos a relação das sub-rotinas do "*rioie*" acompanhadas de uma breve descrição. Os pseudo-códigos correspondentes estão contidos nas Figuras A.2 até A.29.

As sub-rotinas, as quais são definidos os pseudo-código são:

- remapear-metafiles - tornar consistente o metadado de cada objeto multimídia;
- atualizar-metafiles - preparar a correção dos ponteiros dos blocos dos metadados ;
- mudar-acesso-metafile - bloquear/desbloquear o metadado, regravando o *header*;
- mudar-mapa-metafile - regravar novo mapeamento do metadado;
- mudar-por-replica - substituir ponteiros inválidos por réplicas correspondente;
- preparar-mapa-blocos - marcar ponteiro do blocos com indicação para Nós removidos, como ponteiro inválido;

- bloquear-todos-metafiles - bloquear todos os metadados do Servidor RIO;
- desbloquear-todos-metafiles - desbloquear todos os metadados do Servidor RIO
- mudar-acesso-todos - através do cmetafile bloquear/desbloquear todos os metadados;
- verificar-modificacao - ler nova configuração de Nós de Armazenamento;
- verificar-discos - verificar mudança na configuração de Nós de Armazenamento ;
- iniciar-remapeamento - identificar remoção e em caso afirmativo ativar remapeamento e redistribuição do blocos;
- criar-listas - incializar os caminhos para as áreas de trabalho da rotina "*rioie*";
- mudar-status-metafiles - mudar o estado de todos os metadados do RIO;
- mudar-status - mudar o estado de um metadado;
- remapear-exportar-importar - atualizar todos os metadados e em seguida realizar a redistribuição dos blocos;
- executar-exportar-importar - realizar a redistribuição dos blocos;
- executar-download-upload - ativar o *download* e depois ativar *upload* para cada objeto multimídia;
- tratar-redistribuicao - de acordo com o estado do metadado ativar sub-rotina para preparar a redistribuição;
- ver-status-arquivo - de acordo com o estado corrente do metadado, bloquear ou desbloquear o metadado;
- atualizar-por-status - de acordo com o estado corrente do metadado, ativar sub-rotinas para realizar a redistribuição dos blocos do objeto multimídia correspondente;
- agir-STATE-DWL - fazer o *download* do objeto multimídia para a área de trabalho;
- agir-STATE-DET - realizar a remoção do objeto multimídia dos Nós de Armazenamento;
- agir-STATE-UPL - fazer o *upload* do objeto multimídia da área de trabalho;

- limpar-exportar-importar - desfazer uma redistribuição de blocos incompleta;
- limpar-download-upload - desfazer os procedimentos de *download* e *upload* realizado nos objetos multimídia;
- tratar-desmanchar - fazer tratamento por estado do metadado de cada objeto para desfazer a redistribuição de blocos realizada anteriormente;
- ver-status-desmanche - de acordo com o estado corrente do metadado, bloquear ou desbloquear o metadado para desfazer a redistribuição de blocos realizada anteriormente;
- atualizar-desmanche - de acordo com o estado corrente do metadado, ativar sub-rotinas para desfazer a redistribuição dos blocos do objeto multimídia correspondente;
- desmanchar-STATE-DWL - fazer o *download* do objeto multimídia para a área de trabalho com a finalidade de desfazer a redistribuição de blocos realizada anteriormente;
- desmanchar-STATE-DET - realizar a remoção do objeto multimídia dos Nós de Armazenamento com a finalidade de desfazer a redistribuição de blocos realizada anteriormente;
- desmanchar-STATE-UPL - fazer o *upload* do objeto multimídia da área de trabalho com a finalidade de desfazer a redistribuição de blocos realizada anteriormente;

São definidos um conjunto de arquivos e variáveis de trabalho para serem utilizados nas sub-rotinas. Temos a seguir relação e o significado das variáveis e arquivos importantes para a compreensão dos pseudos-códigos:

- Variável "lock" simbolizando bloquear um metafile;
- Variável "unlock" simbolizando desbloquear um metafile;
- Variável "codigo-acesso" que recebe lock ou unlock para ser passada como parâmetro;
- Variável "nome-metafile" recebe o nome do metafile corrente que está sendo manuseado;

- Variável "codigo-status" recebe os possíveis valores do status que o metafile pode ter;
- Arquivo "cmetafile" que contém a lista dos metadados dos objetos multimídia armazenados no Servidor RIO.

```
rotina-principal.
    definir variavel lock    // para  bloquear
    definir variavel unlock  // para  desbloquear
    definir cmetafile        // arquivo com a lista de
                             // todos os metafiles
    capturar parametro da console
    capturar_identifica caminhos_trabalhos
    Selecao (opcao)
      caso IE_OPT_INVALID:
           imprimir-mensagem-erro
      caso IE_OPT_HELP:
           imprimir-tela-mensagem-ajuda
      caso IE_OPT_CLEAR:
           limpar-exportar-importar
      caso IE_OPT_EXECUTE:
           executar-exportar-importar
      caso IE_OPT_STATUS:
           imprimir-tela-estado-execucao
      caso IE_OPT_REALTIME:
           imprimir-tempo-real-resultados
      caso IE_OPT_LOCK:
           bloquear-todos-metafiles
      caso IE_OPT_UNLOCK:
           desbloquear-todos-metafiles
      caso IE_OPT_NEWMAP:
           remapear-metafiles
      caso IE_OPT_NPEXEC:
           remapear-exportar-importar
      caso IE_OPT_VERIFY:
           verificar-modificacao
    fim-Seleção
```

Figura A.1: Rotina principal

A sub-rotina remapear-metafiles (Figura A.2), tem por finalidade tornar consistente o metadado de cada objeto multimídia, após a remoção do Nó de Armazenamento.

```
Remapear-metafiles
   verificar-execução simultânea
   verificar existencia de mapeamento em andamento
   ler configuração atual dos discos
   ler dados globais de execução
   atualizar-metafiles
   retornar a rotina-principal
```

Figura A.2: Remapear metadados

.

A sub-rotina atualizar-metafiles (Figura A.3) tem por finalidade processar a correção nos metadados, dos ponteiros para os blocos com indicação para discos que pertecem a Nó de Armazenamento que foi removido.

```
atualizar-metafiles
    inicializar-contadores-controles
    Se não houver mudanca configuração dos discos
       emitir mensagem-aviso para usuario
       retornar para remapear-metafiles
    fim-se
    Se não houver disco-removido
       emitir mensagem-aviso para usuario
       retornar para remapear-metafiles
    fim-se
    identificar caminho-completo para o arquivo-cmetafile
    qtd_reg = calcular-número-de-objetos-metafile
    abrir cmetafile
    inicializar contabilizacao-tempo
    ind = 0
    enquanto ind menor-igual a qtd_reg
         incrementar ind
          ler registro-metafile
         mudar-acesso-metafile (caminho-completo,lock)
         mudar-mapa-metafile
         apurar-tempos-locais-globais
         mudar-acesso-metafile (caminho-completo,unlock)
    fim-enquanto
    fechar cmetafile
    retornar para remapear-metafiles
```

Figura A.3: Atualizar os metadados

.

A sub-rotina mudar-acesso-metafile (Figura A.4), tem por finalidade bloquear ou desbloquear o metadado, regravando apenas o *header* do metadado.

```
mudar-acesso-metafile
    abrir metafile
    posicionar primeiro-registro
    modificar-acesso-indicado
    regravar metafile
    fechar metafile
    retornar para ultima-chamada   // varias sub-rotinas chama esta função
```

Figura A.4: Modificar o acesso à um metadado

A sub-rotina mudar-mapa-metafile (Figura A.5), tem por finalidade corrigir nos metadados dos objetos multimídia, os ponteiros para os blocos com indicação para discos que pertecem a Nó de Armazenamento que foi removido.

```
mudar-mapa-metafile
   definir i //controlador-loop
   abrir metafile
   ler primeiro-registro-metafile
   Se numero-replicas menor que 2
      retornar para atualizar-metafiles
   fim-se
   alocar estrutura para armazenar blocos
   ler blocos
   posicionar inicio-blocos
   enquanto i < numero-blocos
       mudar-por-replica
   fim-enquanto
   regravar novo-mapeamento
   fechar metafile
   retornar para atualizar-metafiles                                    ▫
```

Figura A.5: Mudar mapeamento do metadado

A sub-rotina mudar-por-replica (Figura A.6), tem por finalidade identificar ponteiros para blocos inválidos e trocá-los por ponteiros para blocos válidos.

```
mudar-por-replica
   alocar vetor validacao-blocos
   preparar-mapa-blocos
   acessar primeiro-bloco-valido
   se nao encontrar-bloco-valido
      retornar para mudar-mapa-metafile
   fim-se
   enquanto existir blocos-invalidos
      trocar blocos-invalidos por blocos-validos
   fim-enquanto
   liberar memoria-alocada
   retornar para mudar-mapa-metafile                                    ▫
```

Figura A.6: Troca de ponteiro de blocos inconsistentes por réplicas

.

A sub-rotina preparar-mapa-bloco (Figura A.7), tem por finalidade marcar os ponteiros para os blocos com indicação para discos que pertecem a Nó de Armazenamento que foi removido, como ponteiros para blocos inválidos.

```
preparar-mapa-blocos
   inicializar ponteiros
   enquanto houver blocos
       se disco indicado no bloco foi removido
          marcar bloco-invalido
       senao
          marcar bloco-valido
       fim-se
   fim-enquanto
   retornar para mudar-por-replica
```

Figura A.7: Marcar ponteiros de blocos inconsistentes no metadado

A sub-rotina bloquear-todos-metafiles e a sub-rotina desbloquear-todos-metafiles (Figura A.8), são semelhantes e tem por finalidade bloquear e desbloquear todos os metadados do RIO respectivamente. A diferença entre as duas sub-rotinas está no parâmetro de acesso que é utilizado.

```
bloquear-todos-metafiles
   verificar execucao-simultanea
   mudar-acesso-todos(lock)
   retornar rotina-principal

desbloquear-todos-metafiles
   verificar execucao-simultanea
   mudar-acesso-todos(unlock)
   retornar rotina-principal
```

Figura A.8: Bloquear/desbloquear todos os metadados

A sub-rotina mudar-acesso-todos (Figura A.9), tem por finalidade bloquear ou desbloquear todos os metadados conforme código de acesso passado pela sub-rotina chamadora.

```
mudar-acesso-todos
   capturar caminho-completo para cmetafiles
   qtd_reg = calcular-número-de-objetos-metafile
   abrir cmetafile
   ind = 0
   enquanto ind menor-igual a qtd_reg
         incrementar ind
          ler registro-metafile
         mudar-acesso-metafile (caminho-completo,acesso-indicado)
    fim-enquanto
    fechar cmetafile
    retornar para bloquear-todos-metafiles
```

Figura A.9: Mudar acesso de todos os metadados

.

A sub-rotina verificar-modificacao (Figura A.10), tem por finalidade ler a configuração atual dos Nós de Armazenamento para verificar uma possível mudança de layout dos Nós de Armazenamento.

A sub-rotina verificar-disco, tem por finalidade verificar se houve movimentação, isto é, alguma mudança na configuração dos Nós de Armazenamento.

```
verificar-modificacao
   verificar execucao-simultanea
   ler configuracao atual dos discos
   ler dados-globais
   verificar-discos
   retornar para rotina-principal

verificar-discos
   verificar-movimentacao
   se não houver-movimentacao
      retornar para verificar-modificacao
   fim-se
   se houver disco-removido
      se  não autorizar
          retornar para verificar-modificacao
      senão
           iniciar-remapeamento
      fim-se
   fim-se
   retornar para verificar-modificacao
```

Figura A.10: Verificação da mudanca da configuração dos Nós de Armazenamento

.

A sub-rotina iniciar-remapeamento (Figura A.11), tem por finalidade identificar a existência da remoção de um Nó de Armazenamento e em caso afirmativo, atualizar todos os metadados e fazer a redistribuição dos objetos multimídia.

```
iniciar-remapeamento
   criar-lista
   identificar caminho-completo para metafile
   se  houver disco-removido
       mudar-status-metafiles(IE_STATE_MPR)
   senao
       mudar-status-metafiles(IE_STATE_MNR)
   fim-se
   fazer backup-restore(IE_BACKUP)   // (backup metafiles)
   se houver disco-removido
      bloquear-todos-metafiles
      remapear-metafiles
      se autorizar
         executar-exportar-importar
      fim-se
   senao
     se autorizar
        executar-exportar-importar
     fim-se
   fim-se
   retornar para verificar-disco
```

Figura A.11: Iniciar remapeamento na verificação de mudança no layout dos nós

A sub-rotina criar-lista (Figura A.12), tem por finalidade preparar os *paths* ou caminhos que são utilizados pelas sub-rotinas para ler corretamente os arquivos de trabalho.

```
cria-lista
   identificar o caminho para arquivo-lista-metafiles
   identificar o caminho de diretorios-excluidos-atualizacao
   identificar o caminho para binario-metafiles
   identificar o caminho para os dados globais de execucao
   excluir arquivos-diretorio-excluidos
   criar arquivos metafiles-selecionados
   capturar informacoes-todos-metafiles
   gravar as informacoes-metafiles em arquivo
   retornar para iniciar-remapeamento
```

Figura A.12: Preparar caminhos e arquivos para remapeamento na verificação

.

A sub-rotina mudar-status-metafiles(Figura A.13), tem por finalidade mudar o status de todos os metadados a partir do codigo-status passado para pela sub-rotina chamadora.

A sub-rotina mudar-status, tem por finalidade mudar o status de um determinado metadados a partir do codigo-status passado.

```
mudar-status-metafiles
   identificar caminho-completo arquivo-cmetafiles
   qtd_reg = calcular-número-de-objetos-metafile
   abrir cmetafile
   ind = 0
   enquanto ind menor-igual a qtd_reg
       incrementar ind
       ler registro-cmetafile
       mudar-status(caminho-completo-metafile,status)
   fim-enquanto
   fechar cmetafile
   retornar para iniciar-remapeamento

mudar-status
   inicializar-registro (metafile passado)
   posicionar registro
   ler registro
   mudar status (status passado)
   regrava o registro
   retornar para mudar-status-metafiles
```

Figura A.13: Mudar estados dos metadados

A sub-rotina remapear-exportar-importar (Figura A.14), tem por finalidade atualizar todos os metadados e em seguida realizar a redistribuição dos objetos multimídia.

A sub-rotina executar-exportar-importar, tem por finalidade realizar a redistribuição de todos os objetos multimídia.

```
remapear-exportar-importar
   remapear-metafiles
   executar-exportar-importar
   retornar para rotina-principal

executar-exportar-importar
   induzir retardo
   verificar execucao-simultanea
   ler configuracao-disco
   obter dados-conexão-servidor-RIO
   ler dados-globais
   executar-download-upload
   retornar para rotina-principal
```

Figura A.14: Ativar sub-rotinas de remapear, exportar e importar

.

A sub-rotina executar-download-upload (Figura A.15), tem por finalidade fazer a redistribuição através dos procedimentos de *download* e *upload* dos objetos multimídia nos Nós de Armazenamento do Servidor RIO.

```
executar-download-upload
   conectar servidor-RIO(root)
   montar nome-dir-root
   identificar caminho para lista arquivos
   identificar caminho-completo arquivo-cmetafiles
   qtd_reg = calcular-número-de-objetos-metafile
   abrir cmetafile
   ind = 0
   enquanto ind menor-igual a qtd_reg
         tratar-redistribuicao
   fim-enquanto
   fechar cmetafile
   escrever nova-configuracao-disco
   gravar relat-individuais-objeto em arquivo-log
   gravar relat-globais-objeto     em arquivo-log
   remover arquivos-da-tarefa
   desconectar-conexão
   liberar-memoria
   limpar-area-backup
   retornar para executar-exportar-importar
```

Figura A.15: Executar o *download* e o *upload* do objeto multimídia

A sub-rotina tratar-redistribuicao (Figura A.16), tem por finalidade ativar sub-rotina para preparar a redistribuição de cada objeto multimídia.

```
tratar-redistribuicao
   incrementar ind
   iniciar registro
   posicionar registro
   ler registro
   identificar caminho-completo para objeto
   identificar caminho-completo temporario para objeto
   ver-status-arquivo
   atualizar-por-status
   retornar para executar-download-upload
```

Figura A.16: Preparar para a redistribuição dos blocos

.

A sub-rotina ver-status-arquivo (Figura A.17), tem por finalidade analisando o status corrente de um metadado, bloquear ou desbloquear um metadado para fazer a redistribuição.

```
ver-status-arquivo
   selecao (status-arquivo)
      caso IE_STATE_MPR:
           imprimir mensagem-erro
           sai-selecao

      caso IE_STATE_WAI:
      caso IE_STATE_MNR:
      caso IE_STATE_MAP:
      caso IE_STATE_DWL:
      caso IE_STATE_DWD:
      caso IE_STATE_DON:
           mudar-acesso-metafile (caminho-completo,unlock)
           sai-selecao

      caso IE_STATE_DET:
      caso IE_STATE_DEL:
      caso IE_STATE_UPL:
           mudar-acesso-metafile (caminho-completo,lock)
           sai-selecao
   fim-selecao
   retornar tratar-redistribuicao
```

Figura A.17: Verificação do estado do metadado

.

A sub-rotina atualizar-por-status (Figura A.18), tem por finalidade analisando o status corrente de um metadado, realizar os procedimentos adequados para completar a redistribuição do objeto multimídia.

```
atualizar-por-status
   selecao (status-arquivo)
      caso IE_STATE_MPR: // Remapeamento requirido
           imprimir mensagem-erro
           sai-selecao

      caso IE_STATE_MNR: // Remapeamento não necessário
      caso IE_STATE_MAP: // Mapeamento executado
      caso IE_STATE_WAI: // inicial
           mudar-status-metafiles(IE_STATE_DWL)

      caso IE_STATE_DWL: // downloading
           agir-STATE-DWL

      caso IE_STATE_DWD: // Downloaded
           mudar-acesso-metafile (caminho-completo,lock)
            mudar-status-metafiles(IE_STATE_DET)

      caso IE_STATE_DET: // Removing
           agir-STATE-DET

      caso IE_STATE_DEL: // Removed
            mudar-status-metafiles(IE_STATE_UPL)

      caso IE_STATE_UPL: // Uploading
           agir-STATE-UPL
           sai-selecao

      caso IE_STATE_DON:
           sai-selecao
   fim-selecao
   retornar tratar-redistribuicao
```

Figura A.18: Ativar a redistribuição dos blocos a partir do estado do metadado

A sub-rotina agir-STATE-DWL (Figura A.19), tem por finalidade realizar o *download* do objeto multimídia.

```
agir-STATE-DWL
   contabilizar tempo-download
   fazer-download
   apurar duracao-download
   apurar tempos-globais
   gravar tempos-globais
   mudar-status-metafiles(IE_STATE_DWD)
   retornar atualizar-por-status
```

Figura A.19: Ativar o *download* do objeto multimídia

.

A sub-rotina agir-STATE-DET (Figura A.20), tem por finalidade realizar a remoção do objeto multimídia.

```
agir-STATE-DET
   induzir retardo
   contabilizar tempo-remocao
   remove objeto
   apurar duracao-remocao
   apurar tempos-globais
   gravar tempos-globais
   mudar-status-metafiles(IE_STATE_DEL)
   retornar atualizar-por-status
```

Figura A.20: Remover o objeto multimídia dos Nós de Armazenamento

A sub-rotina agir-STATE-UPL (Figura A.21), tem por finalidade realizar o *upload* do objeto multimídia.

```
agir-STATE-UPL
   contabilizar tempo-upload
   fazer-upload
   apurar duracao-upload
   apurar tempos-globais
   gravar tempos-globais
   mudar-acesso-metafile (caminho-completo,unlock)
   mudar-status-metafiles(IE_STATE_DON)
   retornar atualizar-por-status
```

Figura A.21: Ativar o *upload* do objeto multimídia

.

A sub-rotina limpar-exportar-importar (Figura A.22), tem por finalidade desfazer uma redistribuição que ainda não tenha sido totalmente completada.

```
limpar-exportar-importar
   verificar-execução simultânea
   obter dados-conexão-servidor-RIO
   ler dados-globais
   limpar-download-upload
   retornar para rotina-principal                                    ▫
```

Figura A.22: Desfazer uma redistribuição de blocos incompleta

A sub-rotina limpar-download-upload (Figura A.23), tem por finalidade desfazer o procedimento de *download-upload* realizado no objeto multimídia.

```
limpar-download-upload
   conectar servidor-RIO(root)
   montar nome-dir-root
   identificar caminho para lista arquivos
   identificar caminho-completo arquivo-cmetafiles
   qtd_reg = calcular-número-de-objetos-metafile
   abrir cmetafile
   ind = 0
   enquanto ind menor-igual a qtd_reg
         tratar-desmanchar
   fim-enquanto
   fechar cmetafile
   remover arquivos-da-tarefa
   desconectar-conexão
   liberar-memoria
   limpar-area-backup
   retornar para limpar-exportar-importar                            ▫
```

Figura A.23: Desfazer os procedimento de *download/upload*

A sub-rotina tratar-desmanchar (Figura A.24), tem por finalidade fazer tratamento por estado do metadado de cada objeto para desfazer a redistribuição de blocos realizada anteriormente.

```
tratar-desmanchar
   incrementar ind
   iniciar registro
   posicionar registro
   ler registro
   identificar caminho-completo para objeto
   identificar caminho-completo temporario para objeto
   identificar caminho-relativo dir-root
   identificar caminho-backup-objeto
   ver-status-desmanche
   atualizar-desmanche
   retornar para limpar-download-upload                              ▫
```

Figura A.24: Prepara procedimentos para desfazer a redistribuição de blocos

.

A sub-rotina ver-status-desmanche (Figura A.25), de acordo com o estado corrente do metadado, bloquear ou desbloquear o metadado para desfazer a redistribuição de blocos realizada anteriormente.

```
ver-status-desmanche
   selecao (status-arquivo)
      caso IE_STATE_MPR: // Nada a fazer
      caso IE_STATE_WAI:
      caso IE_STATE_MNR:
           mudar-acesso-metafile (caminho-completo,unlock)
           continua

      caso IE_STATE_MAP: // retornar backup
           restaurar-backup
           mudar-acesso-metafile (caminho-completo,unlock)
           continua

      caso IE_STATE_DON: // Voltar para downloading
           mudar-acesso-metafile (caminho-completo,lock)
           mudar-status-metafiles(IE_STATE_DWL)
           sai-selecao

      caso IE_STATE_DWL: // Continua o processo, pois tem que ser completado
      caso IE_STATE_DET: // para manter o balanceamento
      caso IE_STATE_DEL:
      caso IE_STATE_DWD:
      caso IE_STATE_UPL:
           sai-selecao
   fim-selecao
   retornar para tratar-desmanchar
```

Figura A.25: Verificar o estado do metadado para desfazer o *download/upload*

.

A sub-rotina atualizar-desmanche (Figura A.26), de acordo com o estado corrente do metadado, ativar sub-rotinas para desfazer a redistribuição dos blocos do objeto multimídia correspondente.

```
atualizar-desmanche
   selecao (status-arquivo)
      caso IE_STATE_MPR: // Remapeamento requirido
           imprimir mensagem-erro
           sai-selecao

      caso IE_STATE_MNR: // Remapeamento não necessário
      caso IE_STATE_MAP: // Mapeamento executado
      caso IE_STATE_WAI: // inicial
           mudar-status-metafiles(IE_STATE_DWL)

      caso IE_STATE_DWL: // downloading
           desmanchar-STATE-DWL

      caso IE_STATE_DWD: // Downloaded
           mudar-acesso-metafile (caminho-completo,lock)
           mudar-status-metafiles(IE_STATE_DET)

      caso IE_STATE_DET: // Removing
           desmanchar-STATE-DET

      caso IE_STATE_DEL: // Removed
            mudar-status-metafiles(IE_STATE_UPL)

      caso IE_STATE_UPL: // Uploading
           desmanchar-STATE-UPL
           sai-selecao

      caso IE_STATE_DON:
           sai-selecao
   fim-selecao
   retornar para tratar-desmanchar
```

Figura A.26: A partir do estado do metadado ativar sub-rotinas para desfazer redistribuição

.

A sub-rotina desmanchar-STATE-DWL (Figura A.27), fazer o *download* do objeto multimídia para a área de trabalho com a finalidade de desfazer a redistribuição de blocos realizada anteriormente.

```
desmanchar-STATE-DWL
   contabilizar tempo-download
   fazer-download
   apurar duracao-download
   apurar tempos-globais
   gravar tempos-globais
   mudar-status-metafiles(IE_STATE_DWD)
   retornar para atualizar-desmanche
```

Figura A.27: Ativar *download* do objeto multimídia para desfazer redistribuição

A sub-rotina desmanchar-STATE-DET (Figura A.28), realizar a remoção do objeto multimídia dos Nós de Armazenamento com a finalidade de desfazer a redistribuição de blocos realizada anteriormente.

```
desmanchar-STATE-DET
   induzir retardo
   remove o objeto
   mudar-status-metafiles(IE_STATE_DEL)
   retornar para atualizar-desmanche
```

Figura A.28: Remover objeto multimídia para desfazer a redistribuição

A sub-rotina desmanchar-STATE-UPL (Figura A.29), fazer o *upload* do objeto multimídia da área de trabalho com a finalidade de desfazer a redistribuição de blocos realizada anteriormente.

```
desmanchar-STATE-UPL
   contabilizar tempo-upload
   fazer-upload
   apurar duracao-upload
   apurar tempos-globais
   gravar tempos-globais
   mudar-acesso-metafile (caminho-completo,unlock)
   mudar-status-metafiles(IE_STATE_DON)
   retornar para atualizar-desmanche
```

Figura A.29: Ativar *upload* do objeto multimídia para desfazer redistribuição

Referências

[1] HONG, G. Y.; FONG, A. C. M.; FONG, B. Adaptive qos control of multimedia transmission over band-limited networks. p. 644 – 649, August 2002. Consumer Electronics, IEEE Transactions on.

[2] WU, K.-L.; YU, P. S.; WOLF, J. L. Segment-based proxy caching of multimedia streams. In: *WWW '01: Proceedings of the 10th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2001. p. 36–44.

[3] NGUYEN, T.; MEHRA, P.; ZAKHOR, A. Path diversity and bandwidth allocation for multimedia streaming. *icme*, IEEE Computer Society, Los Alamitos, CA, USA, v. 2, p. 1–4, 2003.

[4] WU, K.-L.; YU, P. S.; WOLF, J. L. Segmentation of multimedia streams for proxy caching. *IEEE Transactions on Multimedia*, v. 6, n. 5, p. 770–780, 2004.

[5] CHEN, S.-C. et al. An adaptive multimedia transmission protocol for distributed multimedia applications. In: *Distributed Computing Systems Workshops 23rd International Conference on*. [S.l.: s.n.], 2003. p. 537– 542.

[6] SANTOS, J. R. G. *RIO: A Universal Multimedia Storage System Based on Random Data Allocation and Block Replication*. Tese (Doutorado) — University of California - UCLA, Los Angeles, 1998.

[7] BERSON, S.; GOLUBCHIK, L.; MUNTZ, R. R. Fault tolerant design of multimedia servers. In: *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM Press, 1995. p. 364–375. ISBN 0-89791-731-6.

[8] DAN, A.; KIENZLE, M. G.; SITARAM, D. Dynamic segment replication policy for load-balancing in video-on-demand servers. *ACM Multimedia Systems*, v. 3, n. 3, p. 93–103, 1995.

[9] WANG, B. et al. Optimal proxy cache allocation for efficient streaming media distribution. *IEEE Transaction on Multimedia*, v. 6, n. 2, p. 366–274, April 2004.

[10] MORAN, J. M. *O que é educação a distância*. Acessado em 15 de Julho de 2006 as 17:05 hs. Disponível em: <http://www.eca.usp.br/prof/moran/dist.htm>.

[11] MARQUES, C. *Experiências inovadoras em educação a distância no Brasil - com informações do Senac, MEC e do livro ead.br publicado em 2004 pela Editora Anhembi Morumbi*. Acessado em 15 de Julho de 2006 as 16:55 hs. Disponível em: <http://www1.folha.uol.com.br/folha/educacao/ult305u16139.shtml>.

[12] CEDERJ. *CENTRO DE EDUCAÇÃO SUPERIOR A DISTÂNCIA DO ESTADO DO RIO DE JANEIRO*. Acessado em 18 de Agosto de 2006 as 09:50 hs. Disponível em: <http://www.cederj.edu.br/cecierj/>.

[13] COSTA, C. et al. *Analyzing client interactivity in streaming media*. May 2004. 534-543 p. In Proc. of the International World Wide Web Conference, May 2004.

[14] TUDOR, P. N. Mpeg-2 video compression. *IEEE Eletronics & Communication Engineering Journal*, v. 7, p. 257–264, 1995.

[15] WALLACE, G. K. The jpeg still picture compression standard. *Commun. ACM*, v. 34, n. 4, p. 30–44, 1991.

[16] GSM. *Global System for Mobile Communication*. Acessado em 18 de Agosto de 2006 as 10:22 hs. Disponível em: <http://www.iec.org/online/tutorials/gsm>.

[17] MP3. *MPEG Audio Layer-3*. Acessado em 18 de Agosto de 2006 as 11:04 hs. Disponível em: <http://www.iis.fraunhofer.de/amm/techinf/layer3/index.html>.

[18] KUROSE, J. F.; ROSS, K. W. *Rede de Computadores e a Internet - 3o. Edição*. [S.l.]: Pearson Education, 1997.

[19] PROXY. *Tutorial proxy*. Acessado em 18 de Agosto de 2006 as 11:50 hs. Disponível em: <http://penta.ufrgs.br/redes296/proxy/proxy.html>.

[20] HENNESSY, J. L.; PATTERSON, D. A. *Computer Architecture A Quantitative Approach - Third Edition*. [S.l.]: Morgan Kaufmann Publishers, 2003.

[21] PATTERSON, D. A.; GIBSON, G. A.; KATZ, R. H. A case for redundant arrays of inexpensive disks (raid). In: *Proceedings of the International Conference on Management of Data (SIGMOD)*. Chicago: ACM - SIGMOD Conference, 1988. p. 109–116.

[22] FLOYD, S. et al. Equation-based congestion control for unicast applications. In: *SIGCOMM 2000*. Stockholm, Sweden: [s.n.], 2000. p. 43–56.

[23] ALMEROTH, K. The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment. *IEEE Network*, v. 14, p. 10–20, jan 2000.

[24] SHENOY, P. J.; VIN, H. M. Efficient striping techniques for multimedia file servers. In: *Proceedings of the Seventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97), St. Loius, MO*. St. Louis, Missouri: Proceedings of the IEEE, 1997. p. 25–36.

[25] BERSON, S.; MUNTZ, R. R.; WONG, W.-M. R. Randomized data allocation for real-time disk i/o. In: *IEEE Computer Society Conference, COMPCON'96*. [S.l.]: IEEE Computer Society, 1996. p. 286–290.

[26] KORST, J. Random duplicated assignment: an alternative to striping in video servers. In: *MULTIMEDIA '97: Proceedings of the fifth ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 1997. p. 219–226. ISBN 0-89791-991-2.

[27] BIRK, Y. Random raids with selective exploitation of redundancy for high performance video server. In: *Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*. St. Louis, Missouri, USA: Proceedings of the IEEE, 1997. p. 13–23.

[28] FABBROCINO, J. R. S. F.; MUNTZ, R. *An Implicitly Scalable, Real-Time Multimedia Storage Server*. Montreal, Canada, July 1998. 17 p.

[29] SANTOS, J. R.; MUNTZ, R. R.; RIBEIRO-NETO, B. A. Comparing random data allocation and data striping in multimedia servers. In: *SIGMETRICS*. Santa Clara: ACM SIGMETRICS, 2000. p. 44–55.

[30] SILVA, E. de Souza e et al. Performance issues of multimedia applications. In: *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*. London, UK: Springer-Verlag, 2002. p. 374–404. ISBN 3-540-44252-9.

[31] SANTOS, J. R.; MUNTZ, R. Performance analysis of the rio multimedia storage system with heterogeneous disk configurations. In: *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 1998. p. 303–308. ISBN 0-201-30990-4.

[32] BUENO, A. D. *Introdução ao Processamento Paralelo e ao Uso de Cluster de Workstations em Sistemas GNU/LINUX Parte II - Processos e Threads*. 2003. Universidade Federal de Santa Catarina - UFSC.

[33] AZEVEDO, J. A. et al. *FreeMeeting: um ambiente para trabalho cooperativo e ensino a distância*. April 2006. 7th International Free Software Forum May 2004. - Porto Alegre, RS, Brazil.

[34] MPEGTV. *MPEGTV. mtvp - MPEG Movie Player and VCP Player for Linux*. Acessado em 17 de Julho de 2006 as 15:25 hs. Disponível em: <http://www.mpegtv.com>.

[35] CARDOZO, A. de Q. *Mecanismo para Garantir Qualidade de Serviço de Aplicações de Vídeo sob Demanda*. Dissertação (Mestrado) — Universidade Federal Fluminense - UFF, Niteroi, 2002.

[36] MUNTZ, R.; SANTOS, J. R.; BERSON, S. Rio: a real-time multimedia object server. *SIGMETRICS Perform. Eval. Rev.*, ACM Press, New York, NY, USA, v. 25, n. 2, p. 29–35, 1997. ISSN 0163-5999.