Relatório de MAC 5701
Tópicos em Ciência da Computação
Estudo Sobre Estruturas de Cache

Luciano Vieira de Araújo
luciano@ime.usp.br
Orientador:
Prof. Dr. João Eduardo Ferreira

Sumário

Capítulo 1

Introdução

Já nos dias de hoje a computação móvel (CM) é uma realidade inquestionável. O seu crescimento é embalado pela a popularização dos computadores portáteis, pela globalização, pela necessidade de informações rápidas e pelo desejo de mobilidade. Com toda esta demanda surgem os desafios computacionais a serem tratados para garantir este crescimento.

O uso de estruturas de cache, que a partir de agora chamaremos de cache, tem como objetivo permitir um acesso mais rápido aos dados. Além de proporcionar uma redução na sobrecarga dos canais de comunicação sem fios. Fato desejável tendo em vista, a grande limitação de largura de banda existente nestes canais. Estes objetivos são atingidos quando o uso do cache se torna efetivo. O que implica no uso dos recursos computacionais do cliente, através do processamento dos dados no cache. Isso permite que o cliente móvel, conhecido como Mobile Unit(MU), trabalhe desconectado. Ou seja, sem a necessiade de conexão com o servidor.

Para trabalhar desconectado o MU inicialmente solicita os dados que necessita ao servidor. Com os dados em memória, ele se desconecta e as operações de consultas e atualização passam a ser executas localmente. Caso os dados em memória não sejam suficientes para as tarefas solicitadas, o usuário é informado e nova conexão é sugerida. Ao final da execução, caso o cache permita atualizações locais, a conexão com o servidor é restabelecida para que os dados sejam reintegrados no servidor. O trabalho desconectado é muito importante para o MU já que conexões muito longas são mais susceptíveis a queda, devido as limitações de largura de banda, de carga de bateria dos dispositivos e de área de alcance de comunicação. A seguir enumeramos alguns dos benefícios do uso do cache na computação móvel.

1. Maior Mobilidade: já que o MU pode trabalhar, por um período, sem trocar dados com o servidor, ele pode se mover para áreas fora de cobertura da comunicação sem fio.

2. Menor Carga no Canal Sem Fio e no Servidor: ao trabalhar desconectado, a comunicação entre servidor e MU fica concentrada em dois momentos. O

primeiro, quando ele solicita os dados ao servidor. E o segundo momento, quando ele devolve os dados ao servidor, caso isso seja necessário. Mesmo trabalhando conectado, a aplicação só solicitará dados que não estejam no cache.

3. Maior Disponibilidade de Recursos: como o cliente não necessita ficar conectado o tempo todo com o servidor, os recursos computacionais ganham maior potencial de compartilhamento.

4. Economia Financeira: todos os fatores anteriores claramente implicam em economia financeira. Seja em equipamentos ou no custo da comunicação sem fio.

Permitir a agregação destes benefícios a diversas aplicações é a motivação para as pesquisas sobre cache. Mesmo que algumas destas limitações desapareçam com o avanço tecnológico, o cache continuará como uma alternativa na melhoria do desempenho da aplicações, já que sua diretriz básica é evitar a redundância de solicitação de dados já utilizados.

Capítulo 2

Cache

De maneira geral, o uso de cache, tem como objetivo um acesso mais eficiente aos dados. Para que isso aconteça, os dados devem ser armazenados no cache de maneira que possamos manipulá-los com facilidade. Podemos descrever o funcionamento do cache da seguinte forma:

- Inicialmente - O cache não possui dados.
- Formando o conteúdo do Cache - Neste momento os dados começam a ser armazenados no cache. Isso é feito baseados em configurações prévias que definem quais dados devem ser armazenados no cache ou fazendo uso do resultados das consultas feitas pelo cliente.
- Utilização do Cache - Com dados armazenados em sua estrutura o cache pode ser efetivamente usado. Ou seja, a cada consulta ou atualização o cache procura trabalhar prioritariamente com os seus dados.
- Gerenciamento do Cache - Durante a execução da aplicação pode surgir a necessidade de armazenagem de novos dados. Devido as limitações de memória do cache é necessário que seus dados sejam substituídos. Isso é feito, obedecendo a política de substituição de dados.
- Reintegração de Dados - Caso o cache permita atualizações locais, uma nova conexão será solicitada para o envio dos dados atualizados ao servidor. Cabe lembra que a tarefa de tratar os conflitos de reintegração dos dados cabe ao servidor e foge ao escopo do cache.

Estas etapas não seguem necessariamente a ordem que foram apresentadas e alguma delas podem ser realizadas várias vezes.

2.1 Tipos de Cache

Existem vários tipos de cache conhecidos. Apesar de suas diferenças básicas de implementação e gerenciamento, todos tem o objetivo de aumentar a eficiência no acesso ao dado. A seguir descreveremos as abordagens possíveis para a utilização dos dados no cache. Esta classificação segue a abordagem de [2].

- Dados somente para consulta - O cache não permite atualizações locais. Logo, somente operações de consultas são realizadas no cache, sem que haja a necessidade de reintegração de dados.
- Dados não Concorrente - Neste caso, atualizações locais são permitidas. Mas os dados replicados ficam bloqueados no servidor. Assim, a reintegração dos dados é feita naturalmente, sem a necessidade de tratamento de conflito.
- Dados Concorrente - Este caso ocorre quando os dados, mesmo depois de replicados, podem ser alterados no servidor e no cache. Desta maneira a reintegração dos dados deve ocorrer com o tratamento de conflitos.

2.2 Estrutura de Dados do Cache

A estrutura de dados do cache deve prover facilidade de gerenciamento dos dados armazenados e também permitir sua fácil manutenção através da inclusão e exclusão de dados. De maneira geral o cache é formado de um vetor de posições, onde cada posição possui dados necessários ao seu gerenciamento, como: posição, bit de alteração, marca de tempo e valor. Completando a estrutura temos um diretório de cache que possui informações sobre os dados armazenados no cache, como: posição e item de dado[6].

2.3 Classificação dos Tipos de Cache

O funcionamento do cache é baseado em características como a natureza de seu conteúdo e a unidade básica de dados. A partir destas características é definida sua estratégia de gerenciamento de dados, principal responsável pelo bom desempenho do cache. A escolha da combinação destes fatores deve atender as características de cada aplicação e do perfil do usuário. Idealmente, a aplicação deve ser capaz de reconhecer o contexto de seu funcionamento e se adaptar a ele. Por exemplo, no caso da computação móvel é desejável que a aplicação saiba tratar de forma difereciada o cliente móvel de um cliente fixo ou ainda, a sobrecarga do servidor ou sua ociosidade.

Segue uma descrição das características que compõe o cache.

2.3.1 Natureza do Cache

Natureza do cache diz respeito a quando e como o seu conteúdo é definido. O cache pode ter uma natureza estática ou dinâmica.

- Natureza Estática - Indica que o conjunto de dados necessário à aplicação é definido antes da criação do cache e de maneira fixa. Assim, o conteúdo do cache não sofre alterações durante a execução da aplicação. Este tipo de cache é encontrado em aplicações onde o comportamento e necessidades do cliente são bem conhecidos.

- Natureza Dinâmica - Define a capacidade do cache definir seu conteúdo de forma dinâmica e flexível em tempo de execução. Isso é feito através do armazenamento dos resultados das consultas.

2.3.2 Unidade do Cache

Unidade de dados do cache ou grão indica a menor quantidade de dados a ser armazenada ou substituída. A granulosidade do cache esta diretamente relacionada com a quantidade de informações de controle necessárias para referenciar os dados no cache e a melhor reutilização dos dados.

As unidades mais utilizadas em cache são:

- Arquivos
- Tabelas
- Objetos
- Páginas de dados
- Consultas (Query)
- Tuplas

Usaremos dois exemplos para ilustrar a influência da granulosidade do cache no seu funcionamento.

- Granulosidade Pequena - Unidade: Tuplas - Para permitir o uso de seu conteúdo, o cache deve possuir um apontador para cada unidade de dados armazenada. Como neste caso a unidade é a menor possível, um "overhead"de informações de controle é observado. Porém existe um ganho na flexibilidade de gerenciamento dos dados no cache que pode ser feito tupla a tupla. Entretanto, cabe considerar o aumento de complexidade para o gerenciamento do cache devido a necessidade de verificação de um número maior de unidades.

- Granulosidade Grande - Unidade: Arquivos - Neste caso, ocorre o oposto. As informações de controle ficam reduzidas, já que só existem referências para arquivos. O que implica que só poderemos incluir ou excluir do cache um arquivo inteiro, sem a possibilidade de utilizar parte de seu conteúdo. Limitando assim, a reutilização de dados.

Definir qual a unidade básica do cache depende do tipo de flexibilidade que se deseja do cache. Cada aplicação possui sua própria relação de custo/benefício entre o "overhead"de informações de controle e a flexibilização para a reutilização dos dados.

Capítulo 3

Estratégias de Gerenciamento dos Dados

As estratégias de gerenciamento de dados procuram melhorar o desempenho do cache através de regras de solicitação e substituição dos dados. Esta regras são elaboradas considerando as características de cada cache, como: Natureza, unidade de dados, etc.

3.1 Solicitação de dados ao servidor

A solicitação de dados é feita quando o cache não consegue responder completamente a uma consulta. Isso ocorre quando o cache esta vazio ou seus dados não são suficientes. Ela pode ser feita de forma total ou parcial como descrito a seguir.

3.1.1 Solicitação Total

Todos os dados necessários para responder a consulta são solicitados ao servidor. Mesmo que parte destes dados já estejam em cache. Existem duas conseqüências básicas desta abordagem. A primeira é a simplificação da administração da estrutura. Pois, o cache só será utilizado se o resultado completo da consulta estiver em seu conteúdo. A segunda é a ineficiência do uso do cache. Já que não é possível reutilizar somente parte de seus dados.

3.1.2 Solicitação Parcial

Somente os dados que não estão no cache são solicitados. Para isso, o cache deve identificar quais dados estão armazenados e quais devem ser solicitados.

A solicitação parcial de dados é limitada pela granulosidade do cache, pois a referência é feita a unidade de dados e não ao próprio dado. Por exemplo, considerando um cache de páginas de dados, não podemos fazer solicitação de somente uma tupla de dados. A obtenção de uma única tupla passa, necessariamente, pela solicitação de uma página de dados completa. Isso torna a solicitação parcial mais apropriada para caches de tuplas ou consultas.

A solicitação parcial é feita utilizando o processo de divisão de consultas (query trimming) que é descrito a seguir.

- *Query Trimming - Divisão de Consultas* é o processo de dividir a consulta original em duas consultas. Uma recupera dados que estão no cache e outra solicita os dados que estão faltando.
- *Query Probe - Consulta Exploratória* é a subdivisão da consulta original responsável pela recuperação dos dados que estão no cache.
- *Query Remainder - Consulta Restante* é a subdivisão da consulta original que deve solicitar ao servidor somente os dados que faltam.

Para ilustrar melhor o funcionamento do procedimento de query trimming, apresentamos o exemplo a seguir. Nele é utilizado o arquivo Funcionario (id, nome, salario, idade). Inicialmente temos os predicados que representam os dados solicitados pela consulta original e conteúdo do cache. Em seguida, temos o resultado do procedimento de query trimming. Ou seja, os predicados representando a probe query e a remainder query.

- Query Original
  (salario $>$ 50000 and idade $\leq$ 30)
- Conteúdo do Cache
  (salario $<$ 100000 ou (idade $>$ 25 e idade $\leq$ 28))

Resultado do Query Trimming

- Probe Query
  (((salario $>$ 50000 and salario $<$ 10000) and idade $\leq$ 30) ou (salario $>$ 50000 and idade $>$ 25 e idade $\leq$ 28))
- Remainder Query
  ((salario $\geq$ 100000 and idade $<$ 25) ou (salario $\geq$ 100000 and idade $\geq$ 25 and idade $\geq$ 30))

Como podemos perceber no exemplo acima, o processo de divisão da consulta original em query probe e query remainder não é trivial. É necessário que a aplicação seja capaz de avaliar e otimizar opções de SQL. Se for levado em consideração todos os possíveis operandos de um predicado e o uso de join, o problema se torna NP-Completo[1]. Ou seja, sem solução conhecida. Para não descartar a possibilidade de utilização parcial do cache é necessário uma simplificação do modelo. Como a limitação do número de tabela usadas ou até mesmo, a proibição de joins. Também deve ser excluído o uso dos operandos negação.

Apesar destas limitações, muitas aplicações podem ser beneficiadas pela solicitação parcial de dados.

3.2 Substituição dos Dados no Cache

O desafio de performance das estratégias de substituição dos dados no cache é propor um modelo, que de forma eficiente e rápida, decida quais dados devem permanecer ou não em memória. Para tomar estas decisões o cache se utiliza de valores agregados aos seus itens de dados. Como por exemplo: custo de solicitação, histórico de acesso e dados temporais entre outros. Assim, quando uma nova consulta chega, o gerenciador define quais dados devem ser substituídos por ela.

3.3 Modelos de Substituição de Dados

Os modelos de substituição de dados são baseados em indicadores de potencial de reutilização de um conjunto de dados. Através das informações agregadas aos dados o cache infere o potencial de reuso definindo os dados que devem permanecer ou ser substituídos, regiões "quentes" e " frias" respectivamente.

3.3.1 Afinidade Temporal

Este modelo esta baseado na agregação de dados temporais as unidades de dados do cache. Nesta caso, saber quando um dado foi acessado é um forte indicador para possíveis novos acesso.

- *LRU - Less Recently Used* - Indica a região "fria"do cache. Ou seja, dados que não foram acessados recentemente possuem pouca possibilidade de novo uso. Tornando-os candidatos a substituição.
- *MRU - Most Recently Used* - Define o conjuntos de dados que foram acessados mais recentemente. Também conhecida como região "quente"do cache. Segundo a abordagem temporal, esse conjunto de dados possuem grande probabilidade de reuso.
- *FIFO - First In First Out* - Este é o funcionamento clássico de uma fila. O primeiro a entra é o primeiro a sair. Para o cache, o FIFO indica que o dado mais antigo no cache deve ser o primeiro a ser substituído.

3.3.2 Afinidade de Freqüência de Acesso

Abordagem sugerida em [5], que contabiliza informações de freqüência de acesso ao dado. Nesta abordagem, um dado pouco acessado tem baixa probabilidade de ser solicitado novamente e deve ser substituído.

- *LFU - Less Frequently Used* - Define o conjunto de dados com o menor freqüência de acessos. Configurando a região "fria"do cache.
- *MFU - Most Frequently Used* - Indica o região "quente"do cache. Ou seja, os dados que foram mais freqüentemente usados.

3.3.3 Afinidade Regional ou Semântica

Esta abordagem procura estabelecer um relacionamento semântico entre os dados. Um tipo de afinidade que sugere coesão entre eles e possível solicitação seqüencial. Um exemplo desta característica semântica seria uma aplicação forneça informações sobre o trânsito. Devido a suas características geográficas e regionais, podemos dizer que uma rua esta semanticamente relacionada com suas travessas e ruas próximas. O que define uma região semântica.

3.3.4 Afinidade Mista

A abordagem de substituição de dados mista procura aumentar a reutilização dos dados através da união de abordagens. Como é o caso da abordagem proposta em [1]. Ela sugere a cooperação das abordagens temporal e regional e um modelo de substituição chamadado LRU de 2 Níveis, descrito adiante. Assim, os dados recebem informações temporais e de relacionamento semântico para que seja inferido o seu potencial de reuso.

A vantagem da abordagem mista é a possibilidade de uma região considerada fria, por somente uma abordagens, ser considerada quente pela analise de mais de uma afinidade.

Capítulo 4

Cache Semântico

Com o crescimento de aplicações que necessitam acessar os dados de forma associada, o cache semântico (CS) tem surgido como uma alternativa viável de melhoria de desempenho. Seu principal atrativo é a capacidade de relacionar os dados semanticamente e permitir acessos em conjunto. Nos caches tradicionais o acesso é feito diretamente ao dado, que por sua vez não possui associações com outros dados. Devido a estas características, muitas pesquisas tem procurado melhorar as estruturas de armazenamento e administração do CS como em [1][3][4] e [5]. Com esta motivação, nos aprofundaremos um pouco mais em sua descrição.

4.1 O Modelo do Cache Semântico

O CS utiliza informações semânticas entre as consultas para organizar e gerenciar o cache. Ele é composto por conjuntos distintos de segmentos semânticos, os quais originalmente eram decompostos. A junção destes segmentos formam as chamadas regiões semânticas do cache, que podem ter seu tamanho alterado dinamicamente.

Segue uma verificação das características do CS em relação aos fatores de desempenho do cache.

4.1.1 Natureza

Uma das características principais do CS é sua adaptação dinâmica ao perfil da aplicação. Logo, o CS possui natureza dinâmica de definição de conteúdo. Isso é feito através do armazenamento dos resultados das consultas solicitadas.

4.1.2 Granulosidade

O CS usa como unidade básica de dados, regiões semânticas. Estas regiões são definidas dinamicamente, de acordo com as consultas solicitadas pelo cliente, e possuem tamanho variável. Considerando o tamanho da tuplas como sendo fixo.

Podemos enumerar duas diferenças básicas entre esta abordagem e a de granulosidade de tuplas.

1. Diminuição no "overhead"de dados de controle. Tendo em vista que estes dados são agregadas ao conjunto de tuplas e não a cada tupla.

2. Completa flexibilidade na criação e definição do tamanho da região semântica. Permitindo que a região semântica se adapte as necessidades da consulta atual.

4.1.3 Solicitação de Dados

Como CS é composto por regiões semânticas, formadas por um conjunto de tuplas relacionadas. Ele pode explorar esta flexibilidade e solicitar ao servidor somente os dados que ele não possui. Caso, os todos os dados estejam no cache nenhuma solicitação é feita. A solicitação parcial de dados é feita através do processo de query trimming, visto anteriormente.

4.1.4 Substituição de Dados

Em relação a política de substituição de dados no cache, o CS faz uso da abordagem de afinidade regional ou semântica. Levando em consideração que dados semanticamente relacionados são seqüencialmente solicitados. Assim, os dados com menor índice de relacionamento semântico formam a região "fria"do cache, sendo candidatos a substituição. Uma outra característica relevante da substituição de dados em CS é a flexibilidade na variação do tamanho da região semântica. Ela cresce de acordo com a afinidade dos dados armazenados. Por exemplo, se o conteúdo de uma consulta esta relacionada a outra, elas passam a fazer parte da mesma região.

Mesmo fazendo uso de afinidade regional, a performance deste processo depende de quão relacionadas semanticamente são as consultas. Além da habilidade da política de substituição de utilizar a relação semântica dos dados.

4.2 Composição da Região Semântica

A região semântica é composta basicamente de quatro itens, como descrito a seguir.

- *Fórmulas de Restrição* - Elas definem o conteúdo de cada região. Esta fórmulas são utilizadas no processo de solicitação dos dados não encontrados no cache.

- *Dados* - São as tuplas que satisfazem as fórmulas de restrição da região.

- *Apontadores* - Ponteiros para a lista ligada das atuais tuplas que compõem a região.
- *Informações Adicionais* - Dados agregados a região e utilizados na política de substituição de dados. Elas criam um rank entre as regiões, possibilitando a decisão de substituição das regiões.

4.3 Bases do cache semântico

O CS baseia-se na composição dinâmica do grupo de tuplas. Ou seja, a medida que as consultas vão sendo armazenadas no cache, suas afinidades são identificadas e criados os relacionamentos entre elas. Passamos as três idéias básicas do cache semântico[3]:

1. A máquina cliente possui uma descrição semântica do cache, ao invés de uma listagem física de páginas ou tuplas. Esta descrição semântica será usada durante o processamento de solicitação parcial dos dados.
2. A informação usada na política de substituição do cache é mantida de forma adaptável as regiões semânticas, as quais estão associadas a conjuntos de tuplas. Estes conjuntos de tuplas são definidos e ajustados dinamicamente, baseados nas consultas que estão na máquina cliente. O uso de regiões semânticas, ao invés de tuplas, diminui o "overhead"das informações de controle dos dados.
3. A descrição semântica do cache encoraja o uso de funções sofisticadas para definição dos dados a serem substituídos. Funções de valores que incorporam noções de localidade podem ser bastante úteis em aplicações tradicionalmente baseadas em consultas, como é o caso de banco de dados móveis.

4.4 Funcionamento do Cache Semântico

O CS gerencia o seu conteúdo como uma coleção de regiões semânticas, que por sua vez agrupam tuplas semanticamente relacionadas.

Cada tupla no cache esta relacionada a somente uma região semântica. E cada região tem sua fórmula de restrição e seu valor de substituição, usado no gerenciamento do cache. Todas as tuplas de uma região possuem o mesmo valor de substituição.

O gerenciamento do cache esta todo voltada para o tratamento das consultas que vão surgindo. Isso é feito da seguinte maneira, quando uma consulta intercepta um região semântica do cache, ela é divida em duas partes. Uma corresponde a interseção da consulta e a região, que passa a integrar a região interceptada. A outra parte é o restante da query que fará parte de outras regiões existentes ou de uma nova.

Assim, uma consulta que sobrepõe n regiões pode gerar $2n + 1$ regiões. Desta forma, uma decisão que o CS deve tomar é quando criar uma nova região ou agrupar as já existentes. Pois um número muito grande de regiões pode gerar um grande "overhead"de informações de controle. Enquanto, regiões muito grandes pode tornar o cache pouco eficiente na reutilização de dados.

4.5 O Uso de Clusters

Uma proposta para o uso efetivo do cache semântico é feita em [1]. Ela sugere o agrupamento de consultas semanticamente relacionadas. Além de uma política de substituição de dados chamada de LRU de 2 níveis. Nesta abordagem, o cache é dividido em dois níveis.

- Clusters - Conjunto de grupo de tuplas. Ele passa a fazer o papel de região semântica agregando os grupos de tuplas relacionados. Ao invés de colocá-las juntas, elas são colocadas em grupos e mantidas no cluster.
- Grupo de tuplas - Conjunto de tuplas formado por partes das consultas semanticamente relacionadas. Ele é criado quando uma consulta é respondida por segmentos de diferentes grupos de tuplas. Estes segmentos passam a formar um novo grupo e será colocado no cluster com quem ele possuir maior afinidade semântica.

4.5.1 Algoritmo do LRU de 2 Níveis

A criação do LRU de 2 Níveis tem como objetivo explorar melhor as características do cache de clusters. Isso é feito atribuindo valores semânticos e temporais aos cluster e aos grupos de tuplas. A decisão de substituição de dados é feita em duas etapas. A primeira identifica o cluster com o rótulo de tempo mais velho. A segunda busca, dentro deste cluster, o grupo de tuplas com o rótulo de tempo mais velho, para que seja substituído. Caso o espaço liberado não seja suficiente, novos grupos de tuplas ou até mesmo clusters serão eliminados. Sempre obedecendo o padrão LRU.

Segue o Pseudo-código do LRU de dois níveis.

```
Algoritmo LRU 2 Níveis
While (existe clusters no cache) {
  Clu := O cluster do cache com o timestamp mais antigo;
  While(Existe segmento semântico em Clu) {
    S := Segmento semântico em Clu com o timestamp mais antigo;
    Elimine S do cache;
    Apague S do clu;
    Adiciona espaço livre;
    if (espaço livre é suficiente) devolve (sucesso);
```

```
    }
    Apague Clu do cache;
  };
  devolve(falha);
```

A divisão do cache em dois níveis torna mais fino o tratamento das regiões semânticas. Permitindo que sua granulosidade varie, de acordo com a necessidade de espaço, aumentando a reutilização de dados.

Capítulo 5

Conclusões

O objetivo deste trabalho foi conhecer as características das estruturas de cache e seu potencial de uso.

Desse levantamento bibliográfico iniciaremos um estudo para a implementação de "cache" semântico para computação móvel. Em especial abordaremos as especificações do projeto SIDAM para o tratamento das soluções mistas para o cache. Buscando a utilização de predicados mais sofisticados, sem deixar de aproveitar as vantagens da solicitação parcial de dados.

A grande vantagem da abordagem sugerida é a capacidade de captar mais regiões de dados sujeitas ao tratamento da solução de afinidade mista proporcionada pela nova concepção de cache.

Referências Bibliográficas

[1] Qun Ren, Margaret H. Dunham, *Using Clustering for effective Management of a Semantic Cache in Mobile Computing*, procedente de ACM 1999.

[2] P.M. Despande, K.Ramasamy, A. skukla, J.f. Naugthon, *Caching Multidimensional Queries Using Ckunks.* Procedente de SIGMOD, 1998 - pág 259-270

[3] Shaul Dar, Michael J. Franklin, Bjor T. Jonsson, Divesh Srivastava, Michael Tan, *Semantic Data Caching and Replacement.* Procedente de VLDB, 1996 - pág 330-341

[4] Qun Ren, Margaret H. Dunham, *Semantic Caching and Query Processing*, Technical Report 98-CSE-4, Southern Methodist University, May 1998.

[5] Parke Godfrey, Jarek Gryz, *Answering Queries by Semantic Caches*, University of Maryland, Aug 1998.

[6] João Eduardo Ferreira, Macelo Finger, *Controle de Concorrência e Distribuição de Dados: Teoria clássica, suas limitações e extensões modernas*, Escola de Computação - Julho 2000.