Um Framework para Construção de Máquinas de Execução de Consultas

Fausto Véras Maranhão Ayres[1]
fausto@inf.puc-rio.br

Fabio André M. Porto
porto@inf.puc-rio.br

Alvaro Cesar Pereira Barbosa
Dep. de Informática, Universidade Federal do Espírito Santo
alvaro@inf.ufes.br

Rubens Nascimento Melo
rubens@inf.puc-rio.br

PUC-Rio.Inf.MCC15/02 Julho, 2002

Abstract

The integrated access to web published information allows for offering new services not initially foreseen by the individual web publishers. This type of application requires the distributed execution of queries over web. Although very promising, this strategy suffers from serious performance problems due to the unpredictability of access time to the data sources. This work identifies and analyses query execution characteristics that should be supported by a flexible execution engine. We propose and implement a software framework that provides for the instantiation of different query execution engines. In particular, the framework instantiates an adaptive query execution engine that supports web integration applications. This framework is also integrated to the CoDIMS environment (Configurable Data Integration Middleware System).

Keywords: Data Integration, Framework, Query Execution Engine, Adaptive Queries, Configured Systems.

Resumo

O acesso integrado à informações publicadas na web possibilita a oferta de serviços não previstos quando da publicação dos sites. Aplicações como essa envolvem a execução distribuída de consultas com acesso integrado aos sites web. Este processo, no entanto, pode apresentar problemas sérios de desempenho frente à imprevisibilidade do tempo de acesso às fontes de dados. Este trabalho analisa profundamente as características de uma máquina de execução de consultas (MEC) capaz de executá-las de maneira inteiramente adaptativa. Um framework de máquina de execução de consultas é proposto sendo validado pela implementação de uma instância adaptativa da MEC. O framework está integrado ao ambiente CoDIMS de geração de sistemas configurados de integração de dados.

Palavras–chave: Integração de Dados, Framework de Software, Execução de Consultas, Consultas Adaptativas, Sistemas Configurados.

---

[1] Professor parcialmente licenciado do CEFET-RJ

1 INTRODUÇÃO

O acesso integrado à informações publicadas na *web* permite que sejam oferecidos serviços combinando sites, a princípio projetados autonomamente. Suponha, pôr exemplo, três *sites* web disponibilizando as seguintes informações: programação das apresentações musicais da semana, oferta de venda de CDs e *ranking* dos CDs mais vendidos. Um usuário de uma aplicação de integração destes sítios poderia submeter uma consulta como esta: "Informar os CDs, com seus respectivos preços, de um determinado gênero musical que estejam na lista dos mais vendidos". Aplicações como essa envolvem a execução distribuída da consulta com acesso integrado aos três sites Web.

O suporte para execução de consultas, como a sugerida, é provido pôr sistemas de mediadores (ex. TSIMMIS, Garlic, Disco, Tukwila, Le Select) que oferecem uma visão integrada das fontes de dados transformando geralmente os dados obtidos segundo um modelo de dados semi-estruturado para elementos no modelo de dados de integração.

Nestes sistemas, a consulta formulada a partir do modelo de integração deve ser otimizada e executada pelo sistema mediador. O plano de execução de consultas (PEC) assim produzido incluirá operações de *BindJoin*[FMLS99], nas quais formulários web são modelados como relações virtuais com restrições de acesso, permitindo seu tratamento pelo processo tradicional de otimização de consultas baseado em programação dinâmica [Sel79].

Este processo, no entanto, pode apresentar problemas sérios de desempenho frente à situações características do domínio de serviços na Web. Em primeiro lugar, as fontes de dados estão sujeitas a grandes variações no tempo de resposta à consultas, principalmente nos horários de pico quando concentram-se os acessos dos navegadores potenciais. Adicionalmente, as constantes modificações levadas a cabo pelos responsáveis pelos *sites web*, tanto em seu conteúdo quanto em sua forma, tornam extremamente difícil, ou mesmo impossível, obter informações estatísticas sobre os dados disponibilizados. Essas duas características fazem do processo de geração pelo otimizador de um PEC rígido, inadequado para o domínio.

Tal condição particular levou os pesquisadores a buscarem alternativas com variado grau de dinamismo na geração e execução de planos de consultas sobre sites web: Query Scrambling [AFTU96], Eddies[AH00], Tukwila[IFFLW99], Hybrid[BPFV01] etc…

Este trabalho analisa profundamente as características de uma máquina de execução de consultas (MEC) capaz de executar consultas de maneira inteiramente adaptativa. Neste modelo, a máquina de execução recebe como entrada um plano inicial de consulta (PIC) com ordem parcial entre seus operadores. Obedecendo às restrições impostas pelo PIC, um operador especial fica responsável pela distribuição de tuplas aos operadores mais capacitados a execução naquele

momento. Considerando-se consultas compostas apenas de *Bindjoins*, além de seleções e projeções, cuja semântica de execução exclui problemas de assimetria (i.e. especificação das relações internas e externas em uma junção), tem-se total liberdade para alocar, em tempo de execução, tuplas aos operadores previstos no PIC. Com isso, obtém-se uma estratégia de execução onde diferentes planos são dinamicamente executados.

O resultado desta análise sobre as características de execução de consultas levou a construção de um framework capaz de suportar diferentes modelos de execução, adaptativos ou não, permitindo sua configuração para atender a diferentes domínios de aplicação. Este framework foi implementado e integrado ao ambiente CoDIMS [BP01], um middleware configurável para integração de dados heterogêneos e distribuídos, baseado em componentes e frameworks.

Neste trabalho instanciamos a MEC segundo o modelo adaptativo de modo a suportar consultas de integração como a apresentada acima.

1.1 Formulação do Problema

Nesta seção detalhamos o problema motivado pela consulta de integração em *sites web*. Consideramos três *sites* com informações publicadas na web: Shows, CD e Rank. Os esquemas relacionais exportados pelos respectivos *wrappers* são: Shows(banda, cd, gênero), CD(cd, preço) e Rank(banda, cd, posição). O primeiro apresenta uma lista de grupos musicas com apresentações na cidade. O segundo publica preços de CDs e o terceiro apresenta os *n* CDs mais vendidos do momento.

Uma aplicação de integração que objetive recuperar os preços de CDs das bandas com apresentação na cidade e que estejam na lista dos mais vendidos seria expressa, em SQL, pela consulta abaixo:

```
Select cd.cd, cd.preço
From Shows s, CD , Rank r
Where   s.genero = 'Rock' and  s.cd =  cd.cd and cd.cd = r.cd
```

Exemplo 1 Consulta de integração

O predicado *p*=(*genero='Rock')* exemplifica a necessidade do fornecimento de um critério de busca para acesso aos dados publicados pelo site Shows. Neste caso, o site requer a associação de um gênero musical para que se tenha acesso aos Shows de bandas daquele gênero. A partir desta associação, define-se uma ordem parcial entre os operadores que comporão o plano de execução da consulta Exemplo 1. Tem-se, para árvores profundas à esquerda, as seguintes possíveis ordens de avaliação dos predicados: O_1= *Shows* $\rightarrow$ *CD* $\rightarrow$ *Rank* e O_2= *Shows* $\rightarrow$ *Rank* $\rightarrow$ *CD*. Outros planos possíveis incluiriam versões para planos *bushy*.

A determinação do melhor plano a ser executado neste cenário é uma tarefa difícil considerando-se as grandes variações no tempo de resposta aos sites envolvidos. Estratégias puramente estáticas não são capazes de modelar tais variações. Já estratégias adaptativas podem combinar os diversos planos possíveis durante a execução de uma consulta recorrendo aos operadores que estejam mais disponíveis a cada instante. No Exemplo 1, uma máquina adaptativa realizaria um plano equivalente à $O_1 \cup O_2$.

Neste trabalho, são discutidas alternativas de implementação de máquinas de execução de consultas (MEC) que atendem a requisitos de diferentes domínios de aplicação. Em particular, implementa-se uma MEC que atende a aplicações com as características apresentadas no Exemplo1.

O restante deste artigo encontra-se dividido da seguinte forma. Na Seção 2 apresentam-se os trabalhos relacionados. Em seguida, a Seção 3 apresenta o ambiente CoDIMS de sistemas configuráveis. Na Seção 4 é realizada uma análise dos Modelos de execução de Consulta. A seguir, na Seção 5, descreve-se o framework para implementação de uma máquina de execução configurável. A Seção 6, instancia o framework para uma máquina de execução adaptativa. Finalmente, a Seção 7 apresenta as conclusões finais.

2 TRABALHOS RELACIONADOS

O desenvolvimento de sistemas para integração de dados heterogêneos e distribuídos é um problema atual e complexo, considerando a existência de diferentes perfis e interesses dos usuários, tipos e modelos de dados diversos, que requerem diferentes funcionalidades para os sistemas de integração.

Na literatura são encontrados diversos sistemas voltados para integração, mas que são demasiadamente específicos, atendendo a uma única aplicação, ou extremamente genéricos para atender às diversas situações de integração tornando-se sistemas *heavy weight* [BT95]. Ambas situações levam a problemas quando da necessidade de adaptá-las a uma nova aplicação.

Com o intuito de permitir que os sistemas passam a atender uma grande variedade de aplicações com requisitos diferentes, foram propostos sistemas configuráveis e extensíveis. A configuração permite customizar funcionalidades já existentes para o atendimento de uma aplicação específica enquanto que a extensibilidade facilita a incorporação de novas funcionalidades. A solução mais eficiente é projetar e implementar sistemas denominados *light weight* [BT95] que sejam flexíveis e que possam ser configurados para poder atender aos diferentes requisitos das aplicações, de forma que somente o código necessário a uma aplicação específica seja disponibilizado, sendo, então, denominados sistemas enxutos ou de aplicação específica.

Uma abordagem para desenvolvimento destes sistemas propõe que os mesmos sejam construídos a partir de uma arquitetura comum e a utilização de um mecanismo de integração de componentes, que possibilite disponibilizar uma família de sistemas com um menor número possível destes componentes para cada aplicação específica e, principalmente, permitindo o reuso de componentes já existentes. Para o atendimento destes objetivos, a utilização da técnica de *framework* [FS97] [John97] se mostra natural, por oferecer uma infra-estrutura adequada para a solução destes problemas. Framework é uma tecnologia de engenharia de software para produzir uma arquitetura semi-completa e reusável que pode ser especializada para produzir aplicações customizadas.

3 O AMBIENTE CODIMS

O CoDIMS é um ambiente orientado a objetos, baseado em frameworks e componentes, cujo objetivo é gerar sistemas middleware configurados e flexíveis para integração de dados. Tais sistemas são responsáveis por fornecer acesso integrado aos dados que estão armazenados de forma distribuída em fontes de dados heterogêneas.

A essência do ambiente CoDIMS é um componente de Controle que disponibiliza mecanismos para configuração física e lógica de componentes. A configuração física é realizada através da seleção e integração de um conjunto adequado de componentes, que implementam serviços middleware de integração de dados, denominados DIMS (Data Integration Middleware Services). Componentes DIMS são baseados em serviços típicos de gerência de dados de forma análoga aos existentes nos SGBDs, como Gerência de Metadados, Processamento de Consulta, Gerência de Transações, Controle de Concorrência, Gerência de Regras e Comunicação. A configuração lógica representa a seqüência de execução dos serviços DIMS para um sistema configurado específico, sendo baseada no conceito de workflow.

Para permitir flexibilidade, cada componente é implementado como um framework, com *hot-spots* [FSJ99] a serem instanciados. Através do reuso ou adaptação de um componente, o ambiente CoDIMS permite gerar sistemas de integração de dados heterogêneos de acordo com os requisitos de uma aplicação específica.

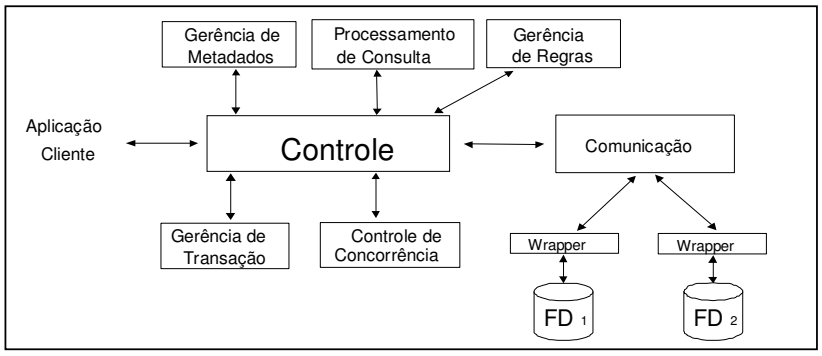

Fig. 1 Arquitetura do CoDIMS

3.1 O Processamento de Consultas

O processamento de consultas no CoDIMS é implementado pelo componente homônimo. Dele participam as atividade tradicionais de: análise semântica e sintática; otimização e execução de consultas. O componente interage com o de *Comunicação* para acesso às fontes de dados e *Gerência de Metadados* para obtenção de dados sobre os esquemas exportados.

4 MÁQUINA DE EXECUÇÃO DE CONSULTAS

Uma máquina de execução de consulta é responsável pela execução de um PEC fornecido pelo otimizador. Consideraremos que uma máquina de consulta poderá executar tanto um plano completo quanto um fragmento de plano e que a topologia dos planos de execução é uma árvore profunda à esquerda onde as folhas são fontes de dados e os nós são operadores algébricos.

Nesta seção analisaremos diferentes modelos de execução de consultas e suas implicações no framework.

4.1 Preliminares

Um PEC é composto por um conjunto de operadores que se relacionam, segundo um modelo de execução, de forma a produzir o resultado de uma consulta enviada à máquina de execução. Operadores do plano físico podem ser classificados como algébricos e de controle. Operadores algébricos representam os algoritmos de implementação das operações algébricas correspondente ao plano lógico de consulta. Por sua vez, operadores de controle estabelecem a comunicação entre os operadores algébricos e exercem a importante função de implementarem o modelo de execução desejado. O desacoplamento dos operadores algébricos e de controle flexibiliza a implementação de diferentes modelos de execução de consulta e tornam os operadores algébricos menos complexos (*ligth weigths*) [Grae93].

Na próxima seção discutiremos sobre os principais pontos definidores do modelo de execução.

4.2 Características de execução

Um modelo de execução é determinado pela combinação de características de execução selecionadas de forma a atender aos requisitos da aplicação. Podemos classificar tais características

em três grupos principais: Transferência de dados, Escalonamento e Autonomia. A seguir detalharemos cada um deles.

4.2.1 Transferência de dados

O grupo Transferência de dados reune as características relacionadas com a transferência de dados entre os operadores. O modelo utilizado é o produtor-consumidor. Num plano de consulta cada operador deve exercer o papel de produtor e/ou de consumidor de dados. A transferência de dados é feita de um produtor para o seu consumidor.

4.2.1.1 Grau de Materialização

Determina o quão imediato será feita a transferência de dados. Existem três abordagens: Materialização Total, Materialização Parcial e Imediata. Na primeira, os dados produzidos são armazenados (durante um certo tempo) em disco até serem consumidos. Na segunda, os dados produzidos são transferidos temporariamente para um *buffer* de memória. Na última abordagem a transferência de dados do produtor para o consumidor é imediata.

4.2.2 Escalonamento

O grupo Escalonamento diz respeito à ordem na qual os dados trafegam bem como ao controle deste tráfego, ao longo da execução, ou seja, está relacionado à alocação de recursos no tempo.

4.2.2.1 Escalonamento de operadores

Os dados nascem nas fontes de dados e são propagados entre produtores e consumidores até atingirem o último operador consumidor produzindo os resultados da consulta. Dependendo da abordagem utilizada, o caminho percorrido pelos dados podem variar, ou seja, a ordem (escalonamento) dos operadores pode não ser a mesma para todos os dados.

Existem duas abordagens para o escalonamento de operadores: Fixo e Adaptativo. No Escalonamento Fixo (Figura 2b), o fluxo de dados segue uma ordem fixa, ou seja, o caminho percorrido pelos dados, a partir da sua fonte de dados até a saída dos resultados, quando possível, é sempre o mesmo. No Escalonamento Adaptativo (Figura 2a), o fluxo de dados segue uma ordem variável e o caminho percorrido pelos dados pode mudar dinamicamente.

O Escalonamento Adaptativo, por sua vez, possui duas abordagens: Adaptatividade por Operador ou por Fragmento. Na adaptatividade por Operador (Figura2b) o caminho percorrido por um dado é baseado no desempenho dos operadores e de um critério de distribuição. Por exemplo, um determinado dado poderá ser processado pelo primeiro operador disponível ou pelo operador que já processou mais dados - os operadores mais rápidos processam mais dados do que os demais. Esta é a abordagem usada pelo *Eddies* [AH00].

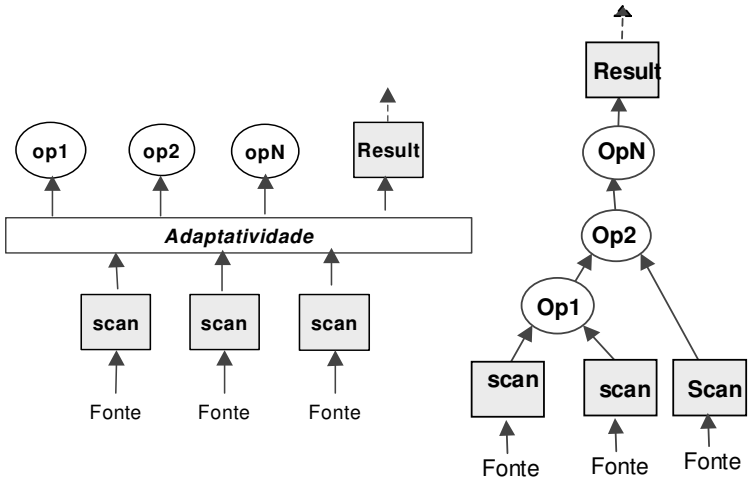

Fig. 2 (a) escalonamento adaptativo e (b) escalonamento fixo

Na adaptatividade por Fragmento considera-se a possibilidade de re-otimização através da intercalação entre a execução do otimizador e da máquina de consulta. Neste caso, o otimizador gera fragmentos de um plano numa ordem parcial e, no final da execução de cada fragmento, é chamado para re-otimizar os demais fragmentos estabelecendo novas prioridades segundo uma avaliação naquele momento. Um fragmento pode gerar dois tipos de eventos: (i) normal, quando o fragmento termina de ser executado, e (ii) anormal, quando algum problema imprevisível ocorre (*timeout*, falta de memória) abortando sua execução. Sendo assim, a máquina deve possuir um tratator de eventos. Esta abordagem é usada em [BFMV00] e [IFFLW99].

4.2.2.2 Escalonamento de dados

O Escalonamento de Dados determina a prioridade do resultado. Existem duas abordagens: (i) Resposta Inicial – a máquina de consulta deve fornecer os resultados da consulta assim que os dados forem sendo processados i.é., antes mesmo que toda a consulta seja executada; (ii) Resposta Final – a máquina de consulta deve fornecer o resultado da consulta somente quando todos os dados forem processados.

No escalonamento Fixo a ordem do plano é feita pelo otimizador para atender uma ou outra abordagem. No Escalonamento Adaptativo os dados são enfileirados durante a transferência entre vários produtores para vários consumidores. Neste contexto, o problema é determinar qual a ordem de enfileiramento dos dados. Para a Resposta Inicial, essa ordem deve ser baseada na quantidade de processamento já realizado, associando-se uma prioridade de execução a cada dado, tal como implementado em [AH00]. A segunda abordagem não requer ordenação.

4.2.2.3 Escalonamento de controle

A transferência de dados, entre um produtor e um consumidor, é controlada por um deles através de duas abordagens: Controle do Consumidor (*demmand-driven*) e Controle do Produtor (*data-driven*). Na primeira, um consumidor solicita um dado ao seu produtor que o envia assim que o produz. Na segunda, um produtor, na medida que produz o dado, o envia para o seu consumidor. Esses controles são implementados, respectivamente, através de operações *get* e *put*.

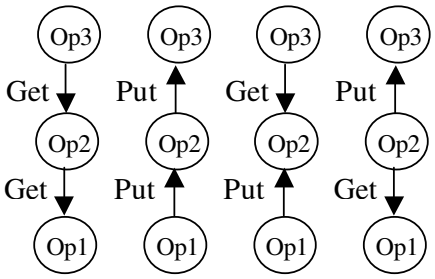

Fig. 3 (a-b) escalonamento uniforme e (c-d) escalonamento combinado

Do ponto de vista de um PEC, o escalonamento de controle diz respeito à propagação desses controles ao longo do plano. A propagação pode ser feita de forma Uniforme ou Combinada. Estas duas abordagens se dividem em quatro casos distintos. As figuras 3a, 3b, 3c e 3d ilustram estes casos. O primeiro deles refere-se ao Escalonamento Uniforme de Consumo (ou de demanda) onde a propagação do controle é feita de cima para baixo usando uma função *Get* (ver tabela). Esta abordagem é conhecido na literatura como modelo *Iterator* [Graefe93] e os seus operadores são chamados de *iterators*. O segundo caso refere-se ao Escalonamento Uniforme de Produção que propaga o controle de baixo para cima usando uma função *Put*. Em ambos os casos, é necessário que cada operador implemente uma interface padrão usando as funções descritas na tabela abaixo:

Open	Prepara o operador para produzir dados
Get	Produz um dado sob demanda do consumidor
Put	Produz um dado e envia para o consumidor
Close	Finaliza o operador

Tabela 1. Interface padrão

No terceiro e quarto casos combinamos os dois primeiros casos, ou seja, os Escalonamentos Uniformes coexistem no mesmo plano. Para isso, um operador deve assumir um dos seguintes papéis: Escalonador Ativo ou Escalonador Passivo [Grae93]. As suas funcionalidades são descritas a seguir:

a) Na figura 3c, op1, ao produzir um dado, o envia para op2 que o armazena consigo. Quando op3 solicitar um dado a op2 este dado será retornado para op3. Logo, combinamos o Escalonamento de Produção vindo de baixo até op2 com o Escalonamento de Consumo vindo de cima até op2. Portanto, op2 participa "passivamente" da combinação dos controles, intermediando op1 e op3.

b) Na figura 3d, op2 solicita um dado ao op1 que retorna para op2 que por sua vez o envia para op3. Logo, combinamos o Escalonamento de Consumo de op2 para baixo com o

Escalonamento de Produção de op2 para cima. Portanto, op2 participa "ativamente" da combinação dos controles, controlando a transferência de dados de op1 para op3.

4.2.3 Autonomia

O grupo autonomia refere-se ao grau de independência física e de processamento dos operadores do plano de execução de consulta.

4.2.3.1 Sincronismo

Quando um consumidor solicita dados ao produtor ou quando um produtor entrega dados ao consumidor, a comunicação se dá através da abordagem Síncrona ou Assíncrona. No modo Síncrono, um produtor e seu consumidor compartilham a mesma linha de execução (*thread*). Neste caso, o consumidor fica aguardando o produtor, ou seja, a continuidade do seu processamento depende do produtor retornar o dado. O modo Assíncrono, ao contrário do anterior, implementa um paralelismo entre dois operadores (consumidor/produtor ou produtor/consumidor) tornado-os independentes ou autônomos quanto ao seus processamentos. As duas abordagens podem coexistir num mesmo plano de execução.

4.2.3.2 Distribuição

Os operadores podem ser executados de forma Centralizada ou Distribuída. A distribuição depende do número de processadores e da arquitetura computacional como, por exemplo, arq. de memória compartilhada (*shared-memory)* e arq. sem compartilhamento (*shared-nothing)* [OV99]. A escolha da arquitetura apropriada está relacionada com a arquitetura do sistema gerenciador ou integrador de dados e com o tipo de paralelismo (próxima sub-seção) adotado pelo modelo de execução. É importante notar que existe outro tipo de distribuição que é a de Dados e é ortogonal a distribuição de Operadores, ou seja, um e/ou outro podem estar, ou não, num mesmo local.

4.2.3.3 Paralelismo

Para que exista paralelismo é necessário que dois operadores sejam executados em dois processos distintos. Existem dois tipos de paralelismo: Intra-operador e Inter-operador. O paralelismo Inter-operador envolve dois operadores distintos e o paralelismo Intra-operador envolve o mesmo operador A abordagem Inter-operador (figura 4a) permite ainda duas formas de paralelismo: o Paralelismo Vertical (*pipelined*) entre um produtor e um consumidor (ex: op1/op3 ou op2/op3) e o Paralelismo Horizontal (*bushy*) entre dois produtores

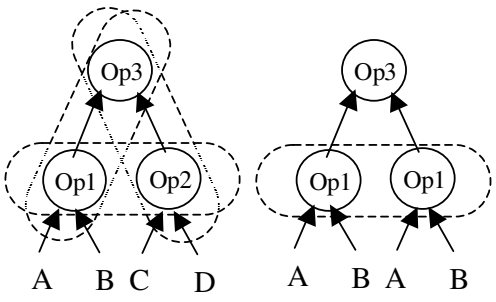

Fig. 4. Exemplos de Paralelismo inter-operador (a) e intra-operador (b) sobre as fontes A, B, C e D.

(ex: op1 e op2). A segunda abordagem, Inter-operador (figura 4b), ocorre quando um mesmo operador é replicado em vários processos distintos. No exemplo da figura 4b, o operador op1 foi duplicado em dois processos, compartilhando as fontes A e B e, em cada processo de op1, as fontes A e B podem ser consumidas integralmente ou parcialmente - havendo necessidade de fragmentação das fontes. No problema da seção 1, o uso de *wrappers* e *bindjoins* permite o paralelismo inter-operador e o uso de múltiplas instâncias de um mesmo *wrapper* permite o paralelismo intra-operador.

4.3 Modelos de Execução

Nesta seção, as características de execução discutidas na seção 4.2 são combinadas, de modo a definir um modelo de execução adequado à cenários sugeridos por domínios de aplicação. A Tabela 2 mostra alguns destes cenários onde cada uma das características ocorre em pelo menos em um cenário. A seguir apresentaremos os quatros cenários escolhidos.

4.3.1 Cenário 1

Este cenário é caracterizado principalmente pelas aplicações que envolvem transação longa havendo a necessidade de Materialização de resultados parciais durante a execução de um PEC. O Escalonamento Fixo é o mais apropriado para a execução possibilitando-se determinar pontos de materialização. Além disso, deve-se priorizar a Resposta Inicial e o Assincronismo.

4.3.2 Cenário 2

Este cenário é mais encontrado nos SGBDs comerciais. É caracterizado principalmente pelos Escalonamentos Fixo e Uniforme de Consumo (*iterator*). A configuração típica combina Materialização Parcial e Resposta Final.

4.3.3 Cenário 3

Este cenário aborda a situação-problema apresentada na seção 1. Neste caso, o modelo de execução deve lidar com a imprevisibilidade do tempo de acesso a *sites*. A configuração principal é o Escalonamento Adaptativo que produz a melhor ordem de execução do plano.

4.3.4 Cenário 4

É voltado para sistemas de tempo real onde as consultas são executadas continuamente. Com a chegada de novos dados, novos resultados são produzidos. A configuração mais importante é o Escalonamento de Produção onde cada produtor envia dados aos seus consumidores à medida que os dados são gerados - não necessitando de materialização.

Grupo	Característica de Execução	Configuração	Cenários			
			1	*2*	*3*	*4*
Transferência	Materialização da transferência	Total	*			
		Parcial		*	X	X
		Nenhuma		X		
Escalonamento	Escalonamento de Operadores	Fixo	X	*		X
		Adaptativo			*	X
	Escalonamento de Controle	Uniforme de Consumo	X	*		
		Uniforme de Produção				*
		Combinado			X	X
	Escalonamento de Dados	Resposta Inicial	*		X	X
		Resposta Final		X	X	
Autonomia	Sincronismo	Síncrono		X		X
		Assíncrono	X	X	X	X
	Distribuição	Centralizado	X	X		X
		Distribuído	X	X	*	X
	Paralelismo	Inter-operador	X	X	*	X
		Intra-operador	X	X	X	X

Legenda: (*)configuração mais relevante, (X)=configuração possível

Tabela 2. Exemplos de cenários de execução

5 ESPECIFICAÇÃO DO FRAMEWORK

Nesta seção apresentamos o framework proposto que generaliza e flexibiliza os modelos de execução baseados nos cenários descritos na seção anterior. O framework deve ser instanciado para cada modelo de execução, produzindo diferentes MEC. Desta forma, a configuração do framework suporta a construção de máquinas de execução para as diversas aplicações. Por exemplo: no caso do modelo de execução adaptativo (cenário 3) a máquina não precisa conhecer o escalonamento (de dados) do plano pois os operadores sofrerão adaptatividade. Fica a cargo do instanciador do framework (no caso, o CoDIMS) fixar ou parametrizar essas configurações. A configuração pode ser feita para cada execução ou para uma das aplicações.

Dependendo do modelo de execução da máquina instanciada, a sua gerência será mais *ligth weigth* (para cenários simples) ou mais *heavy weigth* (para cenários complexos). Vejamos a seguir as principais classes do framework.

A aplicação usuária utiliza o framework através da classe concreta MAQUINA, um *façade* [GHJV95] que disponibiliza vários métodos, tais como: *Execute,* para a submissão da consulta, e *First*, *Next* e *EOF* para interagir com os resultados. O método *Execute* recebe um PEC com os seguintes dados:

- A lista de operadores: ID (algoritmo), tipo (acesso, processo ou resultado), fonte de dados (quando acesso), predicados, etc.
- Ordem dos operadores (quando fixa) ou PIC (quando adaptativa)
- Configurações do modelo de execução adotado

A preparação da execução consiste na seleção de fontes envolvidas e na criação dos operadores pela classe FABRICA.

A execução é de responsabilidade da classe abstrata GERENTE-DE-EXECUCAO que é o principal ponto de flexibilixação (*hotspot*) do framework e deve ser instanciada de acordo com o modelo de execução adotado pela máquina de execução instanciada. Esta classe oferece algumas funcionalidade básicas tais como a inicialização dos operadores (algébricos de controle) através de seus métodos *open*. As demais funcionalidades estão na forma de métodos *template* [GHJV95].

Ao iniciar-se a execução, os operadores de acesso fornecem os dados de suas fontes de dados que são processadas pelos demais operadores e finalizadas pelo último operador. A finalização da execução da consulta é detectada pelo gerente através de uma tupla especial, denominada tupla *flag,* produzindo uma chamada aos métodos *close* dos operadores.

Quando o primeiro e último resultado são produzidos, a classe MAQUINA é sinalizada, permitindo ao usuário ter acesso aos resultados iniciais e finais, respectivamente.

O framework oferece a classe abstrata *FONTE* e a classe concreta METADADO contendo as operações sobre uma relação (métodos *template*[GHJV95]) e sobre os seus atributos (colunas).

Um operador é especificado na classes abstrata *OPERADOR*. Esta classe fornece os métodos *templates open, get, put e close* da interface padrão mencionada anteriormente. A classe Operador se especializa nas classes abstratas *OPALGEBRICO* e *OPCONTROLE* para separar a definição dos operadores algébricos e de controle, respectivamente. Um operador de controle é criado pelo gerente de execução e, dependendo do modelo de execução, pode incorporar várias características (sincronismo, distribuição e paralelismo). Logo, a classe *OPCONTROLE* deve ser configurada para se obter essas características. Outra configuração possível é o grau de materialização que pode ser nenhum, total (usando arquivos temporários), ou parcial (fila em memória).

6 ESTUDO DE CASO

Nesta seção instanciamos o *framework* para o Cenário 3, descrito na seção 4, produzindo uma máquina de execução adaptativa (Figura 5). Utilizamos como exemplo de aplicação usuária da máquina, o exemplo descrito na seção1. Foram instanciados três operadores algébricos *Project*, *Bindjoin* e *Scan*, e três operadores de controle assíncronos denominados Leitor, Coletor e Distribuidor. A função do Distribuidor é a de implementar o Escalonamento Adaptativo. Um Distribuidor é um Escalonador Passivo, ou seja, serve como intermediário entre os seus produtores (Coletores) e consumidores (Leitores). Um Coletor é um Escalonador Ativo que coleta tuplas de seu produtor e as envia para o Distribuidor. Um Leitor é um operador de controle assíncrono que implementa o paralelismo inter-operador e o Escalonamento de Consumo (iterator) entre cada operador algébrico.

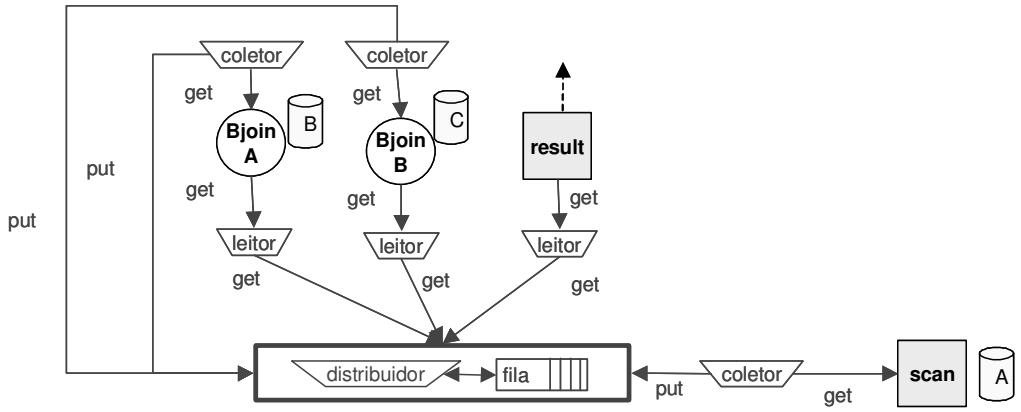

Fig. 5 Modelo de execução adaptativo

Como o modelo de execução prioriza a Resposta Inicial, foi implementada uma fila de tuplas com prioridade que aumenta cada vez que a tupla retorna para o Distribuidor. Desta forma, as tuplas com maior prioridade são projetadas para o resultado, antes que as de menor prioridade.

A partir do plano recebido contendo os operadores algébricos, a máquina de execução constrói o plano adaptativo. O *Scan* acessa a primeira fonte de dados criando tuplas a serem consumidas pelos *Bindjoins*. Estes, por sua vez, realizam *Bindings* sobre suas fontes associadas. As tuplas que completarem todos os *Bindjoins* são projetadas pelo *Project*, gerando o resultado final. Existem quatro Coletores: um coleta tuplas do *Scan* e os demais coletam tuplas dos *Bindjoins* retornando-as para o Distribuidor. Os quatro Leitores

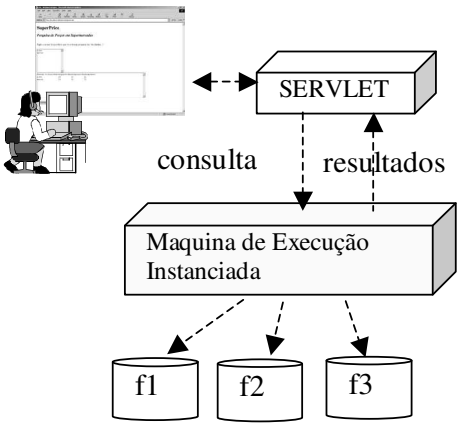

Fig. 6 Exemplo de aplicação

paralelizam a execução dos *Bindjoins* e do *Project*.

A figura 6 mostra um esquema de uso da máquina de execução instanciada. A implementação da máquina e da aplicação usuária foram feitas na plataforma *Java*.

7 CONCLUSÕES

Neste trabalho, apresentamos um máquina de execução de consultas adaptável para atender ao domínio de aplicações de integração de *sites web*. Tais aplicações apresentam desafios na medida em que o tempo de acesso às fontes de dados registram grandes variações e as informações estatísticas são escassas.

De modo a atender aos requisitos de imprevisibilidade, realizou-se um estudo aprofundado sobre as características de execução de máquinas de execução de consultas, de onde extraímos as principais funcionalidade para construção de uma máquina adaptativa. A partir deste estudo, desenvolveu-se um framework configurável integrado ao ambiente CoDIMS de geração de sistemas configurados de integração de dados. Implementamos uma máquina de execução de consultas adaptativa a partir da instanciação deste framework.

Este artigo apresenta as seguintes contribuições:

- Um estudo das características de execução de consultas
- Um framework de MEC
- Uma implementação de MEC adaptativa
- Uma extensão ao ambiente CoDIMS para execução de consultas na web.

BIBLIOGRAFIA

[AFTU96] L. Amsaleg, M. J. Franklin, A. Tomasic e T. Urhan, "Scrambing Query Plans to Cope with Unexpected Delays", em PDIS Conf., Miami, USA, 1996.

[AH00] R. Avnur e J. Hellerstein, "Eddies: Continuously Adaptive Query Processing", em Proc. Intl. Conf. on Management of Data, Dallas, Texas, 2000.

[AZ96] G. Antoshenkov e M. Ziauddin, "Query Procesing and Optimization in Oracle Rdb", em VLDB Journal, Vol. 5, Num. 4, 1996.

[BFMV00] L. Bouganim, F. Fabret, C. Mohan e P. Valduriez, "Dynamic Query Scheduling in Data Integration Systems", em Proc. 16o. International Conference on Data Engineering, Sao Diego, CA, Março , 2000.

[BFPV01] L. Bouganim, F. Fabret, F. Porto e P. Valduriez, "Processing Queries with Expensive Functions and Large Objects in Distributed Mediator Systems", em Proc. 17o. Intl. Conf. on Data Engineering (ICDE), Heidelberg, Alemanha, Abril 2001.

[BP01] Barbosa, A.C.P. & Porto, F.A.M. “Configurable Data Integration Middleware System”. Proceedings of International Workshop on Information Integration on the Web - Technologies and Applications (WIIW2001). Brazil, Abril 2001.

[BT95] Batory, D.S. & Thomas, J. “P2: A Light Weight DBMS Generator”. Technical Report TR-95-26, Department of Computer Sciences, University of Texas at Austin, Agosto 1995. http://www.cs.utexas.edu - 10/04/1998.

[FMLS99] D. Florescu, I. Monolescu, A. Levy e D. Suciu, “Query optimization in the Presence of Limited Access Patterns”, em Proc. Intl. Conf. on Management of Data, Filadelfia, USA, Junho, 1999.

[FSJ99] Fayad, M.E.; Schmidt, D.C. & Johnson, R.E. “Building Application Frameworks – Object-Oriented Foundations of Frameworks”. John Wiley & Sons, Inc. 1999.

[GHJV95] E. Gamma, R. Helm, R. Johnson e J. Vlissides, “Design Patterns”, em Addison-Wesley, 1995.

[GPQ+97] H. Garcia-Molia, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagin, J. Ullman and J. Widom, “The TSIMMIS Project: Integration of Heterogeneous Information Sources”, em Journal of Intelligent Information Systems, 8(2), pp 177-132, Março, 1997.

[Grae93] G. Graef, “Query Evaluation Techniques for Large Databases”, em ACM Computing Surveys, 25(2), Junho, 1993.

[IFFLW99] Z. G. Ives, D. Florescu, M. Friedman, A. Levy e D. Weld, “An Adaptive Query Execution System for Data Integration”, em Proc. Intl. Conf. on Management of Data, Filad\ee lfia, USA, Junho, 1999.

[LeSe] “A Mediator System Developed in the Caravel Project”, INRIA, França, em http://caravel.inria.fr/Fprototype_LeSelect.html.

[OV99] M. T. Ozsu e P. Valduriez, “Principles of Distributed Database Systems”, em Prentice Hall, Nova Jersey, USA, 1999.

[Sel79] Selinger P., et al, “Access path selection in a relational data base management system”, "Proc. Intl. Conf. on Management of Data, Maio, Boston, Massachusetts, USA, 1979.