7
Conclusão

Neste trabalho propomos um novo modelo de execução de consulta criado para suportar o processamento de aplicações iterativas. Em especial, propõe-se uma implementação baseada em uma arquitetura de execução paralela, que faz uso dos nós de uma grade computacional para reduzir o tempo de processamento das consultas. Uma vez que dependendo do volume de dados a ser processado e do número de iterações a serem realizadas o tempo de execução seqüencial pode-se aferir bastante grande.

O modelo proposto é baseado no desenvolvimento de um novo operador de controle do fluxo de dados, conhecido como Orbit. Este operador é responsável por introduzir as tuplas no processo iterativo e controlar o número de iterações realizadas por cada uma destas.

Para viabilizar a execução paralela, utiliza-se o algoritmo de escalonamento G2N que é responsável por escolher o conjunto de nós a serem utilizados e determinar uma distribuição de dados ideal, de forma a minimizar o tempo de execução. O algoritmo de escalonamento G2N é estendido com um novo modelo de custo que considera a natureza iterativa do processo.

No contexto deste trabalho considera-se que o volume de dados disponível é conhecido e que os dados encontram-se disponíveis no início da execução. Estes dados são divididos entres os nós alocados pelo algoritmo de escalonamento G2N e, dependendo da quantidade de dados a ser processada por cada nó e sua taxa de comunicação com o nó de controle, três cenários de custo podem ser vislumbrados: transparência total, transparência parcial e execução seqüencial.

A fim de garantir que a execução aconteça no tempo previsto são utilizadas técnicas adaptativas que permitem ao sistema adaptar o grau de paralelismo utilizado e a distribuição dos dados de acordo com o volume de dados disponível e a vazão (*throughput)* de cada nó.

O volume de dados durante a execução de consultas no modelo Orbit pode variar, uma vez que algumas tuplas podem completar seu número de iterações

primeiro que as outras ou podem ser eliminadas durante o cálculo de sua órbita. Nestes cenários é importante que o nível de paralelismo e a divisão dos dados seja revista para evitar o problema conhecido como disputa por dados.

Os resultados experimentais demonstram os aspectos positivos da solução apresentada, bem como oportunidades de trabalhos futuros delineados na seção seguinte.

7.1. Trabalhos Futuros

Nesta seção encontra-se uma breve descrição de algumas melhorias e pontos de extensão que podem ser realizadas no contexto deste trabalho:

1. Criação de uma linguagem para definição dos planos de execução que suportasse o conceito de módulo.
2. Integração da máquina de execução de consultas com algum sistema que ofereça a taxa de comunicação entre os nós de controle e de execução dinamicamente, ou desenvolvimento de algum mecanismo que permita criar uma estimativa do mesmo. A disponibilidade desta informação permitiria uma maior capacidade de adaptação do ambiente.
3. Definição de Δ para os casos em que custo de processamento de uma tupla pelo nó de execução é aproximadamente igual ao custo de transmissão. Neste cenário o bloco de comunicação utilizado possuirá tamanho unitário e, o desempenho prejudicado em função da constante interrupção para entrada e saída.
4. Desenvolvimento de estratégias de adaptação que permitam a reação local dos nós de execução. Neste sentido seria possível adaptar o plano executado segundo a seletividade de suas operações ou a disponibilidade de recursos.
5. Alteração da política de distribuição de blocos para *PUT*. Na atual versão, foi utilizada uma política de distribuição baseada em demanda (*GET)* na tentativa de definir um mecanismo de detecção da taxa de comunicação. Tal mecanismo não se mostrou confiável e esta política introduz alguns problemas na utilização de operadores assíncronos no fragmento de execução iterativo. Nestes casos os operadores assíncronos tendem a

consumir todos os dados disponíveis fornecendo a impressão de que os dados já foram processados, induzindo o fragmento a consumir mais tuplas.

6. Desenvolvimento de mecanismos que possibilitem tolerância à falhas completa ou parcial. Com este mecanismo seria possível recuperar a execução quando um nó falhar ou apresentar desempenho muito abaixo do esperado através da redistribuição das tuplas enviadas aquele nó.
7. Adição de uma fase de *probe* na inicialização do sistema. Essa fase permitiria ao sistema evitar situações em que o desempenho de um nó foi superestimado. Nestes casos muitas tuplas podem ser enviadas a um nó e prejudicar a adaptação do sistema.
8. Desenvolvimento de uma versão distribuída do modelo orbit, assim as tuplas não precisariam voltar ao nó de controle para serem distribuídas, apenas os resultados intermediários seriam enviados. Nesta versão, um mecanismo de devolução de tuplas deve ser implementado para possibilitar a adaptação.