Avaliação dos Recursos de Programação de Anahy Através dos Problemas Cowichan

Leonardo R. Korndorfer,
Otávio C. Cordeiro, Gerson Geraldo H. Cavalheiro

Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950 - Bairro Cristo Rei - CEP 93.022-000 São Leopoldo - RS - Brasil
leokorndorfer@gmail.com, otaviocc@turing.unisinos.br, gersonc@unisinos.br

Introdução

Aglomerados de computadores representam hoje a classe de arquitetura mais popular dentre aquelas utilizadas para o processamento de alto desempenho. Este tipo de arquitetura é composto por um conjunto de nodos de processamento, normalmente multiprocessados, interconectados por uma rede rápida de comunicação. Neste sentido, a arquitetura oferece dois níveis para exploração de paralelismo: intra-nó, explorando a capacidade de execução paralela entre os processadores de cada nó, e entre-nós, explorando a capacidade de execução paralela entre os diferentes nós que compõe o aglomerado. Neste contexto, a complexidade de programação esta associada ao mapeamento da concorrência das aplicações em programas paralelos contemplando estes dois níveis de paralelismo. Esta afirmação é particularmente verdadeira quando o enfoque da implementação é dado para exploração de aglomerados para processamento de alto desempenho.

Tradicionalmente, programadores lançam mão de ferramentas clássicas de programação, como as apresentadas em, para composição de seus programas paralelos em aglomerados. No entanto, estas ferramentas foram desenvolvidas com propósitos de permitir a exploração de recursos pontuais de processamento, como processadores e rede de comunicação. Desta forma, a atividade de desenvolvimento de aplicações paralelas também requer esforço do programador na definição da forma de como a concorrência da sua aplicação deve ser descrita por um programa paralelo. Como alternativa, diversos ambientes de programação têm sido propostos com vistas a fornecer recursos de programação mais próximos às necessidades dos programadores, e a literatura é farta em apresentações destes ambientes e discussões sobre seus desempenhos [BAL 92, COR 05, CAL 98]. No entanto, são poucos os trabalhos, como o apresentado em [WIL 94], que têm por objetivo avaliar a aplicabilidade de tais ambientes.

Este trabalho tem como objetivo discutir a aplicabilidade dos recursos de programação propostos por Anahy [COR 05]. O conjunto de aplicações selecionado para esta avaliação é denominado Problemas Cowichan, aplicados para avaliação da aplicabilidade do ambiente Orca [WIL 94].

A próxima sessão descreve os recursos básicos de programação em Anahy e a seqüência apresenta os Problemas Cowichan. Um estudo de caso é apresentado ao final.

Anahy

A interface de programação aplicativa (API) de Anahy foi projetada para oferecer primitivas baseadas no modelo de programação multithread. A opção tomada foi adotar um subconjunto do padrão POSIX para threads.

Definição do escopo de uma thread O corpo de uma thread é definida como uma função C convencional, como mostrada no exemplo abaixo:

```
void * func( void * in ) { /* código */ return out; }
```

Syncronização de Thread A sintaxe das operações *fork* e *join* correspondem a criação e sincronização de threads POSIX: `pthread_create` e `pthread_join`. A sintaxe dessas operações é exemplificada por:

```
athread_create(athread_t *th, athread_attr_t *atrib,
               void *(*func)(void *),
               void *in);
athread_join(athread_t th, void **res);
```

Nestes exemplos `athread_create` cria uma nova thread para executar a função definida por `func`; terá como entrada o dado localizado no endereço de memória localizado especificado em `in`. O parâmetro `*th` será atualizado com o identificador da nova thread criada. O valor dado por `*atrib` consiste dos atributos de execução providos pelo programador, para dar informações acerca de características particulares da nova thread. Na operação de `athread_join` a thread que deseja-se sincronizar é identificada por `th`, sendo `res` atualizado com o ponteiro para a posição de memória compartilhada onde a saída da função executada pela thread `th` pode ser encontrada.

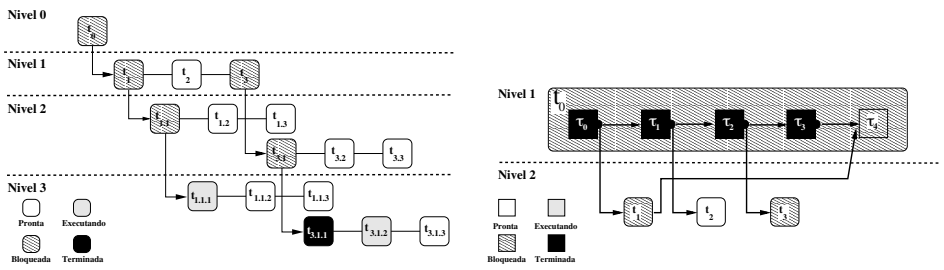

Figura 1: Representação do ambiente Anahy. a) Grafo de dependência de dados entre tarefas Anahy, e b) Grafo de precedência entre threads Anahy.

A Figura-1 exemplifica como Anahy realiza o controle da sincronização do programa, tanto ao nível de threads (Figura-1b), visível pelo programador, como no nível das tarefas (Figura-1a), visão do sistema.

Problemas de Cowichan

O conjunto de problemas de Cowichan foi proposto por G. V. Wilson [CAL 98] para avaliar o ambiente de programação Orca na composição de programas paralelos. Este conjunto de problemas é composto, originalmente, por sete aplicações, onde diferentes aspectos relacionados a concorrência são abordados.

Dynamic Programing: Um programa para multiplicar N matrizes M_i, $0 < i < N$, cada uma com r_i linhas e c_i colunas. Se a dimensão das matrizes variarem, a ordem na resolução das multiplicações pode ter diferentes custos de processamento.

Turing Ring: Alan Turing analisou a interação de dois produtos químicos em um anel de pilhas usando um par de equações diferenciais para descrever o sistema. Analogamente este problema consiste em avaliar o desenvolvimento de um sistema.

Skyline Matriz Solver: Tendo uma matriz $N \times N$ onde existem constantes r_i e c_i, sendo $1 < r_i < i$, $1 < c_j < j$, a linha i tem valores diferentes de zero nas colunas r_i até i. A análise gira em torno da capacidade do sistema em multiplicar e analisar a melhor forma para o calculo da matriz final.

Image Thinning: A entrada do problema será uma imagem bi ou tri-dimensional que contem segmentos de linhas a pouca distancia uma da outra. A imagem deve estar corrompida, com alguns pixels faltando. Esquelatiza-se a imagem e após nomina-se os componentes conectados a imagem.

Polygon Overlap and Display: Exercita as capacidades de I/O dos sistemas, foi formulado para geração, filtragem e combinação de dados. Este problema realiza a fusão de duas figuras, gerando uma terceira figura com aspectos das originais combinados.

Constructing Crosswords: Este problema foi incluído para testar a habilidade do sistema para pesquisa assíncrona e especulativa. A idéia é construir de forma paralela uma aplicação que encontre a solução do jogo, sendo o tabuleiro um quadro de $N \times N$. O jogo possui W palavras especificadas pelo usuário.

Active Chart Passing: Resolver ambigüidades é um problema semântico mas depende da habilidade de gerar estruturas representando diferentes formas de se passar esta informação. Baseia-se em gerar, de forma eficiente, todas as possíveis derivações de uma frase dada uma gramática ambígua.

Estudo de Caso

Como estudo de caso foi escolhido o problema denomidado *Active Chart Parsing* (ACP). ACP, já introduzida na seção anterior, tem seu uso comum na área de Linguagens Naturais, onde é largamente utilizado para verificiar se uma dada frase foi gerada aleatorialmente ou pertence a alguma linguagem.

O problema, especificamente, basea-se em dois pilares fundamentais, um dicionário e uma gramática, mostrada na Figura-2. O primeiro define um conjunto de palavras que podem vir a fazer parte de uma frase. Cada palavra é classificada morfologicamente: sujeito, verbo, adjetivo, preposição e artigo. A gramática define a linguagem, a estrutura gramatical da língua e como as palavradas irão se estruturar formando frases coerentes.

Para grandes gramáticas, e frases que podem ser geradas, o desgaste computacional torna-se excessivo, sendo necessárias técnicas que explorem a concorrência da

Frase	SujeitoFrase × VerboFrase	Frase	VerboFrase
SujeitoFrase	determiner × SujeitoFrase	SujeitoFrase	SujeitoFrase2
SujeitoFrase	SujeitoFrase × PrepFrase	SujeitoFrase2	Sujeito
SujeitoFrase2	adjetivo × SujeitoFrase2	PrepFrase	SujeitoFrase
VerboFrase	verbo	VerboFrase	verbo × SujeitoFrase
VerboFrase	VerboFrase × PrepFrase		

Figura 2: Gramática utilizada na aplicação

aplicação, em outrora, minimizando os custos com o uso da biblioteca de *PAD* Anahy.

Conforme apresentado por Winograd [WIN 83], o problema pode ser abordado com o uso de duas estratégias: *bottom-up* e *top-down*. O ACP em questão faz uso da primeira estratégia, buscando a concorrência da aplicação na estrutura da gramática definida. Uma vez que se conheça certa parte da frase, como por exemplo um *SujeitoFrase* o resultado pode ser armazenado para determinar as categorias de ordem superior.

Conclusão

O presente trabalho apresentou uma visão geral de threads Anahy, um ambiente de exploração de processamento de alto desempenho em aglomerados de computadores e em arquiteturas SMP, assim como uma breve explicação dos problemas de Cowichan. O problema de Active Chart Parsing foi ultilizado como estudo de caso por explorar a concorrência em árvore nas construções gramaticais e assim gerar um resultado de forma paralela. As próximas etapas tem por objetivo implementar e estudar os outros problemas afim de definir os campos da concorrência onde Anahy se torna bom, ou superior, e para qual destes obteve seu melhor resultado.

Referências

[BAL 92] BAL, H.E.; KAASHOEK, M. F.; TANENBAUM, A. S.; LEVELT, W. G.; **A comparison of two paradigms for distributed shared memorys** - IEEE Trans. on Software Engineering, 1992

[CAL 98] GALILÉE, F.; ROCH, J.; CAVALHEIRO, G. G. H.; DOREILLE, M.; **Athapascan-1: On-Line Building Data Flow Graph in a Parallel Language**, Proceedings of the 1998 International Conference on Parallel Architectures and Compilation Techniques, 1998

[COR 05] CORDEIRO, O. C.; PERANCONI, D. S.; VILLA REAL, L. C.; DALL'AGNOL, E. C.; CAVALHEIRO, G. G. H.; **Exploiting Multithreaded Programming on Cluster Architectures**, 19th International Symposium on High Performance Computing Systems and Applications (HPCS), 2005.

[WIL 94] WILSON, G. V.; **Accesing the Usability of Parallel Programing Systems: The Cowichan Problems**, In Procceding of the IFIP Working Conference on Programing Environments for Massively Parallel Distributed Systems, Birkhäuser Verlag AG, April, 1994.

[WIN 83] WINOGRAD, T.; **Language as a Cognitive Process**. Addison-Wesley, 1983.