Avaliação de Desempenho de um Ambiente de Programação Paralelo *

Lucas Correia Villa Real†
Gerson Geraldo H. Cavalheiro

Programa de Pós-Graduação em Computação Aplicada – PIPCA
Centro de Ciências Exatas e Tecnológicas
Universidade do Vale do Rio dos Sinos
São Leopoldo – RS – Brasil
{lucasvr, gersonc}@exatas.unisinos.br

Introdução

A exploração de arquiteturas SMP (Symmetric Multi-Processor) para suporte à execução de programas concorrentes motiva a implementação de programas *multi-thread*, devido ao potencial de exploração dos recursos do hardware que esta ferramenta de programação atinge [CAV 2001]. No entanto, devem ser consideradas a definição das tarefas a serem criadas, o número de *threads* a serem lidadas pelo sistema operacional e critérios de balanceamento de carga, desviando muitas vezes o enfoque do problema para a obtenção de aplicativos que explorem eficientemente uma arquitetura multi-processada.

Anahy é um ambiente de execução sobre agregados de computadores que oferece para o programador uma interface para a criação de programas concorrentes. Esta interface foi definida de forma a ser compatível com o padrão POSIX 1003.1, que define a interface e os serviços a serem oferecidos para *threads*. Diferente do uso manual de bibliotecas para *threads*, Anahy adota um modelo de programação que permite que o programador preocupe-se apenas em determinar a concorrência de sua aplicação, sem a necessidade de especificar como o paralelismo da arquitetura será explorado. Ao ambiente de execução fica o encargo de escalonar as tarefas criadas peo programador, de forma a manter todos os processadores ativos na arquitetura real.

Este trabalho apresenta o impacto no desempenho de uma aplicação desenvolvida sobre o ambiente Anahy. Uma análise das decisões tomadas no rumo deste projeto também é realizada.

Arquitetura do Ambiente

O ambiente Anahy é estruturado em três camadas principais, denominadas Interface Aplicativa (API Anahy), Núcleo Executivo e Máquina Abstrata [REA 2002]. A arquitetura do ambiente permite disponibilizar ao programador um esquema de *threads* "super-leves"[LOW 99]. Neste esquema, as *threads* criadas pelo programa em execução não são mapeadas diretamente em *threads* efetivas, mas tratadas como tarefas executadas por um *pool* de execução. Com esta solução, o núcleo executivo é capaz de realizar

*Apoio: CNPq, FAPERGS, UNISINOS
† ITI-CNPq

Tabela 1: Quantidade de tarefas associadas ao cálculo de elementos da série de Fibonacci

Fibo	15	16	17	18	19	20
Tarefas	1218	1972	3192	5166	8360	13528

o mapeamento da concorrência descrita pelo programa nos recursos de processamento disponíveis na arquitetura.

Interface Aplicativa Oferece uma interface para desenvolvimento de programas. Os recursos disponíveis permitem a descrição da concorrência de uma aplicação sob a forma de atividades concorrentes, denominadas tarefas, capazes de se comunicar via parâmetros de entrada e retorno de resultados.

Núcleo Executivo Esta camada visa retirar do programador o compromisso de explorar o paralelismo real da arquitetura. Ela implementa o mecanismo de escalonamento de tarefas, sendo dotado de recursos para o balanceamento de carga. Este núcleo foi modelado de forma a permitir a adequação da política de escalonamento às características tanto do programa que deverá ser executado quanto da arquitetura (abordagens semelhantes em [BLU 95, GAL 98]).

Máquina Abstrata Nesta plataforma é construído um agregado *virtual*, onde o papel do processador é exercido por *threads* e o compartilhamento de dados entre processadores executando em diferentes nodos é garantido através de uma biblioteca de comunicação. Estas unidades processadoras representadas pelas *threads* são também chamadas *processadores virtuais*.

Escalonamento de Tarefas

A interface de programação de Anahy oferece recursos para a implementação de programas concorrentes. Nesta interface, cabe ao programador descrever as atividades concorrentes de seu programa e as dependências de dados existentes entre as tarefas. Estas informações permitem a criação de um grafo de dependências, que serve como base para que o ambiente realize o escalonamento e a busca por tarefas. O escalonamento de tarefas é feito respeitando a ordem de dependência representada no grafo.

Um fator importante e decisivo na obtenção de bons desempenhos no ambiente é a maneira na qual as buscas e o escalonamento são feitos neste grafo. Em outras palavras, como os recursos de hardware serão explorados para executar de forma eficiente o programa. Este grafo cria novos nodos à medida em que encontra definições de tarefas determinadas pelo programador. Como estas tarefas obedecem uma ordem de dependência, é possível utilizar técnicas de caminhamento neste grafo que levem em consideração a localidade destes nodos. Alguns testes foram realizados para determinar variações de desempenho sobre o ambiente. Em um caso específico analisado, foram obtidos tempos que variaram entre 4 e 1200 segundos. Esta variação é justificada, em parte, pela diminuição da sincronização entre os processadores virtuais. A relação entre o tamanho do grafo e a procura por determinadas tarefas (no caso do *join*, por exemplo) determina também grande parte do tempo ocupado pelo ambiente em operações sobre este grafo. Em especial em aplicações de grande quantidade de tarefas, tal a implementação realizada do algoritmo de Fibonacci, utilizada para avaliação do ambiente construído.

Tabela 2: Tempos em segundos para o cálculo dos elementos de Fibonacci com Pthreads

Fibo	Tempo (mono-processado)	Tempo (bi-processado)
15	6.4387	3.1862
16	17.2131	10.1092

Tabela 3: Tempos para cálculos de elementos de Fibonacci em Anahy (Arq. Monoprocessada)

	Processadores Virtuais					
Fibo	1	2	3	4	5	10
15	0.059	0.044	0.042	0.057	0.066	0.086
16	0.090	0.077	0.084	0.101	0.092	0.136
17	0.188	0.172	0.167	0.164	0.163	0.171
18	0.478	0.408	0.409	0.303	0.336	0.346
19	2.112	1.609	1.583	0.929	1.179	1.103
20	8.461	6.490	5.448	3.636	4.315	5.129

Resultados de Desempenho

Os testes realizados foram baseados em implementações do algoritmo de Fibonacci. Este algoritmo foi escolhido pelo alto grau de concorrência associado à sua implementação recursiva. A Tabela 1 apresenta a quantidade de tarefas associadas ao cálculo de elementos da série de Fibonacci. As execuções foram realizadas em um computador Pentium III 1100 Mhz biprocessado com 256 Mb de RAM. As medidas foram feitas baseando-se em 100 diferentes iterações, e estão representadas em segundos. A implementação realizada apenas com o uso de Pthreads não pôde criar todas as tarefas associadas aos cálculos dos valores acima de 17, devido à restrições na quantidade de tarefas suportadas pelo sistema operacional. Os tempos obtidos na implementação baseada em Pthreads encontram-se na Tabela 2.

Os tempos obtidos com Anahy encontram-se nas Tabelas 3 (monoprocessador), 4 (bi-processador) e 5 (bi-processador). As Tabelas 3 e 4 permitem verificar a otimização na exploração da capacidade de processamento das arquiteturas. Controlando a ativação das tarefas, o núcleo de escalonamento otimiza o consumo de memória ao mesmo tempo que explora eficientemente os recursos de hardware, evitando sobrecustos oriundos de um número fluxos de execução superior à capacidade do hardware.

Já a Tabela 5 apresenta resultados que comprovam a eficiência do escalonamento através de um contra-exemplo. O algoritmo de escalonamento utilizado varre a lista de

Tabela 4: Tempos para cálculos de elementos de Fibonacci em Anahy (Arq. Bi-Processada)

	Processadores Virtuais					
Fibo	1	2	3	4	5	10
15	0.037	0.031	0.037	0.038	0.042	0.048
16	0.075	0.058	0.062	0.067	0.063	0.080
17	0.150	0.116	0.125	0.122	0.137	0.121
18	0.462	0.295	0.284	0.426	0.270	0.258
19	2.540	1.196	0.780	0.778	0.852	0.866
20	7.796	4.876	3.280	3.632	3.076	3.853

Tabela 5: Tempos para cálculos de elementos de Fibonacci em Anahy, sem considerar a localidade dos dados no grafo (Arq. Bi-Processada)

	Processadores Virtuais				
Fibo	1	2	3	4	5
15	1.2096	1.2844	0.8864	0.7910	1.0624
16	4.6483	4.9508	3.2793	2.9443	3.8288
17	19.1264	20.7902	12.6955	11.8655	14.7995
18	78.5395	84.5715	53.9182	45.3047	59.2708
19	295.6852	350.8897	271.9952	187.3620	245.2661
20	1386.7225	1501.5152	705.1867	775.3278	842.4757

tarefas em sentido inverso, aumentando a sincronização (e portanto o sobrecusto) entre tarefas em execução.

Conclusão

Os resultados de desempenho obtidos nas execuções de Fibonacci mostraram que o ambiente Anahy conseguiu manipular com bastante sucesso aplicações que demandam um grande número de tarefas, permitindo uma boa exploração dos recursos disponíveis em uma arquitetura SMP. A utilização de diferentes algoritmos para lidar com o grafo de dependência de tarefas permitiu identificar ciclos de busca desnecessários, de forma a otimizar o ambiente para levar em consideração tais fatores. Como trabalhos futuros, a versão atual será incrementada de forma a manter compatibilidade com aplicações existentes que seguem o padrão POSIX para *threads*. Atualmente está-se estabilizando ao máximo o ambiente em diferentes arquiteturas e buscando métodos para aumentar ainda seu desempenho em arquiteturas SMP.

Referências

[BLU 95] BLUMOFE, R. D. et al. Cilk: an efficient multithreaded runtime system. **ACM SIGPLAN Notices**, v.30, n.8, p.207–216, Aug. 1995.

[CAV 2001] CAVALHEIRO, G. G. H. Introdução à programação paralela e distribuída. In: ERAD 2001, 2001, Gramado. **Anais...** [S.l.: s.n.], 2001.

[GAL 98] GALILÉE, F. et al. Athapascan-1: on-line building data flow graph in a parallel language. In: PACT'98, 1998, Paris, France. **Anais...** [S.l.: s.n.], 1998.

[LOW 99] LOWENTHAL, D. K.; FREEH, V. W.; ANDREWS, G. R. Using fine-grain threads and run-time decision making in parallel computing. **Journal of Parallel and Distributed Computing, Special Issue on Multithreading for Multiprocessors**, 1999. À apparaître.

[REA 2002] REAL, L. C. V.; DALL'AGNOL, E. C.; CAVALHEIRO, G. G. H. Construção de um ambiente de programação para o processamento de alto desempenho. In: SESSãO DE PôSTERES, ERAD 2002, 2002, São Leopoldo, Brasil. **Anais...** [S.l.: s.n.], 2002.