XVII CIC
X ENPOS
Conhecimento sem fronteiras
XVII Congresso de Iniciação Científica
X Encontro de Pós-Graduação
11, 12, 13 e 14 de novembro de 2008

ATHREADS - UM AMBIENTE PARA PROGRAMAÇÃO E EXECUÇÃO PARALELO

VIÇOSA, Elvio Junior[1]; CAVALHEIRO, Gerson Geraldo Homrich[1];

[1]Departamento de Informática – IFM/UFPel

Campus Universitário {evicosa_ifm , gerson.cavalheiro}@ufpel.edu.br

1. INTRODUÇÃO

Durante muitos anos o aumento de desempenho dos computadores foi obtido pela constante diminuição dos transistores que compõe os chips dos seus núcleos de processamento. Esta diminuição possibilitou que mais transistores fizessem parte do chip e desta forma mais operações pudessem ser processadas. A tarefa de desenvolver transistores cada vez menores foi imensamente dificultada devido aos limites físicos encontrados no processo de fabricação, sendo que poucos avanços foram conseguido nos últimos anos e com custos de fabricação proibitivos. A necessidade pelo aumento de desempenho dos processadores motivou a indústria a popularizar a tecnologia *multicore*. A popularização da tecnologia *multicore* trouxe consigo a difícil tarefa de desenvolver aplicações que possam ao mesmo tempo utilizar todos os recursos disponíveis de forma eficiente (KWOK e AHMAD, 1999) e conseguir manter a escalabilidade da aplicação entre diferentes configurações das arquiteturas disponíveis. Este artigo apresenta o protótipo Athreads[1], uma implementação do modelo Anahy Vanilla. Esta ferramenta foi desenhada de forma a obter uma abstração na programação *multithread*, explorando a localidade de dados disponíveis entre as threads em execução e mapeamento da concorrência da aplicação ao paralelismo do hardware, permitindo que a descrição da concorrência da aplicação possa ser feita tomando somente as características da própria aplicação. A Seção 2 apresenta o protótipo Athreads detalhando as características de funcionamento e interface para o uso de serviços. A Seção 3 apresenta os resultados que foram obtidos durante a análise de execução. Este artigo é concluído na Seção 4, detalhando também as funcionalidades que estão sendo desenvolvidas e farão parte das futuras versões de Athreads.

[1]Disponível em http://www.anahy.org

2. MATERIAL E MÉTODOS

O modelo Anahy (CORDEIRO, 2002) foi concebido com o objetivo de fornecer suporte ao desenvolvimento de interfaces de programação para arquiteturas multiprocessadas. Anahy Vanilla é um subconjunto do modelo Anahy, desenvolvido especificamente para programação leve (*multithreading*) e arquiteturas multiprocessadas (com acesso uniforme e não-uniforme à memória). Athreads é uma implementação do modelo Anahy Vanilla, seguindo o padrão POSIX para threads. Utilizando operações do tipo *split/join*, é permitido descrever a concorrência no modelo Anahy Vanilla. Uma operação de *split* implica na criação de uma nova thread. Uma thread criada é sincronizada quando invocada uma operação join sobre ela. O gerenciamento de criação e sincronização é feito pelo núcleo administrativo de Athreads, que mantém todas as threads criadas (e ainda não sincronizadas) em um grafo organizado de forma a descrever o fluxo do programa pela relação de dependência entre as threads. Diferentemente de ferramentas tradicionais de programação paralela, quando uma thread é criada em Athreads ela não cria imediatamente um fluxo de execução concorrente. Durante sua criação, a thread passa a fazer parte do grafo e cabe ao núcleo administrativo de Athreads mapear esta thread a um processador virtual disponível. Os processadores virtuais são threads do sistema que são definidos pelo usuário durante o início da execução da aplicação, sendo eles responsáveis por executar as threads criadas no ambiente Athreads, que estão armazenadas no grafo. Desta forma, o número de fluxos concorrentes é limitado diretamente pelo número de processadores virtuais criados durante o início da execução, não importando quantas threads foram criadas por meio da interface de Athreads. A Figura 1 detalha o ambiente Athreads através das suas camadas. Estas camadas demonstram a ordem de comunicação que é feita pela interface de programação (camada superior) e o efeito colateral desta comunicação com o hardware paralelo (camada inferior), sendo as camadas intermediárias responsáveis pelo gerênciamento de toda comunicação e controle do ambiente.

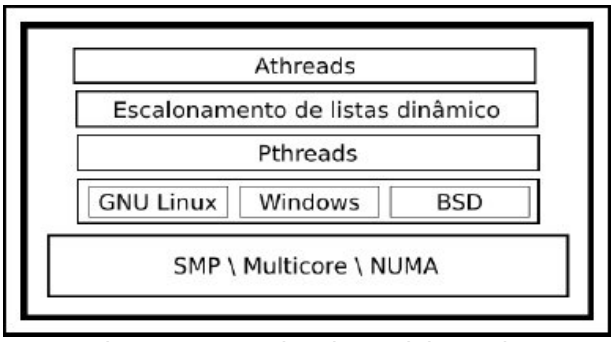

Figura 1: Camadas do modelo Anahy

A implementação de Anahy Vanilla em Athreads segue o padrão POSIX, oferecendo uma interface de serviços semelhante a encontrada em Pthreads. As operações *split/join* possuem a seguinte interface:

```
athread_create(athread_t *th, athread_at r_t *attr, pfunc func, void *params)
athread_join(athread_t th, void *result)
```

A função athread_create permite que uma nova thread seja criada. O

parâmetro func define o corpo da thread, definido como um ponteiro para uma função do tipo void. A thread receberá como parâmetros de entrada o conteúdo apontado pelo ponteiro definido no parâmetro params. Cada thread possui um identificador único durante a execução, que tem seu valor atualizado no parâmetro th. O parâmtro attr permite que o programador configure diferentes características da thread. A função athread_join permite sincronizar uma thread identificada pelo parâmetro th. O retorno da thread é disponibilizado em memória compartilhada, e o parâmetro result recebe o endereço da área de memória alocada.

Para medir o desempenho do ambiente Athreads, foi desenvolvida uma aplicação denominada miner. Miner é uma aplicação recursiva que possui o grafo que descreve seu fluxo de dados, desbalanceado do lado esquerdo. Cada mineiro deve computar uma determinada carga, caso a carga possua um tamanho grande o mineiro pode pedir ajuda a outros dois mineiros, criados recursivamente. Se a carga for pequena, então é retornada a computação em si, terminando a recursão e o resultado é obtido pela combinação das computações de todos os mineiros criados. Para este trabalho foram criadas três variações da aplicação miner, que diferem pela estratégia de sincronização. As variações foram denominadas *leftmost, depthmost* e *rightmost*, indicando a ordem de sincronização respectivamente: mais à esquerda, profundidade e mais à direita.

3. RESULTADOS E DISCUSSÃO

A aplicação miner foi executada em uma arquitetura que possui 16 processadores e 8 GB de memória RAM (com acesso não-uniforme) e sistema operacional GNU\Linux.

A Figura 2 mostra o desempenho obtido pelas três diferentes implementações da aplicação miner. Pode-se notar claramente que a utilização do ambiente Athreads permitiu uma considerável queda no tempo de execução das aplicações, sendo a variação *leftmost* a que possuiu menor tempo de execução.

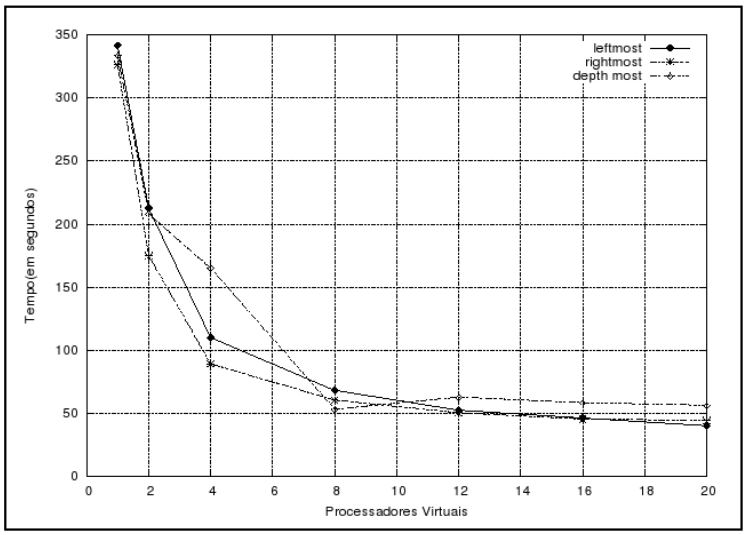

Figura 2: Miner - Desempenho das variações desenvolvidas

A Figura 3 mostra o gráfico de *speedup*. Ele mostra básicamente a diferença do melhor tempo obtido pela execução seqüencial da aplicação (sem utilização de

threads), contra os tempos obtidos com a variação no número de processadores virtuais do ambiente Athreads. Pode-se verificar fácilmente que a diminuição nos tempos de execução estão ligadas ao aumento no número de processadores virtuais do ambiente Athreads.

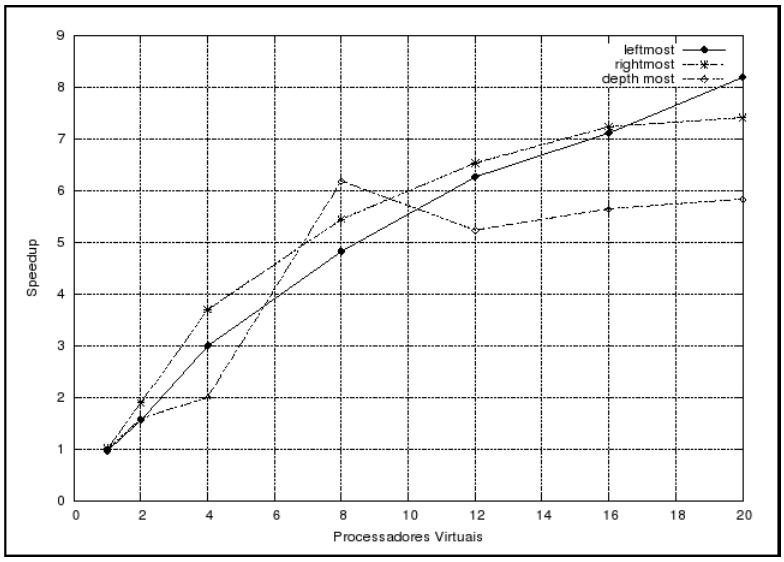

Figura 3: Miner - Speedup

4. CONCLUSÕES

O protótipo Athreads apresentou um bom desempenho no gerêncimento dos recursos disponíveis por diferentes arquiteturas onde foi testado (arquiteturas com 1, 2, 8 e 16 processadores), demonstrando uma capacidade de manter a portabilidade de desempenho das aplicações entre arquiteturas com configurações de recursos diferentes, sem a necessidade de modificações da aplicação.

Novas funcionalidades estão sendo incorporadas ao núcelo de Athreads, que permitem descrever a concorrência da aplicação com mais abstração do que é permitido hoje, e também uma melhora no gerenciamento na utilização de memória pelo ambiente de execução. Uma versão denominada Anahy Vanilla Sugar está sendo implementada, tendo como objetivo a exploração do modelo de computação distribuída entre vários computadores, além da exploração dos recursos de arquiteturas multiprocessadas.

5.REFERÊNCIAS BIBLIOGRÁFICAS

KWOK, Yukwong e AHMAD, Ishfaq. **Static scheduling algorithms for al ocating directed task graphs to multiprocessors**. ACM Comput. Surv., VII High Per formance Computing for Computational Science, Berlin 1999, v. 31, p. 406471.

CORDEIRO, Otávio et al. **Exploiting Multithreaded Programming on Cluster Architectures**, HPCS, 2002.