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

MODIFICAÇÃO DO NÚCLEO DE EXECUÇÃO DA LINGUAGEM PFUN

Lucas Dutra Fonseca[1], Cícero Augusto de Souza Camargo[1], Douglas Eduardo Rosa, André Rauber Du Bois[3], Maurício Lima Pilla[2], Gerson Geraldo Homrich Cavalheiro[2]

1 *Curso de Bacharelado em Ciência da Computação – UFPel*
{ lucas.fonseca, cicero.camargo, douglas.erosa}@anahy.org

2 *Departamento de Informática – IFM/UFPel*
mauricio.pilla@gmail.com, gerson.cavalheiro@ufpel.edu.br

3 *Centro Politécnico - UCPel*
dubois@ucpel.tche.br

1.INTRODUÇÃO

Uma das grandes questões em aberto no desenvolvimento de arquiteturas para processamento paralelo consiste na dificuldade de especificar ferramentas para programação que expressem a concorrência das aplicações de forma adequada.

A grande dificuldade encontrada pelo programador está relacionada com a descrição das aplicações em termos de atividades paralelas e execução do programa de forma a utilizar eficientemente os recursos de processamento disponíveis.

O paradigma de programação funcional tem como característica permitir a construção de programas através de funções matemáticas, fato que torna natural a identificação das computações que podem ser realizadas de forma concorrente. O controle das comunicações de resultados entre as funções se dá pelo controle do fluxo de dados.

O modelo de execução proposto por Cavalheiro (2006) no projeto Anahy prevê um mecanismo de execução eficiente baseado em algoritmo de lista (Graham, 1969). A implementação deste ambiente em Athreads (Cordeiro, 2005) comprovou que programas paralelos cuja estrutura de execução possa ser descrita em termos de fluxo de dados podem ser executados de forma eficiente em arquiteturas multiprocessadas.

Este trabalho apresenta o desenvolvimento de uma ferramenta de programação para arquiteturas *multicore* utilizando uma interface de programação

baseada no paradigma funcional e tendo como suporte de execução um núcleo de escalonamento baseado em algoritmo de lista. O resultado a ser atingido é a apresentação de uma nova versão do ambiente de execução da linguagem pFun, inicialmente proposto como uma extensão da linguagem Haskell a ser executada em arquiteturas com memória distribuída (Du Bois, 2007).

2.MATERIAL E MÉTODOS

O ambiente de programação utilizado é embasado no paradigma de programação funcional, o qual enfatiza a avaliação de expressões matemáticas que realizam o mapeamento de valores de entrada em valores de saída. O paradigma funcional apresenta particularidades interessantes como a possibilidade de avaliação de expressões independente da ordem em que aparecem, o que garante grande potencial de paralelismo na execução, possibilitando um alto grau de abstração na descrição deste paralelismo.

Neste projeto tem-se como base de desenvolvimento as arquiteturas multiprocessadas. Dentre essas encontra-se a arquitetura multi-core que caracteriza-se pela presença de dois ou mais núcleos de execução dentro de um único chip que compartilham um único espaço de endereçamento (memória compartilhada) com latência de acesso uniforme (UMA – *Uniform Memory Access*). Esta arquitetura pode ser classificada como uma variação da arquitetura SMP (*Symmetric Multiprocessors*). Outro tipo de arquitetura multiprocessada existente é a NUMA (*Non-Uniform Memory Access*) (Hwang, 1993), onde cada unidade de processamento possui um módulo de memória próprio, mas acessível por todos os processadores do sistema implicando em tempos de latência diferentes para acessos locais e remotos, ou seja, existe uma classificação hierárquica da memória.

Na adaptação do núcleo de execução da linguagem pFun os métodos utilizados foram baseados no algoritmo de lista de Graham (1969) e na estratégia de escalonamento de *Work Stealing* (Blumofe, 1994), visando melhores táticas de escalonamento. O algoritmo de lista consiste em identificar recursivamente todas as tarefas que necessariamente serão executadas de maneira seqüencial devido à dependência de dados entre elas, verificando-se assim o caminho crítico da execução da aplicação. A partir dessa informação busca-se explorar a localidade de dados alocando-se toda esta linha de tarefas ao mesmo processador, a fim de minimizar o custo de comunicação entre elas. Já a estratégia do *work stealing* consiste em um processador ocioso roubar uma tarefa de outro processador que se encontra sobrecarregado. O roubo de trabalho é feito sobre as tarefas de maior granulosidade, ou seja, que representam potencialmente maior carga para, novamente, diminuir custos de comunicação além de resultar em um melhor balanceamento de carga entre os nós de processamento.

Um fator importante a ser observado é a necessidade de garantir sincronização no acesso às estruturas compartilhadas, de modo a tornar a execução *threadsafe* (Plauger, 1998), ou seja, tornar possível a execução de múltiplas threads simultaneamente sempre mantendo um estado válido de execução. Para tanto, foram utilizados mecanismos de sincronização como variáveis de condição e áreas de exclusão mútua, recursos providos pela biblioteca que dá suporte ao uso de *threads* POSIX. Foi utilizada a linguagem C padrão, por questões de projeto.

3. RESULTADOS E DISCUSSÃO

O ambiente de execução do pFun foi originalmente definido em termos de um ambiente de memória distribuída, com comunicação por troca de mensagens entre os computadores entre os quais as tarefas eram escalonadas. Neste trabalho, foi realizada a adaptação do ambiente de execução para permitir o uso eficiente de multiprocessadores, empregando memória compartilhada para a comunicação. Para tanto, foi necessário modificar as estruturas internas da máquina virtual de forma que o comportamento da implementação original, onde cada um dos vários nós do aglomerado de computadores rodava uma instância da máquina virtual, pudesse ser emulado neste novo ambiente. A organização estrutural da máquina virtual da versão original pode ser vista na figura 1(a).

Neste novo contexto, tem-se uma única instância da máquina virtual rodando sobre uma arquitetura multiprocessada, onde a estrutura de cliente-servidor é substituída pelo uso de processadores virtuais (PV). Cada PV é implementado como uma *thread* da máquina virtual, podendo tanto gerar como consumir tarefas, numa estratégia *peer-to-peer*. A gerência da *threadpool* passa a ser local a cada PV, enquanto a *taskpool* (conjunto de tarefas que aguardam para serem executadas) permanece compartilhada com o intuito de manter a facilidade de acesso às tarefas geradas para o roubo de trabalho. A organização estrutural da nova máquina virtual pode ser vista na figura 1(b).

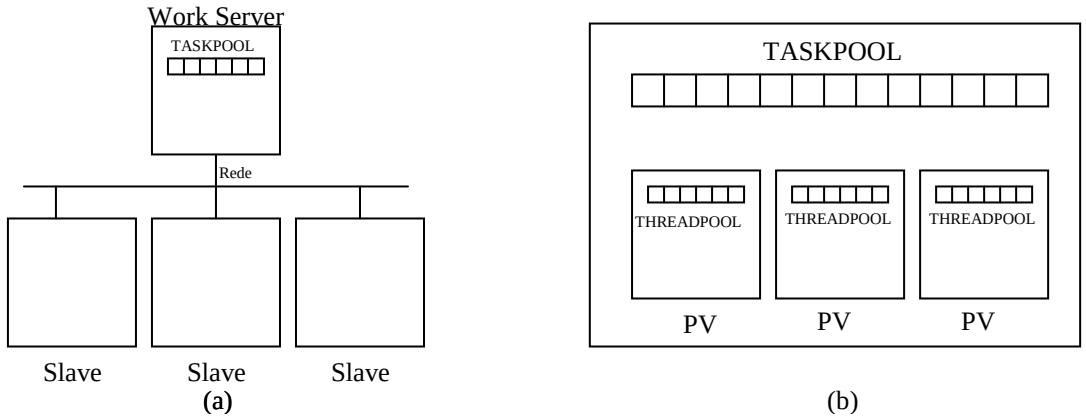

Figura 1 – Comparação entre (a) Implementação original e (b) Nova implementação

A implementação original alcança bom desempenho em aplicações cujas tarefas não possuem interdependência de dados, também chamadas de *bag of tasks*. Neste trabalho espera-se obter um desempenho superior ao alcançado pelo ambiente pFun em memória distribuída, tanto em aplicações que descrevam o modelo de *bag of tasks*, quanto em aplicações em que há dependência entre as tarefas a serem executadas. Este objetivo deve ser alcançado em virtude das novas estratégias de escalonamento adotadas, bem como pela ausência dos custos de comunicação devido ao ambiente de memória compartilhada.

A proposta parcialmente implementada deste trabalho já foi apresentada e discutida em Camargo (2008).

4. CONCLUSÕES

Neste artigo, foi apresentada a evolução do projeto de adaptação do núcleo de

execução da linguagem pFun, uma linguagem funcional pura acrescida de primitivas que permitem descrever paralelismo de execução das aplicações. Foram desenvolvidos novos mecanismos que possibilitaram reproduzir o funcionamento original em um novo ambiente de execução. Buscando obter melhorias no desempenho foi modificada a estrutura da máquina virtual de maneira a usufruir de todo o potencial disponibilizado pelas arquiteturas multiprocessadas. Tais modificações possibilitam a otimização de execução de aplicações que geram carga de trabalho desbalanceada.

Em trabalhos futuros, pretende-se implementar uma versão híbrida do ambiente pFun visando a execução em estruturas computacionais de alto desempenho, como as grades computacionais, buscando explorar o paralelismo inter e intra-nó.

5.REFERÊNCIAS BIBLIOGRÁFICAS

BLUMOFE, R.; Leiserson, C. **Scheduling multithreaded computations by workstealing**. Proceedings of the 35th Annual Symposium on Foundations of ComputerScience, Santa Fe, New Mexico, p. 356–368, November 1994.

CAMARGO, C.; Rosa, D. E.; Fonseca, L. D.; Cavalheiro, G.; Du Bois, A. R. **Adaptação da Linguagem pFun para Ambientes de Memória Compartilhada**. In: Second Workshop on Languages and Tools for Parallel and Distributed Programming. 2008.

CAVALHEIRO, G. G. H. ; Gaspary, L. P. ; Cardozo Júnior, M. A. ; Cordeiro, O. C. **Anahy: A programming Environment for cluster computing**. In: High Performance Computing for Computational Science - VECPAR 2006, 2007, Rio de Janeiro. LNCS High Performance Computing for Computational Science - VecPar 2006. Heildelberg : Springer-Velag, 2006. p. 198-211.

CORDEIRO, O. C. ; Peranconi, D. S. ; Villa Real, L. C. ; Dall Agnol, E. C. ; Cavalheiro, G. G. H. . **Exploiting Multithreaded Programming on Cluster Architectures**. In: HPCS 2005 - International Symposium on High Performance Computing Systems and Applications, 2005, Guelph. 19th International Symposium on High Performance Computing Sustems and Applications. Los Alamitos : IEEE Computer Society, 2005. v. 1. p. 90-96.

DU BOIS, A. R.; Cavalheiro, G.; Yamin, A.; Pilla, M. **pFun: A semi-explicit parallel purely functional language**. In: Electronic Proceedings of the First Workshop On Languages and Tools for Parallel and Distributed Programming. 2007.

GRAHAM, R. L. **Bounds on Multiprocessing Timing Anomalies**. SIAM Journal ofApplied Mathematics, v.17, n.2, p.416-429, 1969.

HWANG, K. **Advanced Computer Architecture**: Parallelism, Scalability, Programability. (s.l.) McGraw-Hill. 1993.

PLAUGER, P. J. **Thread Safety**. C/C++ Users Journal, v.16, n.12, p.10-19, 1998.