ESCALONAMENTO EM AMBIENTE MULTITHREAD DINÂMICO SENSÍVEL À ARQUITETURA E AS DEPENDÊNCIAS DE DADOS ENTRE AS TAREFAS*

RODOLFO MIGON FAVARETTO†[1]; MAURÍCIO LIMA PILLA[1]; GERSON GERALDO HOMRICH CAVALHEIRO[1]

[1]Programa de Pós-Graduação em Computação
Centro de Desenvolvimento Tecnológico - Universidade Federal de Pelotas (UFPel).
E-mails: {rmfavaretto, pilla, gerson.cavalheiro}@inf.ufpel.edu.br.

1. INTRODUÇÃO

As limitações de escalabilidade que os computadores SMPs (*Symmetric Multi-Processor*) possuem em função do problema de disputa entre os processadores por acesso ao barramento, quando do acesso à memória, justificam alternativas como a utilização de arquiteturas NUMA (*Non-Uniform Memory Access*) (Ribeiro, 2011). O interesse no uso destas arquiteturas é ainda maior quando considera-se que estas oferecem uma relação mais atraente entre custo e desempenho quando comparadas às arquiteturas SMPs (ANNAVARAM; SHEN, 2005, KUMAR et al., 2004).

Porém, para que se consiga obter bons índices de desempenho em seus programas, é desejável que o programador conheça os detalhes da arquitetura disponível e inclua no código da aplicação instruções especificas para o mapeamento de suas atividades sobre os recursos de processamento disponíveis. As arquiteturas NUMA têm como característica diferentes tempos de acesso à memória, para diferentes combinações de núcleos de processamento e endereços de memória, uma vez que o espaço de endereçamento encontra-se divido em diferentes módulos de memória e que cada um destes módulos encontra-se fisicamente próximo a um determinado subconjunto de processadores.

Neste contexto, a eficiência pode ser obtida pela gestão, em nível aplicativo, destes acessos via escalonamento, considerando as entradas e saídas especificadas pelas diferentes atividades paralelas (tarefas) criadas pelo programa em execução. Esta questão é tratada no presente trabalho.

O escalonador deve tomar decisões que são influenciadas por diversos fatores. Um destes fatores diz respeito a questões de localidade dos dados, que pode ser levada em conta pela observação dos endereços de memória dos dados acessados, como entrada ou saída, das tarefas e das dependências de dados entre elas. A decisão de alocar uma determinada tarefa sobre um núcleo de processamento específico passa pela análise dos custos associados ao acesso aos seus dados de entrada e saída na estrutura assimétrica de memória.

Outro fator a ser considerado está relacionado às dependências de dados utilizados pelas tarefas. Diferentes heurísticas baseadas em algoritmos de lista podem ser utilizadas para realizar o escalonamento de tarefas em nível aplicativo

* FAPERGS/PqG (11/1065-1), PRONEX/FAPERGS/CNPq (10/0042-8).
† Bolsista de Mestrado FAPERGS.

em ambientes de execução dinâmicos, levando em consideração essas dependências de dados entre as tarefas durante a execução da aplicação.

Neste trabalho, a proposta é trabalhar com uma visão compartilhada destas duas abordagens em dois níveis de escalonamento. No primeiro, será considerada a localidade física dos dados e as dependências destes dados entre as tarefas para que se consiga escaloná-las entre os processadores de maneira mais eficiente. No segundo nível será considerada a topologia da máquina onde a aplicação está sendo executada e suas características relacionadas às diferentes assimetrias de acesso.

A Figura 1 ilustra essa ideia de escalonamento, onde no primeiro nível as tarefas produzidas por um programa paralelo precisam ser distribuídas entre Processadores Virtuais (PVs), considerando as dependências de dados entre elas e, no segundo, esses PVs precisam ser escalonados entre os processadores físicos presentes na arquitetura da máquina, considerando as diferentes assimetrias de latências no acesso aos dados produzidos pelas tarefas.

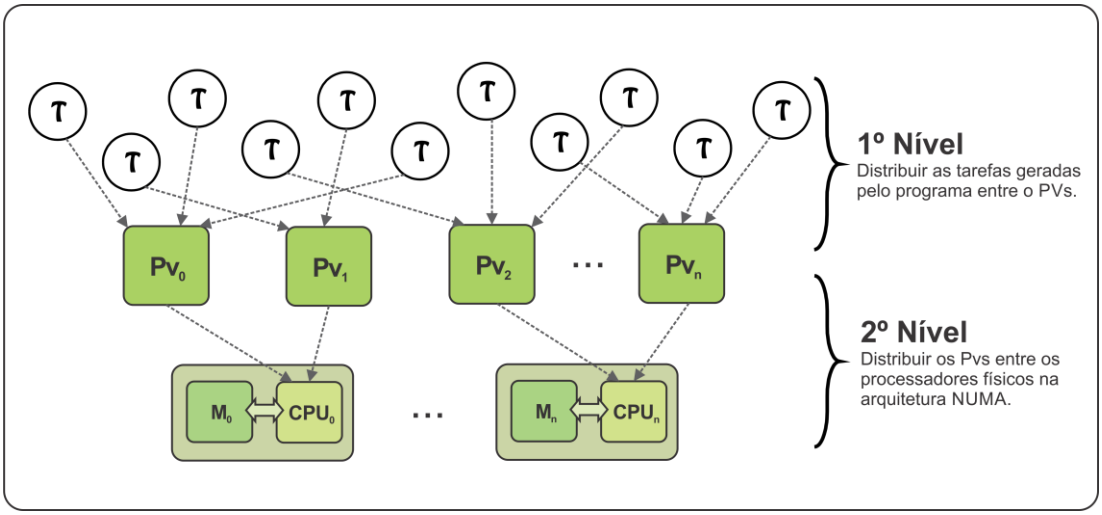

Figura 1 – Ideia Geral do Escalonamento

O objetivo principal do trabalho, então, consiste na modelagem de uma estratégia de escalonamento que contemple esses dois níveis. Em um segundo momento, essa estratégia será inserida no núcleo de escalonamento do *Anahy* (CAVALHEIRO et al., 2007), um ambiente de programação e execução *multithread* dinâmico projetado para arquiteturas multiprocessadas, onde seu núcleo de escalonamento foi concebido para explorar, em nível aplicativo, informações sobre a estrutura do programa paralelo.

2. METODOLOGIA

O desenvolvimento do trabalho inicia com a revisão dos principais conceitos teóricos de escalonamento de tarefas em nível aplicativo disponíveis na literatura e, também, das recentes pesquisas feitas pela comunidade científica nesse sentido, ou seja, o estado da arte o qual se encontram soluções para o problema abordado.

A metodologia empregada consiste no estudo, especificação, implementação e validação de uma estratégia de escalonamento para

arquiteturas NUMA, considerando suas características e as dependências de dados entre as tarefas.

Inicialmente, será realizada a especificação e parametrização da estratégia bem como das heurísticas que serão utilizadas pelo escalonador para realizar a distribuição e o balanceamento de carga. Uma vez especificada, a estratégia de escalonamento será implementada na forma de um protótipo para avaliação. Para tal, será utilizada a ferramenta CHARM++, pois oferece uma estrutura para o balanceamento de carga com estatísticas da aplicação em tempo de execução.

Para extrair as características da arquitetura a serem consideradas pelo escalonador no momento da decisão, será utilizada a ferramenta *HwLoc* (abreviação para *Hardware Locality*). Trata-se de uma ferramenta para coletar as informações da topologia da máquina e também informações sobre memória e interconexões, tais como latências de acessos e largura de banda para transmissão dos dados.

Após a implementação da estratégia, serão realizados diversos testes para validá-la e, eventualmente, realizar alguns ajustes necessários. Uma vez validada, a estratégia será inserida no núcleo de execução do *Anahy*, a fim de contribuir com o aumento de desempenho da ferramenta.

3. RESULTADOS E DISCUSSÃO

Atualmente, este trabalho encontra-se em fase de desenvolvimento, onde a estratégia está sendo prototipada com *Charm++*. A Figura 2 apresenta o fluxo de execução da estratégia de escalonamento que está sendo desenvolvida. Inicialmente, quando o programa inicia a execução, um arquivo contendo as configurações da topologia da máquina é carregado. Caso esse arquivo não seja encontrado em um diretório específico, o programa considera que a máquina apresenta uma arquitetura uniforme, ou seja, os tempos de acesso a memória são os mesmos para cada processador.

Antes que a estratégia de escalonamento entre em ação, o usuário deverá informar os custos computacionais de cada tarefa, bem como os custos de comunicação dos dados entre uma tarefa e outra. Com base nestas informações (topologia da máquina e custos das tarefas e comunicações entre elas) uma estratégia de escalonamento é aplicada enquanto houverem tarefas não executadas.

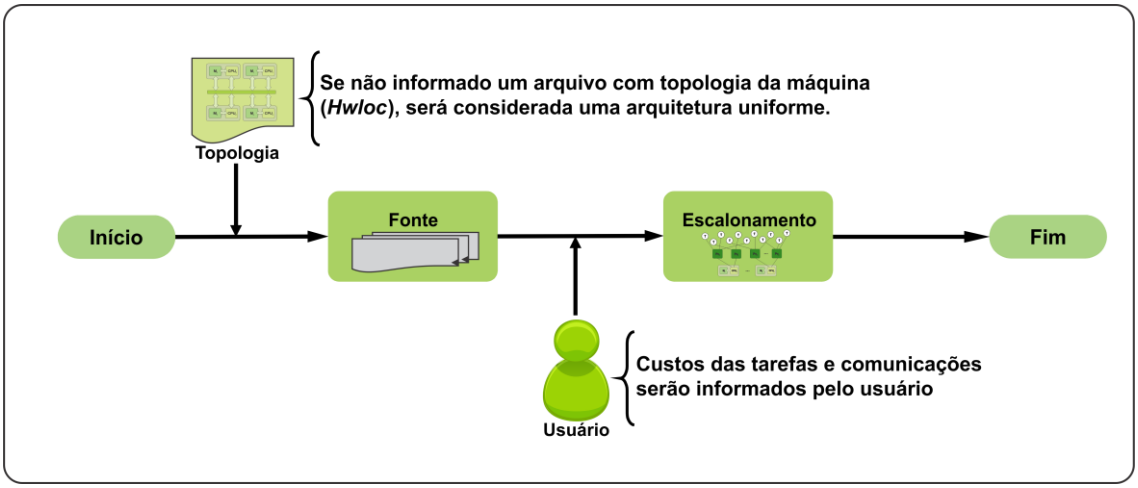

Figura 2 – Fluxo de Execução da Estratégia Proposta

Para investigar o desempenho e a eficiência da estratégia desenvolvida, diversos experimentos serão realizados. Estes experimentos serão compostos por aplicações sintéticas, capazes de explorar as características deste tipo de escalonamento considerando diferentes estruturas de programas paralelos.

4. CONCLUSÕES

Como contribuição principal deste trabalho, espera-se obter uma estratégia de escalonamento para arquiteturas NUMA visando melhorar o desempenho de execução de um programa, considerando as características dessa arquitetura e as dependências de dados entre as tarefas para determinar a alocação destas tarefas em função dos dados por elas manipulados.

Espera-se ainda contribuir no aumento de desempenho do *Anahy*, estendendo este ambiente para que ele ofereça suporte ao escalonamento de tarefas baseado na estratégia desenvolvida.

5. REFERÊNCIAS BIBLIOGRÁFICAS

ANNAVARAM, M.; SHEN, J. Mitigating amdahl's law through epi throttling. In: International Symposium on Computer Architecture. **Anais ISCA**, 2005. p.298 – 309.

CAVALHEIRO, G. G. H.; GASPARY, L. P.; CARDOZO, M. A.; CORDEIRO, O. C. Anahy: A Programming Environment for Cluster Computing. In: VII HIGH PERFORMANCE COMPUTING FOR COMPUTATIONAL SCIENCE, 2007, Berlin. Anais. . . Springer-Verlag, 2007.

KUMAR, R.; TULLSEN, D.; RANGANATHAN, P.; JOUPPI, N.; FARKAS, K. A Single-ISA heterogeneous multicore architecture for multithreaded workload performance. In: Int. Symposium on Computer Architecture. **Anais ISCA**, 2004. p.64 – 75.

RIBEIRO, N. S. **Explorando programação Híbrida no contexto de Clusters de Máquina NUMA**. 2011. (Dissertação), PPGC - PUCRS, Porto Alegre, RS.