XVIII CIC XI ENPOS I MOSTRA CIENTÍFICA
Evoluir sem extinguir: por uma ciência do devir

ESTUDO DA APLICAÇÃO DE ALGORITMOS DE ESCALONAMENTO DE LISTA NO NÚCLEO DE EXECUÇÃO DA LINGUAGEM AIYRA

CAMARGO, Cícero Augusto de S.; CAVALHEIRO, Gerson Geraldo H.

Departamento de Informática – Universidade Federal de Pelotas (UFPEL)
Pelotas – RS – Brasil

cicero.camargo@anahy.org, gerson.cavalheiro@ufpel.edu.br

1. INTRODUÇÃO

Uma das mais importantes áreas do processamento paralelo é aquela que busca exploração de hardware de arquiteturas dotadas de múltiplos processadores para ganho de desempenho na execução de aplicações pelo uso de linguagens e ambientes de execução paralelos. Essa área envolve a utilização de técnicas de escalonamento, as quais aplicam heurísticas para determinar a distribuição no tempo das atividades concorrentes sobre os recursos de processamento disponível.

A literatura cita que algoritmos de escalonamento de lista são os mais eficientes para escalonamento de programas descritos segundo o modelo de grafo de fluxos de dados (GRAHAM, 1969). Estes algoritmos buscam reduzir o tempo de execução de uma aplicação partindo do conhecimento do caminho crítico da aplicação, ou seja, da maior seqüência de dependências de dados entre tarefas. A estratégia básica valora as tarefas do grafo segundo alguma política de prioridade e escalona as tarefas nos processadores de acordo com esta prioridade.

O presente projeto analisa algoritmos de escalonamento de lista com o intuito de identificar estratégias de execução aplicáveis a ambientes *multithread* dinâmicos. O objetivo é desenvolver estratégias de escalonamento de *threads* quando estas reagrupam seqüências de tarefas descritas em termos de fluxos de dados. Os resultados serão aplicados no núcleo de escalonamento de Athreads (CAVALHEIRO, 2007), que dá suporte à execução de programas Aiyra.

2. MÉTODOS E FERRAMENTAS

Programas paralelos podem ser descritos em termos de um DAG (*Directed Acyclic task Graph* – grafo de fluxo de dados) onde cada nó representa uma tarefa e cada aresta entre dois nós representa uma dependência de dados (ou comunicação) entre duas tarefas. Escalonar um DAG consiste em, dado um conjunto finito de processadores, determinar a seqüência de tarefas que cada processador deve executar. Considerando os algoritmos de lista, o escalonamento se dá em dois passos: (1º) as tarefas do grafo são classificadas em uma lista ordenada de acordo com alguma política de prioridade; (2º) as tarefas são retiradas

desta lista obedecendo a prioridade especificada e atribuídas para os processadores na medida em que estes tornam-se disponíveis.

Neste projeto, os algoritmos de lista tomados como objetos de estudo utilizam como atributo de prioridade alguma informação que permita a identificação do caminho crítico. Dois dos mais importantes atributos para conferir

prioridade às tarefas do DAG (AHMAD, 1995) são o t-level (*top level*) e o b-level (*bottom level*). O t-level é um atributo que associa a um determinado nó n_i de um DAG a sua distância até a raiz do grafo – que representa o início do programa. Já o b-level indica a distância de um nó n_i até um nó folha do grafo. Heurísticas aplicadas sobre estes atributos permitem identificar o caminho crítico do grafo.

Embora algoritmos de escalonamento de lista permitam obter execuções eficientes de programas descritos em termos de um DAG, deve-se ter em conta que o máximo de eficiência somente é obtido quando o DAG é inteiramente conhecido antes do início da execução do programa. No caso deste projeto, a aplicação do escalonamento é dinâmica, o que indica que o DAG é gerado em tempo de execução e as heurísticas buscam determinar onde o caminho crítico provavelmente se localiza. Os resultados alcançados serão aplicados no núcleo de execução de Athreads (CAVALHEIRO, 2007).

Athreads implementa sob uma interface de programação *multithreading* o modelo de tarefas concorrentes descritos em um DAG. A Figura 1 representa um DAG gerado pela execução de um programa Athreads.

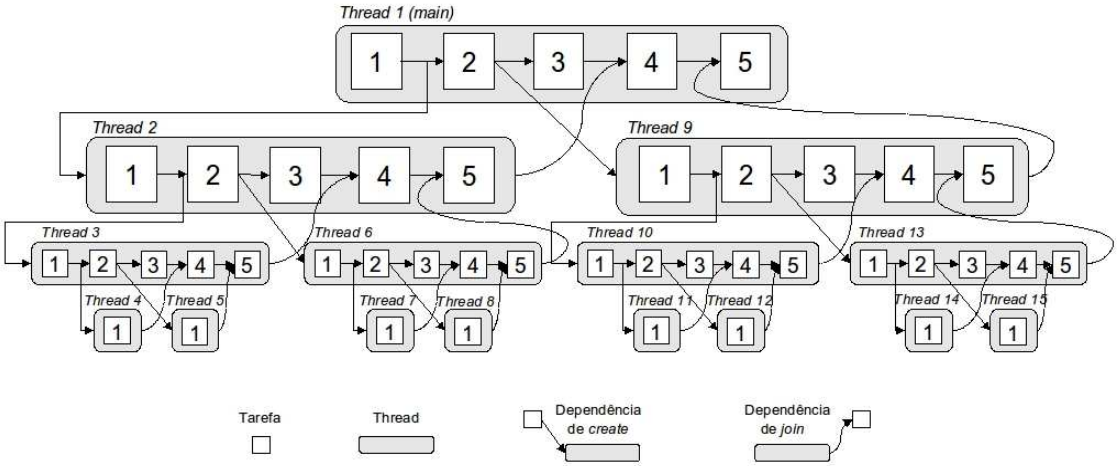

Figura 1. DAG para um programa desenvolvido em Athreads

No DAG da Figura 1 observa-se que *threads* são contêineres de tarefas. Embora algoritmos de lista operem sobre tarefas, a manipulação em termos de *threads* permite reduzir o sobrecusto de manipulação de um grande número de dependências entre tarefas. Como conseqüência, as estratégias de escalonamentos de lista devem ser adaptadas para obter o melhor proveito desta representação e tirar benefícios da execução *multithreaded* (VALIANT, 1990). O estudo indicou a necessidade da criação de uma ferramenta de apoio ao estudo e análise da aplicação de estratégias de escalonamento de listas em programas Athreads executando sobre multiprocessadores. Esta ferramenta foi desenvolvida usando a linguagem de programação C++ e o ambiente GNU/Linux, bem como o software uDraw(Graph) para suporte à visualização dos grafos gerados pela ferramenta, todos softwares livres.

3. RESULTADOS E DISCUSSÃO

Para facilitar a percepção do efeito dos algoritmos de lista aplicados a DAGs do modelo de Athreads, foi desenvolvido um simulador de estratégias de escalonamento de lista. Esta ferramenta é composta de um gerador de DAGs e um aplicador de estratégias de escalonamento. A geração do grafo é parametrizada, para que se possa definir o comportamento do programa a ser avaliado. São quatro os parâmetros necessários: profundidade (P), comprimento (K) e custos de computação (w) e de comunicação (c). A profundidade indica o número de níveis que o grafo possuirá em termos de *threads*. O comprimento indica quantas novas *threads* serão criadas em cada *thread* e os custos de computação e comunicação indicam, respectivamente, os tempos necessário para processamento de cada tarefa e para comunicação de seus resultados.

Após a geração do grafo, o simulador valora cada tarefa do DAG em termos dos respectivos t-level e b-level. Em seguida são identificados, para cada *thread*, os valores mínimos e máximos de t-level e b-level obtidos nas tarefas que estes contêm. O passo seguinte consiste em aplicar uma estratégia de escalonamento de lista sobre o DAG e, por fim, indicar, considerando os t-level e b-level mínimos e máximos de cada thread, o escalonamento desejado para as *threads*.

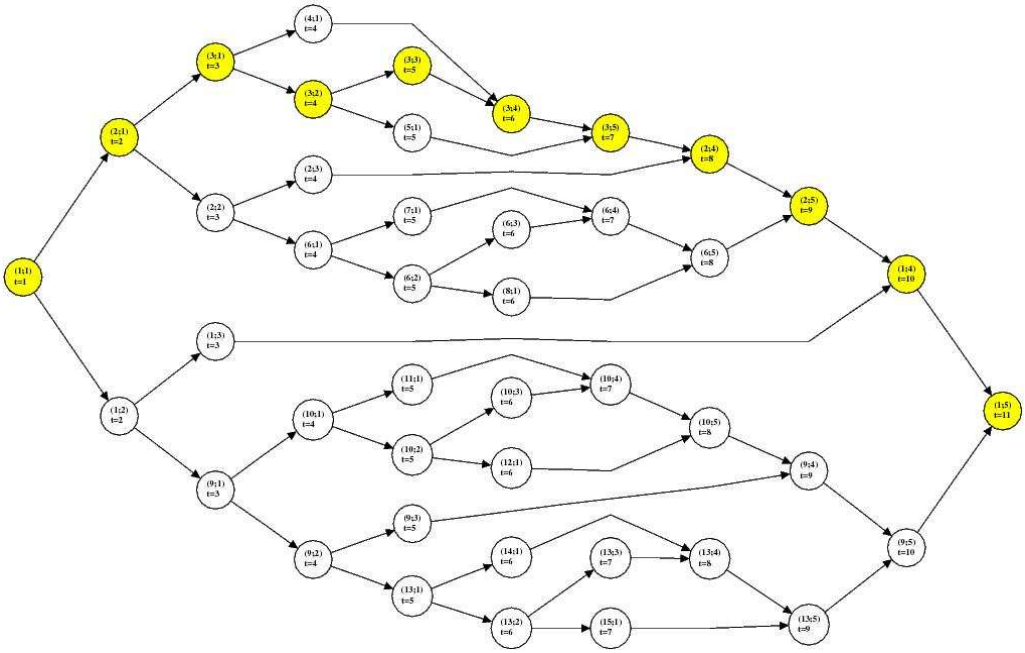

Figura 2. Exibição de um grafo de tarefas usando uDraw(Graph)

Foi desenvolvida uma interface do simulador com o software uDraw(Graph) (2009) para que se possa obter uma representação gráfica dos grafos de tarefas obtidos a partir dos DAGs (no modelo de Athreads) gerados. Assim, o simulador usa o software de desenho de grafos indicando como devem ser desenhados os nodos e as arestas que os ligam, bem como indica quais nodos representam tarefas que estão no caminho crítico e devem ser destacados. A Figura 2 mostra a exibição obtida a partir do mesmo DAG da Figura 1, simulado na ferramenta, observa-se nesta exibição que as tarefas do caminho crítico encontram-se destacadas. Os parâmetros utilizados foram: P = 3, K = 2, w = 1 e c = 0

A Tabela 1 apresenta o escalonamento obtido para o grafo da Figura 2 utilizando o algoritmo DLS (*Dynamic Level Scheduling*) (SIH, 1993) tanto em nível de tarefa como em nível de *thread*. Em cada instante de tempo, um processador

virtual (PV) executa um par “(thread;tarefa)” ou fica ocioso, se não há mais tarefas prontas no momento.

Tabela 1. Escalonamento do grafo da Figura 2 com quatro processadores virtuais

PV	Instante de Tempo													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	(1;1)	(1;2)	(1;3)	(4;1)	(6;1)	(6;2)	(6;3)	(6;4)	(2;4)	(14;1)	(2;5)	(9;4)	(9;5)	(1;5)
1	-	(2;1)	(2;2)	(2;3)	(10;1)	(10;2)	(10;3)	(10;4)	(12;1)	(15;1)	(10;5)	(1;4)	-	-
2	-	-	(3;1)	(3;2)	(3;3)	(3;4)	(5;1)	(3;5)	(8;1)	(6;5)	-	-	-	-
2	-	-	(9;1)	(9;2)	(9;3)	(11;1)	(7;1)	(13;1)	(13;2)	(13;3)	(13;4)	(13;5)	-	-

A Tabela 1 mostra que é possível obter um escalonamento ótimo do ponto de vista de tarefas e *threads* ao mesmo tempo. No escalonamento obtido observa-se que, a cada instante de tempo, os processadores virtuais buscam uma tarefa que possibilite dar segmento ao *thread* em execução, porém sem descuidar da execução das tarefas do caminho crítico

4. CONCLUSÃO

Encontra-se em curso de desenvolvimento a ferramenta Aiyra para programação paralela. O objetivo desta ferramenta é prover recursos de programação de alto desempenho em arquiteturas multiprocessadas, tais processadores *multicore*. Com Aiyra é possível descrever programas paralelos em termos de grafos de dependências de dados entre tarefas, os quais podem ser escalonados de forma eficiente por meio de estratégias de escalonamento de listas quando este grafo for conhecido de forma estática.

Considerando que Aiyra é voltada para implementação de aplicações cujo grafo é construído em tempo de execução e que os programas são descritos em termos de *threads* e não de tarefas, é necessário avaliar o impacto do uso de estratégias de escalonamento de listas na execução deste perfil de aplicação. Este artigo apresentou uma ferramenta de avaliação de estratégias de escalonamento de lista. Esta ferramenta conta com um gerador de DAGs parametrizável, um núcleo de aplicação de estratégias de escalonamento e oferece resultados de forma gráfica facilitando o estudo e a compreensão dos comportamentos observados.

O atual estágio de desenvolvimento contempla a introdução de estratégias de escalonamento de lista no simulador. O trabalho prosseguirá com o estudo dos mapeamentos possíveis dos escalonamentos obtidos com estas técnicas para o nível de *threads* do grafo e, então, na proposição de estratégias dinâmicas de escalonamento de listas para *threads* e da incorporação destas no núcleo de execução de Athreads.

5. REFERÊNCIAS BIBLIOGRÁFICAS

AHMAD, I.; KWOK, Y.; WU M. Performance Comparison of Algorithms for Static Scheduling of DAGs to Multiprocessors, In: 2nd AUSTRALIAN CONFERENCE ON PARALLEL AND REAL-TIME SYSTEMS. **Proceedings of...** Set. 1995, p. 185-192.

CAVALHEIRO, G. G. H.; GASPARY, L. P.; CARDOZO, M. A.; CORDEIRO, O. C. Anahy: A programming environment for cluster computing, In: VECPAR 2006 (LNCS 4395).

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

SIH, G. C.; LEE, E. A. A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures, **IEEE Trans. On Parallel and Distributed Systems**, vol. 4, n.2, p.75-87, 1993.

UDRAW(GRAPH). Disponível em: http://www.informatik.uni-bremen.de/uDrawGraph/en/index.html Acesso em: 19 ago. 2009.

VALIANT, L.G. A bridging model for parallel computation, **Communications of ACM**, vol. 33, n.8, p.103-111, 1990.