Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 1, 2015.
Trabalho apresentado no XXXV CNMAC, Natal-RN, 2014.

Casos de Estudo com Ferramentas de Programação Multithread

Felipe Haertel, Lucas Xavier, Rodrigo Bazo, Gerson Cavalheiro, Renata Reiser
Universidade Federal de pelotas - Centro de Desenvolvimento Tecnológico
Caixa Postal 354
96010-900, Campus Porto, Pelotas, RS
E-mail: {flhaertel,lmdsxavier,rbazo,gerson.cavalheiro,reiser}@inf.ufpel.edu.br

RESUMO

Arquiteturas paralelas têm se tornado onipresentes como suporte computacional em função da popularidade da tecnologia multicore. Nestas arquiteturas, as ferramentas de programação mais adequadas são aquelas que implementam modelos de programação multithread, permitindo que programas sejam descritos por múltiplos fluxos de execução concorrente capazes de ocupar as várias unidades de processamento de forma paralela. Existem dois modelos básicos para decomposição do paralelismo de uma aplicação: a decomposição em termos de paralelismo de dados ou de tarefas. A opção por um ou outro destes modelos se dá em função da natureza do problema que está sendo implementado, sendo que diferentes ferramentas de programação multithread oferecem diferentes facilidades de desenvolvimento para cada caso.

Este trabalho avalia a interface de programação oferecida por quatro ferramentas multithread como suporte à implementação de diferentes padrões de paralelismo expresso por problemas recorrentes em aplicações matemáticas. Neste texto são considerados quatro padrões de problemas: (i) trivialmente paralelizáveis, (ii) recursivos, (iii) programação dinâmica, e (iv) com grande número de dependências entre as atividades paralelas. Estes padrões são representados, respectivamente, pelas seguintes aplicações: (i) problema de Satisfatibilidade Booleana (denominado 3Sat no restante deste texto), (ii) cálculo da série de Fibonacci (Fibo), (iii) o algoritmo Smith-Waterman para o alinhamento de sequências (S-W), e (iv) decomposição LU de uma matriz (LU). O estudo é completado com uma análise do desempenho dos programas construídos para estas aplicações nas diferentes ferramentas considerando suas estratégias de escalonamento. Com este tipo de trabalho, interdisciplinar, entende-se obter avanços tanto na área de desenvolvimento de software matemático como aprimoramento nas técnicas de execução paralela, conforme discutido em [1]

Este texto limita-se a caracterizar as ferramentas de programação selecionadas nesta fase do trabalho, que são a acadêmica Athreads [2], e as comerciais Cilk, OpenMP e TBB, sumarizadas em [4], e exemplificar os resultados pela análise de um conjunto de resultados de desempenho. Um dos objetivos a ser atingido é identificar a usabilidade de ferramentas de programação multithread [3] no desenvolvimento de diferentes padrões de problemas matemáticos.

Athreads, Cilk e TBB embutem um núcleo de escalonamento que explora informações sobre a ordem lexicográfica definida entre os *threads* pelo programa em execução. Destas dependências são inferidos os custos computacionais de cada *thread*, assumindo que quanto mais antigo um *thread* for, maior será seu custo computacional. Esta informação de custo é considerada nas tomadas de decisão sobre os recursos de processamento disponíveis. A diferença entre estas ferramentas se dá na interface de programação. Enquanto Cilk restringe o modelo de programação em uma estrutura de criação e sincronização aninhada de *threads* (nested fork/join), Athreads permite a construção de qualquer estrutura de dependências. Como consequência, a estrutura regular em Cilk pode ser explorada com uma eficiência maior por uma heurística simples de escalonamento que considere esta regularidade nas dependências entre *threads*. Por outro lado, em Athreads cabe ao programador conhecer a estratégia de escalonamento e organizar a criação de *threads* de forma a refletir a heurística de escalonamento empregada. TBB também tem como padrão a criação aninhada de *threads*, mas, para não limitar tanto como Cilk o poder de expressão de paralelismo, esta ferramenta permite ao programador criar *threads* externas a esta estrutura aninhada.

DOI: 10.5540/03.2015.003.01.0114

Com uma interface de programação totalmente diversa das anteriores, OpenMP é voltado para problemas onde ocorre paralelismo de dados, sendo seus programas caracterizados pela criação, também aninhada, de lotes de trabalho. Seu núcleo de escalonamento considera a distribuição da carga de trabalho contida neste lote. Operando, portanto, em rodadas de lotes de trabalhos recebidos. Dependências entre os trabalhos não são consideradas, apenas o programador pode indicar que diferentes trabalhos, em um mesmo lote, possuem cargas computacionais diferentes para auxiliar nas heurísticas de escalonamento.

Na Figura 1 são apresentados os desempenhos, em termos de speedup (relação entre o tempo de execução do problema com uma *thread* e seu tempo de execução paralelo) em uma arquitetura com oito cores. Por questões de espaço, o gráfico reflete o speedup apenas para o caso de execução paralela com também oito *threads*.

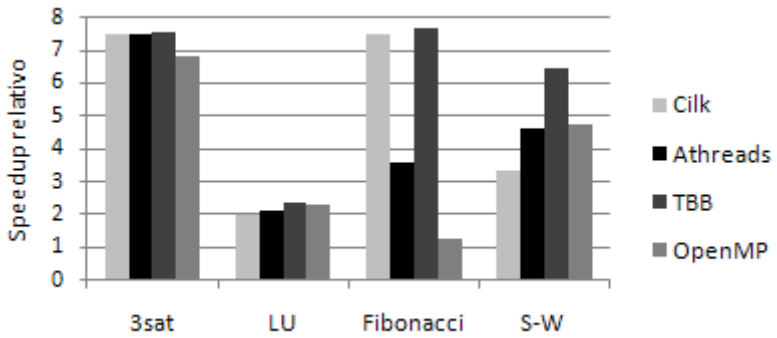

Figura 1. Gráfico de speedup para o caso de oito *threads*.

Observamos que, para uma aplicação trivialmente paralelizável, como o 3Sat, não houve grande impacto do mecanismo de escalonamento e também as interfaces de programação oferecidas pelas ferramentas não impuseram nenhuma restrição a sua implementação. No outro extremo, em aplicações como LU, em que o número de dependências entre *threads* é alto, também observamos que o núcleo de escalonamento não possui margem de trabalho que permita diferenciar as ferramentas. Neste tipo de problema também notou-se uma maior dificuldade de desenvolvimento do programa em ferramentas vocacionadas ao paralelismo de dados, como OpenMP. Já em aplicações onde existe uma estrutura regular na criação de tarefas, como no cálculo recursivo de Fibonacci, a heurística de escalonamento adotada por Athreads, Cilk e TBB reflete em ganho de desempenho. Note-se que, quanto maior for a restrição imposta pela ferramenta para expressão do paralelismo (como em Cilk e TBB), maiores as chances de sucesso do escalonamento. Por outro lado, em Athreads, pequenas variações na ordem de criação de *threads*, o que é permitido pela ferramenta, ocasionam impacto no desempenho obtido. Finalmente, no caso de um algoritmo de programação dinâmica, como o S-W, encontramos dificuldades em modelar o problema considerando as diferentes interfaces de programação utilizadas. No gráfico é traduzido o desempenho obtido na melhor implementação realizada até o momento pelo grupo.

Palavras-chave: *Computação aplicada, padrões de paralelismo, multithread*

Referências

[1] Buttari, A. et all. “The Impact of Multicore on Math Software”. In: Applied Parallel Computing: State of the Art in Scientific Computing. LNCS 4699. Berlim:Springer. 2007.

[2] Cavalheiro G. H., Gaspary, L. P., Cardozo, M. A., Cordeiro, O. C. “Anahy: A Programming Environment for Cluster Computing”. In: VECPAR 2006: 198-211

[3] Marowka, A. “A Study of the Usability of Multicore Threading Tools”. Int. J. of Software Engineering and Its Applications Vol. 4, No. 3, 2010.

[4] Olivier, S. L., Prins, J. F. “Comparison of OpenMP 3.0 and other task Parallel frameworks on unbalanced task graphs”. Int. J. Parallel Prog. Vol. 38. pp 341-360, 2010.

