EFICIÊNCIA ENERGÉTICA ATRAVÉS DO ESCALONAMENTO DINÂMICO DE THREADS E FREQUÊNCIA DE PROCESSAMENTO EM AMBIENTE ANAHY SOBRE ARQUITETURAS MULTI-CORE

ARAUJO, Alan Schlindvein; CAVALHEIRO, Gerson Geraldo Homrich

Departamento de Informática – Dinfo. Universidade Federal de Pelotas – UFPEL

1 INTRODUÇÃO

A crescente presença de recursos computacionais em diferentes áreas da sociedade moderna conflita com a também crescente necessidade de uso racional de recursos energéticos. Em particular, os sistemas computacionais multi-processados consomem ainda mais energia para manter em operação, em sua capacidade máxima, todos os recursos de processamento disponíveis. Este tipo de configuração foi recentemente popularizado pela disponibilidade de opções de computadores dotados de processadores *multi-core* (Wolf, 2004) (Hill et al., 2008) de baixo custo. A tecnologia do *hardware* deste tipo de processador já foi desenvolvida sob a ótica da necessidade de redução de consumo energético, no entanto, a camada de software ainda não contempla satisfatoriamente os requisitos de redução de consumo de energia.

Neste trabalho é avaliada uma estratégia de escalonamento de programas *multithread* em arquiteturas *multi-core*. O objetivo desta estratégia é garantir um uso consciente da energia necessária para execução de programas sem causar impacto significativo no seu desempenho final, refletido em termos de tempo total de execução. As técnicas utilizadas exploram afinidade de fluxos de execução (*threads*) a *cores* e controle de frequência de operação dos *cores*. O estudo foi realizado sobre plataforma Anahy (Cavalheiro et al., 2007).

2 GERÊNCIA EM SOFTWARE DO CONSUMO DE ENERGIA

Os novos processadores *multi-core* disponibilizam, em sua arquitetura, diversas facilidades para controlar o modo de operação de cada *core*, de forma individual (Araujo et al., 2009). O recurso de determinação de afinidade permite que seja especificado para um determinado *thread* qual *core* deve ser responsável pela sua execução. O recurso de controle de frequência de operação permite adaptar a velocidade com que as funções de cada *core* são executadas. O uso combinado destes recursos permite a realização decisões de escalonamento que reduzam o consumo de energia necessário para executar o programa. No entanto, para que estes recursos sejam utilizados de forma eficiente, é necessário que exista conhecimento sobre a estrutura do programa em execução, para que decisões sobre o uso sejam feitas considerando a evolução do programa.

A associação de afinidade de *threads* aos *cores* permite que a frequência de operação dos *cores* seja estabelecida individualmente em função das atividades que a eles forem atribuídas, evitando com que *cores* processem a altas velocidades, consumindo, portanto, mais energia, desnecessariamente. Neste escalonamento, a frequência de processamento, de cada núcleo de execução, pode ser ajustada conforme a necessidade de cada subconjunto de

threads, permitindo o uso consciente de energia com ônus do controle da perda de desempenho.

Faz-se notar que, quanto maior for a frequência de operação da unidade de processamento, maior será o consumo de energia. Portanto, adequar as frequências necessárias aos *cores*, baseadas na carga de trabalho de cada fluxo de execução, permite economizar o consumo de energia, reduzindo a dissipação de calor da unidade de processamento (Uhrig, 2005).

A relação existente entre o consumo de energia e frequência de *clock* pode ser obtido pela Equação 1 (Wechsler et al., 2006), onde C_{DD} representa a capacitância comutada, V_{DD} denota a voltagem do processador, ambas dependentes da entrada do processador, e f expressa a frequência de operação de cada *core*.

$$Consumo = C_{DD} \times V_{DD}^2 \times f \quad (1)$$

A Equação 1 reflete a dependência da tensão, em Volts, à frequência de *clock*. É possível, portanto, reduzir o consumo de energia do sistema com a redução da frequência de processamento de *cores* individuais. Observa-se em consequência uma perda de desempenho global, pois os *cores* que compõem o sistema operariam em velocidades de *clock* reduzidas, podendo então limitar o desempenho de execução de um programa (Uhrig, 2005).

3 EXTENSÃO AO AMBIENTE ANAHY

O ambiente de execução Anahy oferece uma interface para programação *multithread*. Esta interface fornece duas primitivas para manipulação de *threads* e comunicação de resultados: ***create*** e ***join***. Como resultado, o programa é descrito em termos de Grafos Dirigidos com Ciclos (DCGs) de *threads*. Uma discussão mais ampla desta interface foge ao escopo deste artigo, mas esta pode ser encontrada em (Cavalheiro et al., 2007).

O escalonamento implementado em Anahy é responsável pela execução dos *threads* criados pelo programa. Este escalonamento é realizado em nível aplicativo e considera que os *threads* devem ser executados sobre um conjunto de *p* processadores virtuais (PVs). A heurística implementada pelo núcleo de escalonamento de Anahy privilegia a execução dos *threads* que expressam o caminho crítico do programa, este caminho concentra a maior carga computacional gerada. Os PVs, por sua vez, são escalonados pelo sistema operacional nativo da máquina hospedeira sobre os *m cores* disponíveis, tal que $p \geq m$. Em nível aplicativo, um *thread* uma vez associado a um PV não sofre migração, sendo assim possível estimar a carga computacional associada a cada processador virtual. De posse desta informação, é possível também estabelecer a afinidade dos processadores virtuais aos *cores* em função das cargas individuais de cada *thread* e adequar a velocidade de processamento destes de acordo com a necessidade. A extensão realizada sobre o núcleo de execução de Anahy prevê a realização da distribuição de carga computacional dos *threads* do programa e do consumo de energia por meio da adequação da frequência de operação dos *cores* em função da demanda computacional exigida.

4 ESTUDO DE CASO

O estudo de caso apresentado realiza uma avaliação do mecanismo introduzido para gerir a utilização dos *cores* como suporte a execução de programas *multithread*. Outros estudos de caso encontram-se documentados em (Araujo et al., 2010). O desempenho é apresentado em termos do *speedup* e consumo. O *speedup* é a relação entre o tempo decorrido entre a execução do programa utilizando apenas um processador virtual (e apenas um *core*, por consequência) e o tempo de execução paralelo. O consumo é expresso pela Equação 2, onde f_i corresponde a frequência de cada *core* e α_i o percentual de uso do circuito de cada unidade de processamento, com $1 \leq i \leq m$, e Δt representando a variação do tempo de execução do programa.

$$Consumo = \sum f_i \times \alpha_i \times \Delta t \qquad (2)$$

Todos os testes foram realizados a partir do custo anotado da execução encontrando dinamicamente o caminho crítico em um grafo DCG descrito pela aplicação. O estudo de caso foi aplicado sobre um algoritmo recursivo para obtenção da computação de posições na série de Fibonacci, sendo o experimento realizado com 2, 4, 8, 12 e 16 processadores virtuais. Esta aplicação caracteriza-se pelo seu alto grau de concorrência e pela possibilidade de identificar o caminho crítico em função do desbalanceamento dos ramos gerados durante a criação recursiva de *threads*.

A Figura 1.a representa o *speedup* para o cálculo Fibonacci de 40 e de 45. Os valores correspondem à média de 20 execuções, tendo sido observado um desvio padrão menor que 5% nos tempos. As curvas registram os desempenhos considerando uma execução regular e uma execução onde a gestão de energia e afinidade é realizada. A Figura 1.b ilustra a demanda de energia para execução do Fibonacci de 43.

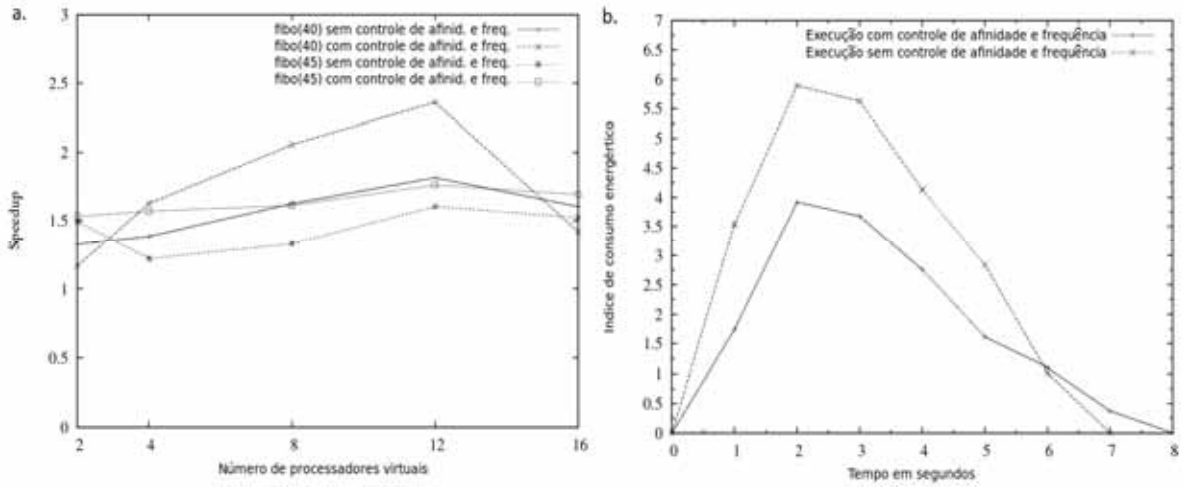

Figura 1: A. Speedup para execução paralela do Fibonacci de 40 e 45 e B. Comparação do consumo energético do Fibonacci de 43.

A partir do grafo da Figura 1.a observa-se que mesmo com utilização de controle de afinidade e frequência a execução obteve um ganho médio

desempenho de 1,05 vezes maior em termos de *speedup*. As curvas da Figura 1.b indicam que a estratégia de escalonamento proporciona uma eficiência energética de aproximadamente 1,53 vezes maior que a execução sem a utilização deste recurso.

A análise combinada dos resultados apresentados nos gráficos da Figura 1.a e 1.b permite concluir que a redução da velocidade de operação de cores não confronta com necessidades de processamento de alto desempenho, pelo menos quando a estrutura do programa é regular e conhecida *a priori*.

5 CONCLUSÕES

Este trabalho apresentou uma estratégica de escalonamento de *threads* para aplicações desenvolvidas a partir da interface *multithreaded* de Anahy, esta estratégia permite a redução do consumo energético considerando-se a carga de trabalho associado ao caminho crítico de um grafo DCG. A heurística para o aumento da eficiência energética introduz uma estratégia básica de escalonamento, onde assumimos que *threads* que não pertencem à execução do caminho crítico, em um grafo DCG, executem em processadores com frequência de operação reduzida ao mínimo suportado pelos processadores, no estudo de caso apresentado neste trabalho 66%, decrementando com isso o consumo energético do sistema sem perda significativa de desempenho ou até seu incremento como pode ser visto em (Araujo et al., 2009). Os resultados obtidos indicam que, pelo menos no caso em que a estrutura de um programa é conhecida *a priori*, é possível fazer uso de uma estratégia de execução que faça uso consciente da energia sem causar perda de desempenho quando considerado o tempo total de execução.

REFERÊNCIAS

ARAUJO, A. S.; CAMARGO, C. A. S.; CAVALHEIRO, G. G. H.; PILLA, M. L. Towards a Power-Aware Application Level Scheduler for a Multithreaded Runtine Envorinment. In I Workshop on Applications for Multi and Many Core Architecture. Gramado, 2010.

ARAUJO, A. S.; CAVALHEIRO, G. G. H. Utilização de Afinidade no Escalonamento de Threads em Multiprocessadores. In IX Escola Regional de Alto Desempenho. Caxias do Sul, 2009.

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. Berlin, 2007.

HILL, M. D.; MARTY, M. R. Amdahl's Law in the Multi-Core Era. IEEE Computer Society. Vol. 41, p. 33-38. Madison, 2008.

UHRIG, S.; UNGERER, T. Energy Management for Embedded Multithreaded Processors with Integrated and Scheduling. In ARCS, Austria, 2005.

WECHSLER, O. Setting New Standards for Energy-Efficient Performance. White Paper, inside Intel Core Microarchitecture, p. 4-5. 2006.

WOLF, W. The Future of Multiprocessor System-on-Chip. In XLI Design Automation Confenre on DAC. Preceeding of the DAC. San Diego. 2004.