Estudo de Caso de Escalonamento de Threads para Redução do Consumo de Energia

Alan Schlindvein de Araujo[*], Gerson Geraldo H. Cavalheiro

[1]Departamento de Informática – Universidade Federal de Pelotas (UFPel)
Caixa Postal 354 – 96.010-900 – Pelotas – RS – Brasil

alan@anahy.org, gerson.cavalheiro@ufpel.edu.br

1. Introdução

A redução do consumo de energia dos sistemas computacionais é um campo de pesquisa muito importante a ser explorado, uma vez que grande parte das novas aplicações não são compatíveis com os anseios da sociedade moderna por economia de recursos energéticos. Uma das soluções apresentadas é a utilização de recursos de *hardware* paralelos, em particular, o uso de arquiteturas multicore. James Reinders[1], diretor de *software* e especialista na área de paralelismo da Intel Corporation, apresenta um estudo de caso sobre processadores multicore, em que ele mostra relações entre frequência de *clock* e a tensão do processador. O estudo realizado por Reinders mostra portanto, que em uma arquitetura hipotética multiprocessada, em que a relação de consumo e desempenho seja, de 0.51 para 0.87 (por cento), respectivamente, ao aumentarmos o consumo para 1.02% incrementamos o desempenho para 1.73%. Logo, arquiteturas multicore são aclamadas por proverem altas taxas de desempenho com uma significativa eficiência energética.

O consumo de energia pode ser obtido, pela fórmula expressa abaixo, como um produto da capacitância (C), quantidade de energia elétrica que pode ser armazenada em um capacitor, pelo quadrado da tensão (V) de entrada e o linear da frequência (f) de operação do processador [Wechsler 2006].

$$Power = C \times V^2 \times f \tag{1}$$

Essa fórmula reflete a dependência da tensão (em Volts) em relação à frequência de *clock*. É possível, portanto, reduzir o consumo de energia pela redução da frequência de processamento. Com isso, temos como consequência direta a perda de desempenho do processador [Uhrig et al. 2005]. Logo, é necessário buscar alternativas para a gestão da frequência de *clock* que não comprometa o desempenho global do sistema.

O estudo sobre a redução do consumo de energia relaciona-se diretamente a computação sustentável, abordada em Green Computing [Kurp 2008]. Um dos focos da *green computing* concentra-se em explorar o potencial computacional de um sistema visando à economia em termos de eficiência energética. Neste contexto, este trabalho é focado em questões relacionadas ao escalonamento de *threads* a nível usuário, em arquiteturas multicore, e elaboração de políticas eficientes para esse escalonamento de tarefas, bem como heurísticas para seu desenvolvimento, que juntamente com a gestão de afinidade, permita reduzir o consumo de energia durante a execução de um programa. O

[*]Bolsista BIC FAPERGS

[1]Informações em: http://news.zdnet.com/2435-13818_22-0.html

objetivo a ser atingido é a introdução de um mecanismo no núcleo de escalonamento de Anahy [Cavalheiro et al. 2006], que permita controlar o consumo de energia dos processadores em função da carga computacional dos *threads* executados.

2. Recursos

No contexto deste trabalho, assume-se que o escalonamento é um mecanismo que permite alocar fluxos de execução (*threads*) ativos aos processadores (ou *cores*) disponíveis em uma determinada arquitetura. Este escalonamento pode ser considerado em nível sistema e nível usuário. O escalonamento sistema é aquele realizado pelo sistema operacional, considerando critérios de exploração eficiente do *hardware* pelo conjunto de aplicações que se encontram em execução. Quando realizado no nível usuário, ou seja, na camada de *software* do usuário, o escalonamento é atento a otimização da execução de um único programa, sendo que as características deste programa podem ser utilizadas para as tomadas de decisão do núcleo responsável pelo escalonamento. Diferentemente do escalonameento no nível sistema, o escalonamento no nível usuário proposto neste trabalho explora atributos próprios de um programa em execução, identificando a carga computacional de cada *thread* em um programa para apoiar as tomadas de decisão de escalonamento. Estas decisões incluem a identificação da afinidade de *threads* por processador e o controle da frequência de operação dos processadores.

A afinidade é o conceito aplicado à programação de processadores com arquiteturas multiprocessadas. Este recurso permite associar um *thread* a um, ou a um grupo, de processadores/*cores*. O uso da afinidade permite a referência aos dados existentes em *caches*, ou seja, *threads* podem ser associados a processadores/*cores* que já contenham em suas *caches* dados que o *thread* manipulará, uma vez que a migração de *threads* entre processadores gera um custo adicional de execução na troca de contexto. Explorar a afinidade não é uma tarefa simples, pois exige do programador um alto grau de atenção tanto codificar sua aplicação quanto identificar quais *threads* deverão ser atribuídos a quais processadores/*cores*, nem sempre retornando ganhos significativos de desempenho [Foong et al. 2008].

Controlar a gestão de frequência a processadores independentes não é uma tarefa trivial, primeiramente a tensão do processador deve ser alterada, para que essa possa suportar a frequência exigida, somente após isto, a frequência pode ser alterada para a desejada. Devido a isto, o processador deve encerrar sua atividade esperando a frequência ser ajusta, ou seja, perdemos tempo de processamento [Uhrig et al. 2005]. Afim de diminuir o impacto na gestão de energia, o monitoramento dessa pode ser obtido com uma nova fórmula baseada na fórmula citada anteriormente. A nova medida para o consumo pode ser expressa como:

$$Consumption = \frac{f \times \alpha}{t} \tag{2}$$

O uso desta técnica para a avaliação do consumo de energia não gera custos de processamento significativos sobre o programa em execução. Não havendo *hardware* para realizar o monitoramento, o que seria o ideal, o uso deste método explora dados sobre a frequência (f) de *clock* e percentual de uso do processador (α), que são fornecidos pelo próprio sistema operacional.

3. Estudo de Casos

O primeiro estudo de caso apresenta o problema de Josephus [Cormen 2001]. Este problema consiste em, inicialmente, criar uma lista circular de trabalhos, com cargas geradas aleatoriamente. Na sequência são realizados, também de forma aleatória, tantos sorteios quanto for o número de trabalhos na lista. A cada trabalho selecionado é disparado um *thread* para processá-lo. O gráfico na Figura 1 apresenta a relação do consumo de energia entre unidades de processamento durante a execução de uma instância deste problema. Como as curvas mostram, sem o uso da gestão de frequência e de afinidade, o consumo de energia é 1.52 vezes maior do que na execução onde estes mecanismos são empregados. A redução de consumo de energia obtida foi de 38.47%.

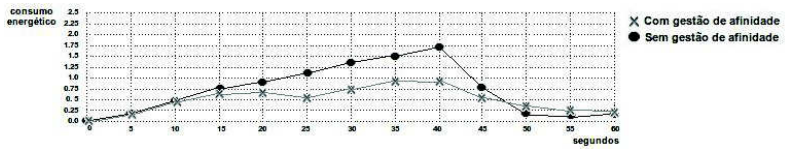

Figura 1. Relação de tempo de execução, frequência e percentual de uso do circuito.

O estudo de caso registrado na Figura 1 foi realizado sobre uma arquitetura *dual-core*. Na geração de cargas de trabalho sintéticas foi considerado que 65% dos trabalhos apresentariam uma carga menor ou igual a 40% da maior carga gerada. Este artifício possibilitou classificar os *threads* como "leves" ou "pesados". Assim, na arquitetura *dual-core*, o *core* 1, responsável pela execução do *threads* leves, teve sua frequência reduzida à 63% da original, o *core* 2, sem alteração em sua frequência, sendo que a política de condução de frequência do sistema operacional seja *performance*, executa os trabalhos pesados. Embora a eficiência energética obtida pelo escalonador tenha sido 13% maior que o esperado, como mostra na Figura 1, em termos de desempenho observou-se que o tempo médio sem a aplicação do escalonamento foi igual a 43,27 segundos, enquanto a execução com o escalonamento proposto gerou um tempo de 46,53 segundos, correspondendo a um decréscimo de desempenho de 7,5%. Destaca-se, neste momento, que os resultados obtidos são dependentes de aplicação, o desempenho pode variar conforme a implementação do programa e como esse faz uso dos recursos disponíveis. Um estudo em que o desempenho e o consumo são satisfatórios é apresentado na sequência.

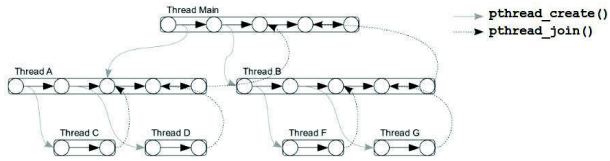

Figura 2. Grafo gerado pela aplicação do método de Monte Carlo sobre threads.

O segundo caso de estudo aplica o método de Monte Carlo para obter aproximações numéricas em funções para realizar o cálculo de integrais. A Figura 2

mostra um grafo de *threads* gerado por uma instância desta aplicação. O problema consiste em, dado um intervalo, usado para a estabelecer os limites da área, dividí-lo em porções para cada *thread*. No estudo de caso realizado, optou-se por gerar um grafo de threads desbalanceado, assim os intervalos das funções não foram divididos de forma equilibrada: os *threads* à esquerda do grafo executam suas operações sobre intervalos maiores que os *threads* à direita.

O programa é executado com e sem o uso do escalonamento, em uma arquitetura *quad-core*, para o primeiro caso, a média do tempo de execução foi igual a 10,02 segundos, enquanto que para a outra aplicação obtivemos um tempo médio de 6,173 segundos. Portanto, há um aumento no desempenho do programa de aproximadamente 38,41%. Em termos de consumo de energia, a aplicação que faz uso do escalonamento proposto apresentou uma economia de aproximadamente 63,2% superior a outra execução. Logo foi obtido um melhor desempenho com aumento na eficiência energética, sem alterações nas propriedades do escalonador e nem em sua implementação.

4. Conclusão

O resultado do uso do escalonamento, apresentado neste trabalho, pode variar de acordo com a aplicação e como essa atua sobre os recursos da arquitetura disponível. Nem sempre pode-se obter um relacionamento ideal entre desempenho e consumo de energia. O presente trabalho não pretende avaliar custos de execução de *threads*, o foco principal é voltado à elaboração de um mecanismo de escalonamento, que inclua heurísticas de distribuição de cargas de trabalho a fluxos de execução ativos em um sistema computacional. Podendo assim, adequar diferentes frequências de operação a processadores/*cores*, os resultados alcançados serão incorporados no núcleo de escalonamento de Anahy.

Referências

Cavalheiro, G. G. H., Gaspaty, L. P., Cardozo Júnior, M. A., and Cordeiro, O. C. (2006). Anahy: A Programming Environment for Cluster Computing. *High Performance Computing for Computational Science*. In. Proc. of the High Performance Computing of Computational Science. Heildelberg: Springer.

Uhrig, S.; Ungerer, T. (2005). Energy Management for Embedded Multithreaded Processors with Intergrated EDF Scheduling. *18th International Conference on Architecture of Computing Systems, Hall in Tirol/Innsbruck, Austria.* Systems Aspects in Organic and Pervasive Computing – ARCS. Março, 2005, p. 1-17.

Wechsler, O. (2006) Setting New Standards for Energy-Efficient Performance. Setting New Standards for Energy-Efficient Performance. White Paper, Inside Intel Core Microarchitecture. Mobility Microprocessor Architecture Intel Corporation. 2006.

Foong, A.; Fung, J.; Newell, D. (2008) Improved Linux* SMP scaling: User-directed processor affinity. Outubro, 2008. *http://softwarecommunity.intel.articles/eng/1781.htm (acesso em 10/12/2008).*

Kurp, P. Green Computing: Are You Feady for Personal Energy Meter? *Scientific American Magazine. Communications of the ACM,* Vol. 51, No. 10. Outubro, 2008.

Cormen, T. H; Leiserson, C. E.; Stein, C. Chapter 14: Augmenting Data Structures. Introduction to Algorithms (Second Edition ed.). MIT Press and McGraw-Hill. ISBN 0-262-03293-7. 2001, p. 318.