Escalonamento ciente de energia com roubo de tarefas

Luís Felipe Millani[1], **Lucas Mello Schnorr**[1], **Nicolas Maillard**[1]

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{lfgmillani,schnorr,nicolas}@inf.ufrgs.br

Paralelismo de tarefas é uma das formas de paralelizar uma aplicação. A computação é dividida em diversas tarefas, que são executadas por diferentes unidades computacionais. Para distribuir as tarefas entre as unidades computacionais frequentemente faz-se uso de um sistema de escalonamento em tempo de execução, como Cilk, X-Kaapi ou OmpSs. Alguns desses sistemas apresentam suporte a heterogeneidade, possibilitando uma implementação diferente para cada arquitetura [Gautier et al. 2013], [Duran et al. 2011]. Uma estratégia frequentemente empregada para fazer o balanceamento de carga entre as diversas unidades computacionais é o roubo de tarefas. Nessa estratégia as unidades computacionais que não possuem tarefas para executar tentam "roubar" tarefas das outras unidades.

A tendência atual é de crescente consumo energético [Beloglazov et al. 2011], resultando em maiores custos e dificuldades em dissipar o calor produzido pelas unidades computacionais. Apesar disso, grande parte dos sistemas em tempo de execução não considera o custo energético ao escalonar as tarefas.

O consumo de energia de uma unidade computacional depende, dentre outros fatores, do tempo de execução, da voltagem e da frequência de *clock* utilizada [Zhuravlev et al. 2013]. Frequências mais altas resultam em menor tempo de execução, contudo aumentam o consumo significativamente. Isso ocorre pois há uma relação aproximadamente cúbica entre consumo e frequência, devido a frequências maiores requererem voltagens maiores.

Para reduzir o custo energético os processadores atuais permitem que a voltagem e a frequência na qual operam sejam alteradas em tempo de execução. Isso permite o uso de técnicas de *dynamic voltage and frequency scaling* (DVFS), que alteram a voltagem e a frequência durante a execução de uma aplicação de forma a reduzir o consumo mas sem sacrificar completamente o desempenho [Hsu and Feng 2005]. Para diminuir a perda de desempenho que frequências menores causam, reduz-se voltagem e frequência nos momentos que menos dependem da velocidade do processador, como intervalos de comunicação ou períodos de inatividade [Wang et al. 2010].

O consumo de energia pode ser obtido para apenas alguns componentes ou para o sistema como um todo. Em ambos os cenários, o consumo pode ser mensurado utilizando-se equipamentos específicos [von Laszewski et al. 2009]. Esse método é transparente para a execução cujo consumo deseja-se mensurar, não adicionando sobrecarga. Contudo, ele requer que o equipamento seja fisicamente colocado na computador cujo consumo será mensurado, o que requer esforço proporcional ao número de computadores.

Outra possibilidade para mensurar o consumo é através do uso de recursos disponíveis em alguns hardwares. O consumo total da placa-mãe pode ser obtido por meio da *Intelligent Platform Management Interface*, que permite obter os dados do *Baseboard Management Controller*, que é um microcontrolador presente em algumas placas-mãe. Em

alguns modelos de processadores Intel dados de consumo podem ser obtidos através de alguns dos *Model-Specific Registers*, mais especificamente pelas interfaces RAPL (*Running Average Power Limit*). Nas placas de vídeo NVIDIA Tesla e Quadro dados de consumo da GPU inteira, inclusive memória, podem ser obtidos pela NVIDIA Management Library. Para utilizar os recursos disponíveis em hardware para mensurar o consumo é necessário executar código para tal durante a execução do aplicativo que deseja-se mensurar. Esse processo intrusivo adiciona certa sobrecarga, ou seja, o consumo mensurado e o consumo da aplicação sem a mensuração diferem. Além disso, é necessário suporte no kernel do sistema operacional.

O objetivo do trabalho a ser desenvolvido é incorporar técnicas de conservação de energia, como DVFS, em sistemas em tempo de execução baseados em roubo de tarefas. Espera-se, com isso, obter redução no consumo de energia sem perda excessiva de desempenho. A implementação dessas técnicas será comparada com a versão que não considera o custo energético, de forma a verificar os efeitos em consumo e desempenho. Essa comparação será feita através de *benchmarks* como Cholesky, fatoração LU, entre outros.

Referências

Beloglazov, A., Buyya, R., Lee, Y. C., Zomaya, A., et al. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82(2):47–111.

Duran, A., Ayguadé, E., Badia, R. M., Labarta, J., Martinell, L., Martorell, X., and Planas, J. (2011). Ompss: a proposal for programming heterogeneous multi-core architectures. *Parallel Processing Letters*, 21(02):173–193.

Gautier, T., Ferreira Lima, J. V., Maillard, N., and Raffin, B. (2013). XKaapi: A Runtime System for Data-Flow Task Programming on Heterogeneous Architectures. In *27th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Boston, Massachusetts, États-Unis.

Hsu, C.-h. and Feng, W.-c. (2005). A power-aware run-time system for high-performance computing. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, pages 1–, Washington, DC, USA. IEEE Computer Society.

von Laszewski, G., Wang, L., Younge, A., and He, X. (2009). Power-aware scheduling of virtual machines in dvfs-enabled clusters. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, pages 1–10.

Wang, L., von Laszewski, G., Dayal, J., and Wang, F. (2010). Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 368–377.

Zhuravlev, S., Saez, J., Blagodurov, S., Fedorova, A., and Prieto, M. (2013). Survey of energy-cognizant scheduling techniques. *Parallel and Distributed Systems, IEEE Transactions on*, 24(7):1447–1464.