Suporte à Execução Eficiente de Aplicações em Plataformas com Paralelismo Multi-Nível

Vinícius Garcia Pinto, Nicolas Maillard

[1]Grupo de Processamento Paralelo e Distribuído - Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS) - Porto Alegre - RS - Brasil

{vgpinto, nicolas}@inf.ufrgs.br

Resumo. *Este trabalho caracteriza o problema do suporte à execução eficiente de aplicações em arquiteturas paralelas atuais. São apresentados alguns mecanismos que possibilitam ao programador explorar de forma mais produtiva e eficiente o potencial de paralelismo dos sistemas de HPC contemporâneos.*

A emergência, em um futuro próximo, de sistemas exascale com capacidade de processar 10^{18} operações de ponto flutuante por segundo tente a acentuar a dificuldade em explorar eficientemente o potencial de paralelismo oferecido pelos sistemas de HPC.

Os sistemas HPC atuais possuem diversos níveis de processamento paralelo e de hierarquia de memória. O Tianhe-2, supercomputador mais rápido do mundo, possui 16 mil nós de processamento. Cada nó possui 2 processadores Intel E5-2692v2 com 12 *cores* físicos e 3 coprocessadores Intel Xeon Phi 31S1P com 57 *cores* cada. A hierarquia de memória contabiliza 1 PB, distribuídos em cada nó em 64 GB de memória RAM no *host* e 24 GB nos coprocessadores[1].

O projeto e a programação de aplicações que executem de forma eficiente nesse cenário de *hardware* têm se tornado um desafio. A solução mais frequente é a adoção de APIs específicas para cada camada de paralelismo do *hardware* da arquitetura. Podemos citar como exemplo o uso de MPI para explorar o paralelismo entre os nós, OpenMP para entre os núcleos do processador e CUDA para aceleradores como GPUs. Essa combinação de diferentes ferramentas, mesmo que ainda ofereça desempenho satisfatório em alguns cenários, tem gerado aplicações menos legíveis e pouco portáveis ao custo de um esforço de programação considerável [Asanovic et al. 2009].

Além disso, soluções pontuais tendem a ser difíceis de serem portadas em casos de necessidade de alterações parciais no *hardware* e/ou no *software*. Uma forma de viabilizar e facilitar a exploração do paralelismo nestas plataformas altamente paralelizadas e hierarquizadas é a utilização de um mecanismo de abstração sobre os recursos de processamento e espaços de memória.

Um mecanismo de abstração pode auxiliar os programadores na implementação das aplicações permitindo expressar o paralelismo e ao mesmo tempo abstrair detalhes da arquitetura computacional. Esse mecanismo pode ser composto de uma API e de ambiente de execução [Asanovic et al. 2009].

O programador descreve sua aplicação com uma API, expondo ao máximo o paralelismo do problema, desconsiderando detalhes de *hardware*, como por exemplo, o número de *cores* da arquitetura. O código do usuário é então executado sobre um ambiente de execução que procura implementar a API da maneira mais eficiente possível.

[1]http://www.top500.org/system/177999

O ambiente de execução trata de aspectos como escalonamento e atribuição das tarefas entre os recursos de processamento, balanceamento de carga, transferências de memória e tolerância a falhas. Além destas questões recorrentes em aplicações paralelas, o ambiente de execução também pode considerar desafios mais recentes como gerenciamento do consumo de energia e resiliência.

Diversos ambientes de execução têm sido propostos, em especial, para facilitar a utilização de plataformas compostas por recursos de processamento heterogêneos. Embora tais ambientes tenham objetivos comuns, as soluções adotadas e os sistemas alvo divergem. Existem ferramentas direcionadas para sistemas com memória distribuída enquanto outras concentram-se em explorar o paralelismo de nós multiprocessados auxiliados por aceleradores.

Ambientes de execução como o WORMS [Pinto and Maillard 2012] e o XKaapi [Gautier et al. 2013] suportam sistemas com CPUs e GPUs utilizando algoritmos de balanceamento de carga por roubo de tarefas. Porém, o XKaapi oferece uma abstração entre os espaços de memória, permitindo expressar dependências entre tarefas por meio dos dados acessados e sobrepor transferências de memória e computação automaticamente.

O StarPU [Augonnet et al. 2010] oferece uma abstração sobre os espaços memória de arquiteturas híbridas constituídas de recursos como *multicores*, GPUs e Cell BE. Além disso, a ferramenta escalona as tarefas com base em modelos de desempenho de execuções prévias que consideram tempo de execução e transferências de memória.

O ambiente de execução Charm++ [Vasudevan et al. 2013] provê balanceamento dinâmico de carga em arquiteturas com memória compartilhada e distribuída. A versão G-Charm acrescentou suporte à GPUs incluindo sobreposição entre transferências de dados e computação possibilitando o aumento da taxa de ocupação dos recursos.

Este trabalho visa explorar e validar técnicas que permitam a execução eficiente de aplicações paralelas em plataformas paralelas atuais. As etapas futuras deste trabalho consistem em identificar, projetar e avaliar soluções para escalonamento, balanceamento de carga, abstração sobre diferentes níveis de memória e algoritmos que auxiliem a decisão sobre divisão e atribuição de trabalho em arquiteturas híbridas.

Referências

Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., Wessel, D., and Yelick, K. (2009). A view of the parallel computing landscape. *Communications of the ACM*, 52(10):56–67.

Augonnet, C., Thibault, S., and Namyst, R. (2010). StarPU: a Runtime System for Scheduling Tasks over Accelerator-Based Multicore Machines. Research Report RR-7240.

Gautier, T., Lima, J. V. F., Maillard, N., and Raffin, B. (2013). XKaapi:A Runtime System for Data-Flow Task Programming on Heterogeneous Architectures. In *Proc. 27th Int. Symp. Parallel and Distributed Process.(IPDPS)*, pp. 1299-1308. IEEE Comput. Soc.

Pinto, V. G. and Maillard, N. (2012). Work stealing on hybrid architectures. In *13th Symp. on Computer Systems (WSCAD-SSC)*, pp. 17-24, Los Alamitos. IEEE Computer Soc.

Vasudevan, R., Vadhiyar, S. S., and Kalé, L. V. (2013). G-Charm: An Adaptive Runtime System for Message-driven Parallel Applications on Hybrid Systems. In *Proc. 27th Int. Conf. on Supercomputing*, ICS '13, pages 349–358, New York, USA. ACM.