Programação em Ambientes Computacionais com Múltiplos Níveis de Paralelismo

Claudio Schepke, Nicolas Maillard

[1]Instituto de Informática - Universidade Federal do Rio Grande Do Sul
Caixa Postal 15.064 - CEP 91.501-970 - Porto Alegre - RS - Brasil
{cschepke,nicolas}@inf.ufrgs.br

Paralelismo Multinível

No transcorrer dos últimos anos tem-se visto um grande número de sistemas computacionais à disposição no mercado, com uma considerável gama de recursos que satisfazem as necessidades dos desenvolvedores de software. Geralmente, tais soluções estão baseadas em arquiteturas paralelas. O uso de máquinas vetoriais, multiprocessadas e de sistemas *multi-core* têm sido algumas das alternativas exploradas. Tais tecnologias podem ainda ser combinadas através da formação de *clusters* e *grids*. Diante disso, chega-se à composição de plataformas com múltiplos níveis de paralelismo.

Um paralelismo multinível ocorre quando existem diversos níveis de abstração paralela. Isto pode ocorrer nos próprios processadores (*multi-core*), internos a um computador (multiprocessados) ou entre vários computadores (*clusters* e *grids*). O gerenciamento de cada um destes níveis de abstração é feito através de mecanismos específicos em nível de processador, núcleo do sistema operacional e *middleware* de gerenciamento.

Nesse contexto, este trabalho discute a programação paralela em ambientes com múltiplos níveis de paralelismo, apresentando os principais recursos de programação atualmente utilizados em cada nível de paralelismo, bem como algumas propostas atualmente existentes que exploram a programação em multiníveis.

Recursos de Programação

As APIs de programação paralela atualmente existentes são bastante específicas, focando apenas um determinado nível de paralelismo. Assim, por exemplo, para a programação com *threads* geralmente é utilizado o padrão **Posix Threads**, para a programação com multiprocessadores usa-se **OpenMP** e para a comunicação interprocessos são adotados os padrões **MPI** e **Java RMI** [ANDREWS(2001)]. Embora estes recursos tenham se tornado um padrão para cada tipo de ambiente, dificilmente eles oferecem o mesmo desempenho quando portados para outro. Algumas tentativas de superar essas limitações já estão sendo propostas:

- **UPC** - É uma extensão da linguagem de programação C, para a computação de alto desempenho em máquinas paralelas de larga escala, desenvolvida em Berkeley [YELLICK; BONACHEA; WALLACE, 2004]. A linguagem disponibiliza um modelo de programação uniforme para sistemas de memória compartilhada e distribuída. UCP usa o modelo de programação *Single Program Multiple Data* (SPMD) onde o paralelismo é pré-definido antes da execução, sendo que cada fluxo de execução é destinado a um processador. Assim, o ambiente é visto como um único sistema de memória compartilhada no qual os processadores podem ler

e escrever as variáveis, embora estas sejam fisicamente associadas a um único processador.

- **Chapel** - É uma lingagem de programação desenvolvido pela Cray, sendo parte do projeto conhecido por *Cascade* [CHAMBERLAIN; CALLAHAN; ZIMA(2007)]. Chapel prove um nível mais alto de abstração para a expressão de programas paralelos do que outras linguagens de programação. Além disso, a linguagem oferece uma separação entre o desenvolvimento do algoritmo e os detalhes da implementação das estruturas de dados. Chapel suporta um modelo de programação *multithreaded*, oferecendo abstrações para paralelismo de dados e tarefa.
- **X10** - É uma linguagem de programação experimental desenvolvida pela IBM em parceria com instituições acadêmicas [CHARLES et al.(2005)]. O objetivo da linguagem é oferecer novas técnicas de implementação que ofereçam um paralelismo escalável e optimizado para um ambiente gerenciado em tempo de execução. X10 oferece todos os recursos clássicos da linguagem de programação Java, tanto para ambiente SMP, como para *clusters*.
- **Co-array** - Possui propriedades semelhantes à implementação UPC, sendo implementada em Fortran [NUMRICH; REID(1998)].

Todas essas ferramentas acabam funcionando como uma camada de abstração, tornando homogêneo o mecanismo de implementação. Entretanto, não é possível extrair paralelismo em todas camadas simultaneamente, visto que estas nem sempre foram abstraidas adequadamente, como é o caso das arquiteturas *multi-core*. Desta forma, tem-se um ponto de pesquisa aberto, a fim de obter o paralelismo em um número maior de níveis.

Referências

ANDREWS, G. R. **Foundations of Multithreaded, Parallel, and Distributed Programming**. Reading: Addison-Wesley, 2001. 664p.

CHAMBERLAIN, B. L.; CALLAHAN, D.; ZIMA, H. P. Parallel Programmability and the Chapel Language. **International Journal of High Performance Computing Applications**, [S.l.], v.21, n.3, p.291–312, August 2007.

CHARLES, P.; GROTHOFF, C.; SARASWAT, V.; DONAWA, C.; KIELSTRA, A.; EBCIOGLU, K.; PRAUN, C. von; SARKAR, V. X10: an object-oriented approach to non-uniform cluster computing. **SIGPLAN Not.**, New York, NY, USA, v.40, n.10, p.519–538, 2005.

NUMRICH, R. W.; REID, J. Co-array Fortran for parallel programming. **SIGPLAN Fortran Forum**, New York, NY, USA, v.17, n.2, p.1–31, 1998.

YELLICK, K.; BONACHEA, D.; WALLACE, C. **A Proposal for a UPC Memory Consistency Model, v1.0**. [S.l.]: Lawrence Berkeley National Lab, 2004. Technical report. (LBNL-54983).