Models and Cost Analysis for Parallel Programs with Multiple Threads

Airam Jônatas Preto
Laboratório de Computação e Matemática Aplicada
Instituto Nacional de Pesquisas Espaciais
airam@lac.inpe.br

John Richard Gurd
Centre for Novel Computing
Manchester University
john@cs.man.ac.uk

August, 3 1998

Resumo

As especificações de programas paralelos tratam particularmente de padrões de interação entre objetos no programa. Conforme a computação em paralelo é executada, os vários objetos evoluem, bem como o sistema como um todo. É muito provável que esta evolução dos objetos individualmente se dê em taxas diferentes, o que pode exigir a sincronização de certos eventos para que a consistência da computação como um todo seja mantida. Portanto, é interessante o emprego de meios de modelagem da evolução de objetos que possam abstrair seus padrões de interação, inclusive a maneira com que se sincronizam. Estes meios de modelagem também devem permitir a construção de sistemas mais complexos a partir de componentes mais simples. Finalmente, a metodologia deve incluir expressões de custo que permitam a análise de desempenho em termos de tempo de execução. Estes objetivos podem ser atingidos através do emprego de uma técnica de modelagem de programas que represente a especificação de interações entre objetos num programa paralelo como uma coleção de expressões de processos.

Keywords: Parallelism modelling, cost analysis, π-calculus, formal specifications, data encapsulation, programming skeletons.

The behaviour of parallel programs should be predictable from their specifications. In order to satisfy this requirement, it is desirable to have a strong correlation between the formal system and the programming system. Many different schemes have been proposed to bridge the gap between the formal world and the programming world, in particular those which rely on procedural and data abstraction [1, 2, 3]. These schemes satisfy the requisites for program modelling and support for reuse. However, they fail or are incomplete in terms of support for performance analysis.

The criteria for evaluation of parallel programming models should include: data and parallelism abstraction, architectural independence, verification of properties, abstraction-level reuse and performance estimation. Some programming models have been proposed recently which try to address these issues. Important examples of this research are the works on Categorical Data Types [5] and Bulk Synchronous Parallel Computing [6].

This paper proposes a technique to model parallel programs based on an alternative approach, a compositional methodology[4], which consists of three basic components:

- A description of problem entities in terms of abstract data types according to a dynamic binding approach.
- An algebraic representation for computations involving data and/or task parallelism. From this representation, an analytical model for performance is derived, based on the estimation of execution time.
- A library of basic templates (skeletons) which support operations on abstract data types and can be composed to implement different forms of parallel computations.

One of the basic features of a compositional programming technique is that a composition preserves the properties of its parts. This feature not only prevents the introduction of side effects when combining different software components, but also creates opportunity for their reuse. This paper proposes to describe the semantics of parallel computations in terms of collections of process expressions. The advantage of using such algebraic representation is that it allows one to define equivalence between expressions. Once a formal definition of equivalence is established, it becomes possible to build up mathematical rules that can be used to reason about the models and to demonstrate that programs exhibit some desired properties.

A useful model for parallel programming should provide a mechanism for determining the costs of an implementation (execution time and memory usage) early in the development cycle and in a way that is not closely tied to a certain architecture. During the development, these cost measures provide a means of evaluating the implications of using one implementation in preference to another.

It is desirable to formalise these measures in a calculus that allows one to determine the cost of parts of a program and of their compositions. A cost calculus can thus be expressed by a set of equations that may help, or even automate, the cost analysis during development of an implementation.

An immediate by-product of an algebraic description of a program is an analytical model for performance. Process expressions can be viewed as a formalisation of state transitions which correspond to the actions executed by objects in a parallel program. The time required to perform each transaction can be quantified, and the values so obtained can be associated with variables in a time expression. Such time expressions are translations of process expressions into the time domain.

An experimental framework to determine the costs derived from the modelling technique proposed above is beyond the scope of this paper. Some experimental results have been obtained with the implementation of Poisson Solvers, Airshed Simulation, Multigrid and FFT algorithms. These results are left to be presented in a subsequent paper.

References

[1] M. Cole. *Algorithmic Skeletons.* Pitman, 1989.

[2] M. Danelutto, R. di Meglio, S. Orlando, S. Pelagatti, and M. Vaneschi. "A Methodology for the Development and the Support of Massively Parallel Programs". *Future Generation Computer Systems*, 8:205–220, August 1992.

[3] J. Darlington, Y. Guo, H. W. To, and J. Yang. "Parallel Programming Using Skeleton Functions". In *Parallel Languages and Architectures, Europe: Parle '93*, June 1993.

[4] A. J. Preto. *"Composing Systems with Multiple Threads"*. PhD thesis, Department of Computer Science, University of Manchester, 1997.

[5] D. Skillicorn. *Foundations of Parallel Programming.* Cambridge University Press, 1994.

[6] L. G. Valiant, T. Cheatham, A. Fahmy, and D. C. Stefanescu. "Bulk Synchronous Parallel Computing: A Paradigm for Transportable Software". Technical Report TR-36-94, Center for Research in Computing Technology, Harvard University, December 1994.