A SIMULATION METHODOLOGY FOR THE PREDICTION OF SPMD PROGRAMS PERFORMANCE

P.M.A. Sloot, J.F. de Ronde, M. Beemster, L.O. Hertzberger.
Parallel Scientific Computing and Simulation Group
Department of Computer Science - University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
The Netherlands

ABSTRACT

A simulation methodology for predicting the performance of data parallel (SPMD) programs on parallel and distributed platforms is presented. The proposed methodology comprises a parameterised description of the applications as well as the target machines. An integrated simulation procedure estimates the time complexity of the SPMD application given a well defined hardware platform and predicts the execution behaviour.

1. INTRODUCTION

Migration of large sequential applications to parallel and distributed platforms is yet still in its infancy. This is largely due to the fact that industry (that has many large applications of great social and economical relevance) is reserved towards spending great amounts of money and labour on adjusting their codes to parallel distributed memory machines (DMM) while it is not clear that the effort spent will pay off eventually. A simulation environment that can predict the performance of SPMD applications on parallel DMM is of crucial importance in the decision phase of such large projects. In addition it can assist in choosing the most suitable parallel platform for a specific application (e.g. a workstation network or a massively parallel monolith). We have developed a technique that can predict data parallel program performance by means of parameterisation of the distributed memory machine as well as the data parallel application. The Amsterdam Simulation Toolkit (AST) consists of several tools to obtain these parameterisations and to perform simulations using formal machine and application descriptions (that is mapping the parameterised application to the parameterised machine) The AST is partially developed within an ESPRIT III project (Sloot and Reeve 1993).

2. THE AMSTERDAM SIMULATION TOOLKIT (AST)

The AST consists of several tools:
1) A machine database that to a high level of detail describes DMM characteristics.
2) A symbolic application description language in which the SPMD code of interest can be expressed.
3) A mapping procedure to find an optimal mapping of segmented data to a specific processor topology.
4) A simulation environment that integrates these three items to obtain an estimation of both the time complexity of the application and the execution behaviour.
An application simulator is used to support the development of the several tools (deRonde, Sloot et al).

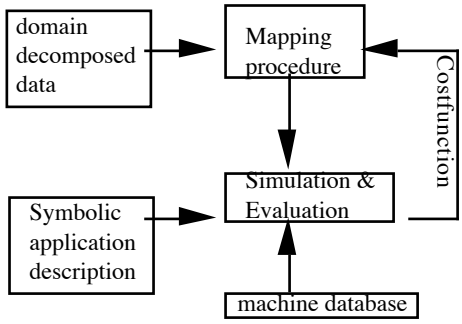

Figure I: overview of the AST

2.1 THE MACHINE DATABASE

A model that can represent (up to a predefined level of detail) the characteristics of (virtual) DMM has been developed. It is validated by means of timings on real machines (e.g. a Parsytec 512 T800 transputer platform (Parsytec 1991)). For the development of the machine model the machine parameters that influence the performance of data parallel applications had to be identified:

1) communication parameters: e.g. hardware and virtual topologies, latencies, throughput, start-up time for a message and routing (allocating) processes.
2) processor parameters: e.g. time constants for addition, multiplication, division etc... for different numeric types and flow control performance (branch costs-> pipeline efficiency).
3) memory parameters and caching: in the model it is assumed that enough memory is available. This in order to avoid modelling of I/O during execution. Memory speed (instruction and operand fetch) is incorporated in other parameters. Cache speeds are possible to time but cache size is impossible to be used without execution of target applications.
4) "programming models": We consider SPMD programming models supported by PVM (Sundaram 1990), EXPRESS (Parasoft 1992) and MPI (Lusky 1993) Of these message passing environments the most cost intensive primitives must be modelled: *spawn* (create a process), *synchronise* (synchronisation between processors), *send/receive* (message to/from neighbours), *multi-cast* (send a message to a group).
5) I/O is not modelled.

2.2 THE SYMBOLIC APPLICATION DESCRIPTION

We describe SPMD programs by the following functional hierarchy:

1) Statement block level
2) Control flow level
3) Data locality level

1) The time complexity of a so called statement block is given by cumulation of all the individual time complexities occurring in the block. For example an expression like:

```
a = a + b * c
```

has a time complexity of Tassign + Tmultiplication + Taddition. The corresponding actual time measures in such formal expressions are filled in by the machine database (the parameterised machine description).

2) The control flow level introduces indeterminability into the time complexity description. In general the execution path taken, given a specific set of input parameters, is only determinable by means of explicit execution. We approach this level in a (quasi) statistic manner by describing the possibility of branching in some specific direction along the execution graph. Branch directions in *if..then..else* constructs are specified by probabilities *P1, P2..,.PN* (where we assume the number of branching directions = *N*) whereas in *loop c*onstructs the number of unkown iterations is presented by a stochastic variable that behaves according to a (user defined) probability density function.
So a parameterised description on these two levels leads to a mixture of the time-complexities of basic statement blocks, probabilities and stochastic variables.

3) The locality level describes the fact that some fraction *X* of data is involved in communication and the remaining part *(1-X)* is not. In case of static domain decomposition these fractions are constant. In case of programs where these fractions are forced to change dynamically it is necessary to introduce a stochastic description.

2.3 THE MAPPING PROCEDURE

The mapping procedure aims to allocating data parallel processes on a multi-processor system such that the time complexity function T is minimised. In other words T is a cost function that has to be optimised using some generic optimisation algorithm such as simulated annealing or genetic algorithms. Assuming that an optimal workload balancing over the processes already has been achieved this implies that the communication effort between processors has to be minimised: Neighbouring processes should be on (physically) neighbouring processors.

2.4 THE SIMULATION ENVIRONMENT

The symbolic description of the program of interest consists of stochasts and statement block time complexities. A specific actualisation of all the parameters present will result in a number that represents the time consumption for the specific parameter actualisation.
It is quite probable that parameters that were initially described as stochasts turn out to be dependent on input parameters in a simple straightforward manner. For example loop boundaries. It is necessary to identify these parameters and replace them by their absolute behaviour instead of describing them in a stochastic manner. Furthermore identification of dominant parameters in the time complexity formula and the actual program part they originate from is required. If necessary the program expert can supply background information on the behaviour of such parts. For example some loop boundary can turn out to have a well determinable stochastic behaviour dependent on several input parameters.
The symbolic description therefore has to allow for reversibility: given a certain parameter the program part from which it originates must be tractable.

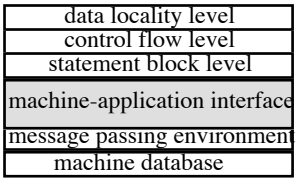

Figure II: parameterised application mapped on parameterised machine

3. CONCLUSIONS

The simulation methodology introduced offers the possibility to estimate the time consumption of SPMD applications on arbitrary parallel distributed memory machines. All tools are in their final stage of development and validation. Pilot experiments indicate that this technique can supply us with reasonable estimates of data parallel programs performance. We feel that this toolset can have a significant contribution in lowering the barrier for industry to use DMM. Our current work concentrates on validating the toolset by porting a very large complex crashworthiness code to a massively parallel machine.

Acknowledgements:

Part of this research is funded by the Commission of European Communities within the Esprit Framework under project number: NB 6756. The excellent contributions of Berry van Halderen and Arjan de Mes to the implementation of the toolset are gratefully acknowledged.

References:

R. Lusky and B. Knighten.1993. "Minutes of the Message Passing Interface Forum" Dallas, Texas. May 12 -14 1993.

Parasoft 1992. "Express User Guide version 3.2"

Parsytec 1991. "Parsytec GC, technical Summary" version1.0

J.F de Ronde and P.M.A. Sloot 1993. "The Application Simulator". Technical Report: CAMAS- TR-2.3.1.1. ESPRIT III. (May).

P.M.A. Sloot and J. Reeve. 1993. "The CAMAS Workbench". Technical Report: CAMAS- TR-2.1.1.2 ESPRIT III. (March).

V.S. Sundaram. 1990. "PVM a framework for parallel distributed computing " Concurrency and practice, vol.2(4) (December): 315-339