Mermaid : Modelling and Evaluation Research in MIMD ArchItecture Design

A.D. Pimentel J. van Brummen T. Papathanassiadis
P.M.A. Sloot L.O. Hertzberger

Dept. of Computer Systems, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
{andy,brummen,theopapa}@fwi.uva.nl

Abstract. The Mermaid project focuses on the construction of simulation models for MIMD multi-computers in order to evaluate them and to give estimates of the system's performance. A multi-layered approach was adopted in which three levels can be distinguished. The application level describes application behaviour in an abstract manner, unrelated to any hardware or architecture specifics. Subsequently, the generation level translates these application descriptions to a hardware dependent trace of operations that drives the simulators. And finally, the architecture level consists of the trace-driven architecture simulation models.
Preliminary simulation results show that, despite the high abstraction level at which is simulated, good accuracy can be obtained.

1 Introduction

With simulation, new computer architectures can be evaluated and tested long before the actual hardware is built. Such evaluation can either focus on functional or performance behaviour. In the scope of the Mermaid project MIMD multi-computer architectures are simulated from a performance perspective [van Brummen94].

Two guiding principles were formulated at the beginning of the project. The level of architecture abstraction should allow simulation within reasonable time, preferably avoiding low-level (bus-cycle) emulation. And, architecture choices are important design options which must be simulated without too much remodelling effort.

The (distributed memory) MIMD architectures modelled within Mermaid are the GCel and PowerStone architectures of Parsytec [Langhammer93].

As a starting point for our research we decided to restrict application modelling to applications adhering to the SPMD (Single-Program, Multiple-Data) programming model [Gupta91], which is considered to be a popular model when programming massively parallel MIMD platforms.

The remainder of this article is structured in the following manner. In section 2, the simulation environment of Mermaid is described. Section 3 briefly describes the architecture simulation models. In section 4, the different levels of application modelling are described. In section 5 some preliminary simulation results are presented. Finally, in section 6 conclusions are drawn.

2 The Mermaid simulation environment

To evaluate MIMD multi-computers, a parameterized algorithmic model was created. It is capable of supplying the simulator with a trace of events, called *operations*, representing processor activity, memory I/O, and communication message passing.

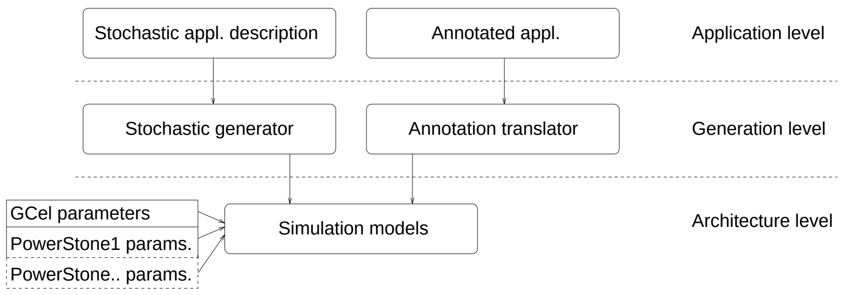

Fig. 1. Simulation scheme.

Simulation of an application load on an architecture takes place at three different levels, as depicted in Figure 1. The application level contains application descriptions used as input to our simulation models. This is done by either stochastically describing the behaviour of programs using probabilities or by instrumenting real programs with annotations representing the exact execution behaviour.

At the generation level a trace of operations is generated from the application descriptions. This generation process exploits knowledge of the target architecture and runtime model in order to tune these operation traces.

Finally, the architecture level consists of operation trace-driven simulation models. Every model has a set of machine parameters that has been calibrated by either published information or benchmarking. To allow simulation within reasonable time, these simulation models do not fully emulate the hardware and keep limited state information during simulation.

2.1 Computation versus communication

Simulation of application behaviour was split into two different models. This because most SPMD applications exhibit a behaviour in which coarse-grained computations are alternated with periods of communication and latencies of computation and communication are different in terms of architecture modelling. Both models define their own set of operations and function at different abstraction levels. The first model is typical for the application's computational behaviour. It drives the second model by providing it with information about computation at the level of computational tasks. Subsequently, the second model is typical for the application's communication behaviour. It implements the computational tasks, derived from the computational model, as delays between communication requests.

3 Architecture modelling

As application behaviour consists of computational and communication behaviour, two different architecture models of the MIMD platform are required. The architecture model for computation involves one single-node of the MIMD platform, whereas communication is modelled by a multi-node model. Both models, implemented in Pearl [Muller93], are generic in the sense that both the GCel and the PowerStone architectures can be represented by means of parameterization. The base computational architecture model consists of the MIMD node's processors, caches, bus and memory. These components are fully parameterized to be able to evaluate design options at board-level.

The base architecture model for communication implements the MIMD node in a very abstract manner as a processor, a router and four communication links. The nodes are linked together resulting in a multi-node model to reflect the platform's interconnection scheme. The routing strategy is parameterized allowing for performance evaluation of a range of strategies.

The communication load of an SPMD application results in an operation trace for each node in the multi-node model. This approach allows for all kinds of synchronization and load balancing scenario's.

4 Application modelling

At the current stage of the project trace generation is only possible through instrumentation of applications. Formalisms to stochastically describe application behaviour are being devised. Therefore, this section describes the methods used for application modelling by means of annotations. Additionally, the idea of an SPMD communication model is presented.

4.1 Modelling computation

It was decided to restrict modelling of the computational behaviour of an application to just those parts of the application being computationally intensive. These parts are often referred to as the *kernel functions* of the application. They are the most important ingredient to the overall performance estimate of the application.

At the application level, these kernel functions are instrumented with annotations that follow the original control flow of the function and represent the function's memory and computational behaviour.

At the generation level, the annotations are translated to a representation of basic operations. This translation is guided by a description of variables used in the kernel function. Such a description provides information on the type and location of variables, i.e. register based.

4.2 Modelling communication

Annotations representing message passing directly map onto the operations of the communication model. Therefore, only communication defined on the physical topology can be modelled. Support for modelling message passing within application-defined interconnection structures, called virtual topologies [Rottger94], is not yet available.

4.3 The SPMD communication model

Modelling of communication that is strictly defined on the architecture's physical communication topology still reflects all of the underlying hardware characteristics. Ideally, no architectural details are visible at the level of application modelling. By modelling virtual communication topologies, such details can more or less be hidden. In this section, we will go one step further and present an SPMD communication model that abstracts from all kinds of explicit communication at the application level. This model will be used as a basis for future development of formalisms expressing application's communication behaviour.

At application level, the SPMD communication model regards communication behaviour as an implicit result of accessing distributed data in one shared address space. This approach is similar to that originally proposed by [Kennedy88] and later adopted by the HPF-FORTRAN initiative [HPF-Forum93]. The SPMD processes share a single program which must be accessible from every processor. The virtually shared data space incorporates the SPMD application's distributed data. Access to this distributed data will result either in a memory request or in a communication request depending on its distribution scheme.

This model requires the issue of synchronization to be carefully considered in order to avoid data hazards. These hazards are similar to the data hazards with instruction pipelining [Hennessy90].

At generation level, the model translates the SPMD data requests to communication requests within a pre-established virtual communication topology. At the level of virtual communication the connectivity of the unconstrained point-to-point SPMD data request is reduced to just those connections available within the virtual topology. After generating these virtual communication requests the generation level takes care of translating them further to physical communication requests.

5 Experiments

This section describes the results of the initial validation of the separate architecture models of the GCel platform. Simulation results of a number of computational and communication benchmarks are compared with the results of real execution.

5.1 Computation

The computational benchmarks are described in Table 1. The *ddot* and *daxpy* benchmarks have been executed several times, with each run working on a different data size.

Table 1 also shows the average error, the standard deviation σ of the average error and the worst-case error of simulation compared to execution. As can be seen from the results, the simulation model behaves reasonable well, even for the more complex *loop18* and *dipole* benchmarks. With a sufficiently low average error margin and a worst-case error that is not exceeding the 7.9 percent, good accuracy is obtained.

Benchmark	Description	Av. Error %	σ	Worst-case Error %
ddot	A double precision innerproduct	4.4	1.9	6.7
daxpy	A double precision vector update	5.2	2.2	7.9
loop18	Loop 18 of the Livermore Fortran Kernels	3.2	0.0	3.2
dipole	The kernel function of an elastic light scattering simulation	3.8	0.0	3.8

Table 1. The computational benchmarks.

5.2 Communication

The communication benchmarks are described in Table 2. The *ping-pong* benchmark is used for fine-grain validation of the communication model, whereas the other three benchmarks are used for stress-testing. The communication behaviour of these benchmarks is extreme, and is not likely to be found in real SPMD applications.

Benchmark	Description	Average Error %	σ	Worst-case Error %
ping-pong	A single message is sent to a node after which it is returned	1.8	1.7	6.9
all-to-all	A synchronized all-to-all communication load in which every node communicates with all other nodes	4.2	2.9	10.3
equal-dist	A synchronized point-to-point communication load in which every node communicates with one fixed partner, resulting in approximately equal-distanced routing paths	9.7	10.4	32.8
unequal-dist	A synchronized point-to-point communication load in which every node communicates with one fixed partner, resulting in a large variety of routing paths	9.1	7.9	31.9

Table 2. The communication benchmarks.

The benchmarks have been executed for a range of message sizes and a number of participating nodes. The average error, the standard deviation and the worst-case error of the communication benchmarks are also shown in Table 2. These results show that simulation of both single-message transmissions and stress-testing communication loads gives proper estimations. Some instances of the *equal-dist* and *unequal-dist* benchmarks produce complex and heavy routing loads, which results in a worst-case error of nearly

33 percent. But considering the excessive communication load that is generated and the abstraction level at which is simulated, this worst-case behaviour is tolerable. From the standard deviation can be seen that simulation of these benchmarks in the general case behaves reasonable.

6 Conclusions

Within the Mermaid project, a frame-work for MIMD architecture modelling and simulation with the purpose of performance evaluation has been developed.

Initial validation of the computational and communication architecture models demonstrated that good simulation results can be obtained. The simulation of computational benchmarks showed average errors up to 5.2 percent and worst-case errors up to 7.9 percent. Results of simulated communication benchmarks demonstrated slightly greater errors due to the higher abstraction level at which is modelled. Nevertheless, the average errors up to 9.1 percent and worst-case errors up to 32.8 percent that were measured for some instances of heavy stress-testing communication loads are reasonable.

Acknowledgments

We would like to thank Marcel Beemster for his comments on draft versions of this article and Alfons Hoekstra for his contributions with respect to the Elastic Light Scattering benchmark. We also want to express our gratitude to Parsytec for their support.

References

[van Brummen94] J. van Brummen, A. D. Pimentel, T. Papathanassiadis, P. M. A. Sloot, and L. O. Hertzberger. MERMAID*: Modelling and Evaluation Research in* MIMD *ArchItecture Design, TR-94-26.* Technical report, Dept. of Computer Systems, University of Amsterdam, 1994.

[Gupta91] R. Gupta. *SPMD Execution of Programs with Dynamic Data Structures on Distributed Memory Machines.* Technical report, Department of Computer Science, University of Pittsburgh, 1991.

[Hennessy90] J. L. Hennessy and D. A. Patterson. *Computer Architecture A Quantitive Approach.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[HPF-Forum93] HPF-Forum. *High Performance Fortran Language Specification, version 1.0 CRPC-TR92225.* Technical report, Center for Research on Parallel Computation, Rice University, Houston, TX, 1992 (revised May 1993).

[Kennedy88] K. Kennedy and D. Callahan. *Compiling Programs for Distributed-Memory Multiprocessors. The Journal of Supercomputing,* vol. 2, pp. 151-169, 1988.

[Langhammer93] F. Langhammer. *The PowerStone Project, Version 1.2.* Technical report, PowerStone Parallelrechner GmbH, Herzogenrath, Germany, 1993.

[Muller93] H. L. Muller. *Simulating computer architectures.* PhD thesis, Dept. of Comp. Sys, Univ. of Amsterdam, 1993.

[Rottger94] M. Rottger, U.-P. Schroeder, and J. Simon. *Virtual Topology Library for PARIX.* Technical report, Paderborn Center for Parallel Computing (PC^2), Department of Mathematics and Computer Science, University of Paderborn, Germany, 1994.