A Realistic Cost Model for the Communication Time in Parallel Programs *

Matthias Fischer† Jochen Rethmann‡ Alf Wachsmann†

†Universität-GH Paderborn
Heinz Nixdorf Institut und
Fachbereich Mathematik-Informatik
D-33095 Paderborn
email: {`mafi,alf`}`@uni-paderborn.de`

‡Heinrich-Heine-Universität Düsseldorf
Institut für Mathematik
Universitätsstr. 1
D-40225 Düsseldorf
email: `rethmann@cs.uni-duesseldorf.de`

Abstract. In this paper we develop a model for communication time on parallel computers consisting of processors and a service network, i.e., a network performing services like broadcast, synchronization, and global variables. The implementation of the service network is done on a free configurable Transputer network.

Our cost model describes the communication time of accesses to global variables and consists of a multi-linear function. The cost model includes the parameters packet size, send hot spot, and the number of processors accessing global variables. These parameters influence the communication time in a high degree and capture important parameters like contention.

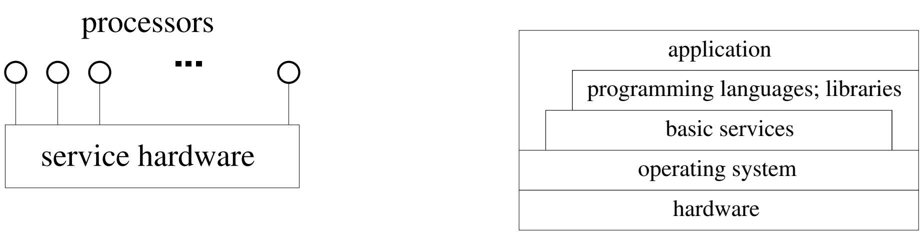
We implement a Bitonic Sort and a Connected Components algorithm (among others) and we show that our model is able to predict the communication time within a 10 % error if indirect service networks are used. The applications show that it is easy for a programmer to determine the parameter values for our model and that our new cost model precisely predicts the communication time of parallel algorithms.

Furthermore, we minimize the communication time of accesses to global variables by finding a balance between the number of messages in the network and their size. Our model predicts the optimal values for these parameters which we validate by experiments. A modified implementation of our routing which determines on-line the optimal parameter values for an access to a global variable achieves good speed ups.

1 Introduction

In this paper we introduce a realistic cost model for communication times of parallel programs. It is well known that the uniform cost model of PRAMs does not apply for todays parallel computers so that many approaches try to overcome this problem. The new models consider blockwise communication, message latency and other effects which can be observed on parallel computers. There are two cost models, the BSP-model by Valiant [3, 2] and the LogP-model by Culler et al. [1] which claim to be more realistic. But it turns out that again some important parameters for the communication time are neglected.

*Supported in part by the DFG-Sonderforschungsbereich 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen."



(a) Hardware view

(b) Software view

Figure 1: Concept of basic services.

The hardware we use is a processor network which is connected via routing facilities. The routing strategies considered are mostly store-and-forward or wormhole routing. On top of this physical network a software layer is built which realizes some useful basic services like virtual shared memory or synchronization (cf. Fig. 1).

The algorithm we use to implement these basic service functions on a store-and-forward Transputer system is a simple greedy shortest paths routing which uses FIFO queues for storing the messages.

The network-based parallel computers can be divided into two classes depending on whether there is a processor and a memory module at each node of the network (direct network) or the processors and memory modules are interconnected by a network of switches (indirect network). We use the term virtual shared memory to refer to a memory that is distributed over distinct memory modules but can be accessed by each processor via routing.

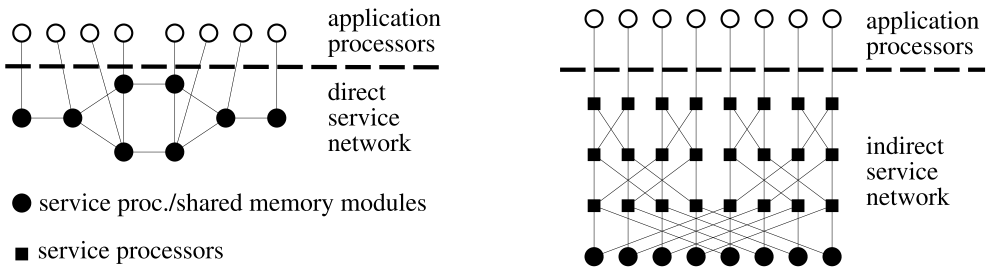
Our intention is to develop a cost model that allows precise predictions of the time needed to access shared memory in direct and indirect network-based machines. The parallel computer we had at our disposal is a Transputer system. Even the Transputer has routing devices on-chip and direct memory access the CPU is involved every time a message has to be forwarded, because buffering of messages and some other work must be done by the CPU. In order to emulate real autonomous routing devices we split the processor network into two parts: one for executing the application program – we call this part application network – and one for simulating the service function – the service network (cf. Fig. 2). Throughout the paper we assume that the number of application processors is equal to the number of shared memory modules. In our indirect network the service processors emulate the routing devices and the shared memory modules.

To be more flexible, our implementation and hardware allows us in contrast to [4], where only direct processor networks are investigated, to use direct service networks, like the Cube-Connected-Cycles network (CCC) or the Shuffle-Exchange network (SE), and indirect service networks, like the Butterfly network (BF).

Making experiments on this system it turns out that neither the BSP- nor the LogP-model can predict the runtime of programs realistically. Both models take an upper bound for the latency for sending messages of a fixed, short size. This assumption does not fit for our intended hardware.

We introduce the hardware we use and describe the implemented software to show that the existing cost models can not be used in our case (Section 2).

We introduce k-k-accesses as basic communication pattern, where a portion r of the application processors are sending/receiving exactly k messages of length l. For this routing pattern we develop a cost function for each service hardware (network topology)



(a) direct service network: Shuffle-Exchange network of dimension three.

(b) indirect service network: Butterfly network of dimension three.

Figure 2: Different possibilities for simulating a service network on a processor network.

which depends on the three parameters k, l and r. The used hard- and software implies a linear dependence of the function on each of these parameters which results in a trilinear function. We validate these considerations by performing measurements on our implemented system where we use the basic service functions to realize the k-k-accesses (Section 3).

To demonstrate the precision of our model we implement several application programs on top of our system (Section 4). In this paper we present two of them (Bitonic Sort and Connected Components). We predict the communication times and compare these with the measured communication times. This comparison shows that our model is well suited for indirect service networks but lacks precision for direct service networks.

As an application of our model we optimize the runtime of the routing we used for realizing the basic service functions (Section 5). For routing there is a trade-off between the number of messages in the network and the lengths of these messages. We speed up the routing by splitting long packets into smaller ones such that the routing becomes more like wormhole routing. We use our cost model to calculate the optimal packet size in order to optimize the routing time. We show that measurements validate the prediction of our model.

References

[1] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subromanian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. *Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 1–11, 1993. Also appears as TR No. UCB/CS/92 713.

[2] A. V. Gerbessiotis and L. G. Valiant. Direct bulk-synchronous parallel algorithms. Technical Report TR-10-92, Aiken Computation Laboratory Harvard University, July 1992.

[3] L. G. Valiant. A Bridging Model for Parallel Computing. *Communications of the Association for Computing Machinery*, 33:103–111, 1990.

[4] A. Wachsmann. *Eine Bibliothek von Basisdiensten für Parallelrechner: Routing, Synchronisation, gemeinsamer Speicher.* Dissertation, Universität-GH Paderborn, 1995.