An Analytical Model for a Class of Architectures under Master-Slave Paradigm

Yasemin Yalçınkaya and Trond Steihaug

University of Bergen
Department of Informatics
Bergen, Norway

Abstract. We build an analytical model for an application utilizing master-slave paradigm. In the model, only three architecture parameters are used: latency, bandwidth and flop rate. Instead of using the vendor supplied or experimentally determined values, these parameters are estimated using the analytical model itself. Experimental results on Cray T3E and SGI Origin 2000 indicate that this simple model can give fair predictions.

While building a performance model, it is crucial to catch the main factors of behavior of the program in question. These factors are the parallelization strategy used, the amount of communication and computation, and the architecture of the parallel computer. The software employed combined with the chosen message passing paradigm plays a significant role in the effective values of the architecture parameters. A promising approach is to build a simplified model for a "real" application on a target architecture. The purpose of this paper is to build an analytical model to predict the behavior of iterative numerical algorithms on a class of architectures using master-slave paradigm. Under the master-slave paradigm, the execution of the parallel program can be seen as a sequence of parallel and purely sequential phases. In the parallel phase the slaves compute concurrently and in the sequential phase only the master does computation. Between these phases there is communication between the master and slaves either in form of single node *broadcast* from the master, or single *send*s and *receive*s between the master and any one of the slaves. We are going to analyze one of many possible parallel programs under this paradigm to show that in certain cases, it is possible to quantify the influence of the factors mentioned above on the performance, and a small number of architecture parameters can be defined to be used in an analytical model. In order to estimate the architecture parameters and their interactions we will make some assumptions.

The iterative algorithm used in the analysis is a block Jacobi algorithm for the solution of linear least squares problems outlined by Dennis and Steihaug [2]. One important aspect of this algorithm is that the amount of communication and computation changes with different values of an algorithmic parameter p [4].

For the performance analysis of the algorithm we need to estimate the run time of one loop and the total number of loops. The run time of one loop, t_{loop}, is estimated by two "components", computation and communication time.

M. Bubak et al. (Eds.): HPCN 2000, LNCS 1823, pp. 601–604, 2000.

When the number of processors at hand is smaller than the number of tasks, we assign more tasks to each slave. Assuming that when a slave finishes sending its result to the master the data is received instantaneously by the master and the slave continues with its next task until it runs out of tasks, t_{loop} is:

$$t_{loop} = \max_{\forall \text{ slaves}} \{ \sum_{i=1}^{k_{slave}} (t_{c\_slave}(i) + t_{s\_send}(i)) \} + t_{c\_master} + t_{broadcast}, \qquad (1)$$

where $t_{c\_slave}(i)$ is the computation time of task i on *slave*, $t_{s\_send}(i)$ is the time to send computation results from *slave* to the master, t_{c_master} is the computation time on the master, $t_{broadcast}$ is the time used by the master to *broadcast* data to the slaves and k_{slave} is the number of tasks assigned to *slave*. The computation time is assumed to be proportional to the number of floating point operations (flops). The communication time is assumed to consist of two parts: a startup time (latency) and a part proportional to the amount of data sent. The proportion factor depends on the number of processors in use and the intercommunication topology.

To analyze the computation time, we count the number of flops, additions and multiplications, for each task. We assume that the time it takes to do a multiplication operation is equal to the time for an addition operation. The values of t_{c_master} and t_{c_slave} are estimated by multiplying the respective flop counts with the inverse of the flop rate of the architecture in use.

In the implementation, MPI is employed as the message passing library. When the slaves begin sending their results, the master is already ready to accept. We assume that the time used by the master to retrieve the arrived data from the buffer is negligible.

Let us define $t_{send} = l + \tau\beta\pi$, where τ is the message size, $\beta = 8/\text{BW}$ is the transfer time per byte, l is latency, and π is a variable depending on the number of hops between the sender and receiver. BW is the bandwidth of the parallel computer architecture and 8 is the number of bytes in one data element. For simplicity, bandwidth is taken as constant in the model. When the number of slaves is only one, we assume that the master and slave processors are neighbors, and $\pi = 1$. On SGI Origin 2000 the interprocessor communication network has hypercube topology and on Cray T3E, each PE has 6 neighbors [4]. The topological structure of the network, hence the number of network links traversed by a message, is critical. When the number of slaves is increased we do not know the topology of the partition assigned to our program but we will assume that the partition is "dense". We can approximate the average distance between any two processors by $1/2 \log_2 n_s$ on Origin 2000, and with $3/4 n_s^{1/3}$ on Cray T3E, where n_s is the number of slaves. Hence, π is $1/2 \log_2 n_s$ and $3/4 n_s^{1/3}$ [1] on Origin 2000 and Cray T3E respectively when $n_s > 1$. Single node *broadcast* operation in MPI is implemented using a binomial tree-structure approach [3]. Therefore, the *broadcast* time is proportional to $\log_2 n_s$.

We use only three parameters in the analytical model: flop rate, latency and bandwidth. However, instead of using the vendor supplied or experimentally

Table 1. Estimated values of architecture parameters

architecture	l	BW	β	α
SGI Origin 2000	22.69 μs	13.38 Mbyte	0.59 μs	7.42 ns
Cray T3E	74.16 μs	16.54 Mbyte	0.48 μs	8.67 ns

determined values, these parameters are estimated using the analytical model itself. The variable π is not considered as a parameter since it is determined by the topology of the communication network. To estimate the values of architecture parameters, we measure the execution time of an actual implementation on increasing problem sizes using one slave. Using only one slave enables us to avoid the effect of other interference from the system. Execution time on different problem sizes reflect the possible change in bandwidth. We take algorithmic parameter p to be zero. The implementation is run for a couple of times for different problem sizes, and the average of execution times for each problem is taken. We calculate the number of flops, count the number of *send*, *receive* and *broadcast* operations and estimate the latency, bandwidth and flop rate of the architectures in question as the solution of a least squares problem.

The architecture parameters estimated to be used in the model are given in Table 1. The parameter α in the table is the time to do one flop. To validate the model we look at the difference between actual execution time measurements and model predicted values. In the figures, increased test problem number means increased problem size [4]. Figures 1 and 2 depict the model validation for execution times on Cray T3E and SGI Origin 2000. We see that on Cray T3E, the predictions are quite accurate, whereas on SGI Origin 2000, when the problem size is increased we are slightly overestimating the execution time. The average deviation between the actual execution times and model predictions is 8.6% of the actual execution times on SGI Origin 2000, and 6.0% on Cray T3E.

Figure 3 displays the predicted versus actual execution time on Cray T3E using two slaves and $p = 1$. When $p = 1$ the amount of communication remains the same as when $p = 0$ but the computation on slaves is increased.

In Fig. 4, we predict the execution time of the application with a different choice of p, $p = Cs$ using 4 slaves on SGI Origin 2000. We see that for the two largest test problems the overestimation observed in the model is magnified. There are two main differences between this case and the model problem: computation on the slaves is increased along with the size of the messages sent by the slaves, and the number of the slaves is quadrupled. These size increases in the system results in increased error in the prediction.

The model can also be used in the analysis of scalability of the application [4]. The estimated parameters do not reflect the architectural characteristics only. The network load, the message passing paradigm used, the program code, the size of the problems are all factors that cause changes in the vendor supplied values for the parameters. This implies that the "estimated" latency is higher than the "machine constant latency". Similarly, the bandwidth is lower [4].

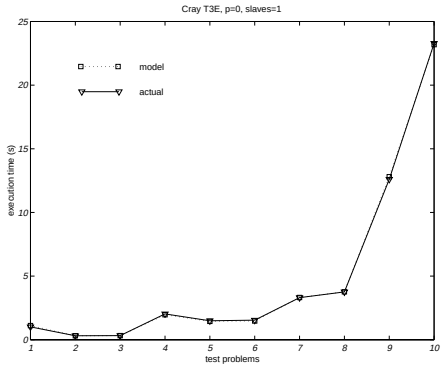

Fig. 1. On Cray T3E, using one slave, $p = 0$

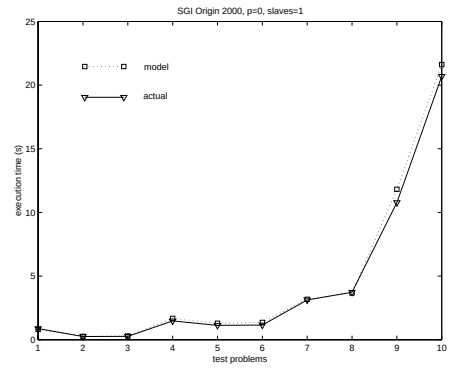

Fig. 2. On SGI Origin 2000, using one slave, $p = 0$

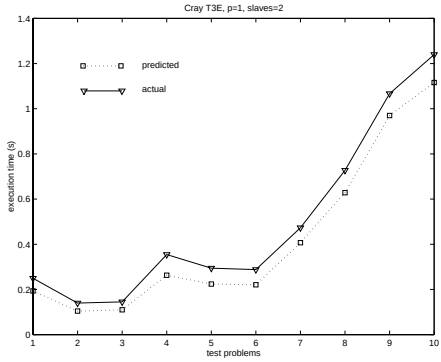

Fig. 3. On Cray T3E, using two slaves, $p = 1$

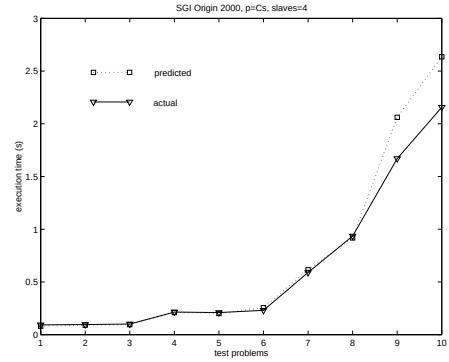

Fig. 4. On SGI Origin 2000, using 4 slaves, $p = Cs$

References

1. Culler, D. E., Karp, R., Patterson, D., Sahay, A., Schauser, K. B., Santos, B., Subramonian, R., von Eicken, T.: LogP: Towards a Realistic Model of Parallel Computation. In: Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. ACM (1993) 1–12
2. Dennis, J. B. Jr., T. Steihaug, T.: A Ferris-Mangasarian Technique Applied to Linear Least Squares Problems. Tech. Rep. No. 150. Dept. of Informatics, Uni. of Bergen, Norway (1998)
3. Touriño, J., Doallo, R.: Performance Evaluation and Modeling of the Fujitsu AP3000 Message-Passing Libraries. In: Amestoy, P. et. al. (eds.): Euro-Par '99. Lecture Notes in Computer Science Vol. 1685. Springer Verlag, Berlin Heidelberg New York (1999) 183–187
4. Yalçınkaya, Y., Steihaug, T.: Are Three Parameters Enough to Represent a Parallel Computer Architecture?. In: Sixth Meeting of the Nordic Section of the Mathematical Programming Society. Proceedings. Opuscula; 49. ISSN 1400-5468, Mälardalen University, Västerås, Sweden (1999)