Analysing the Load Balancing Scheme of a Parallel System on Multiprocessors[*] – A Modeling Approach –

Johann Schumann and Manfred Jobmann

Institut für Informatik, Technische Universität München
email: `jobmann@informatik.tu-muenchen.de`

Abstract. Efficient utilisation and scalability of a parallel system strongly depend on the load distribution and balancing mechanisms. We analyse the behaviour of a load distribution and balancing mechanism by modeling the parallel system as an extended queueing model with subsequent evaluation by the modeling and analysis tool, MAOS. Our focus is on search-based parallel systems (here, the OR-parallel theorem prover PARTHEO), running on a loosely coupled multiprocessor environment.

1 Introduction

For the performance evaluation of a parallel search-based system, there exist a number of important criteria; of most interest for the user is the speedup. For a more detailed evaluation, the following items are to be taken into consideration: the *load-balance* and the resulting utilisation of the processors with respect to the values of the parameters of the load distribution mechanism, and the behaviour of the system wrt. number of processors and the interconnection topology.

In this paper we focus on the second topic, *scalability.* Our approach of performance evaluation of parallel search-based systems using an extended queuing model is illustrated by a concrete example: the theorem prover PARTHEO [5], an OR-parallel theorem prover for first order predicate logic which is realised as a network of sequential theorem provers (SETHEO) communicating via message passing and running on a net of Transputers T800 in a torus-like topology with 4×4 (in general: $m \times m$) processors. Formally, the search for a proof can be depicted by a tree ("OR-tree"). This OR-tree is searched in a depth-first, left-to-right manner with backtracking (for details see [5]). Since the branches of the OR-tree can be executed independently from each other, OR-parallelism is exploited by distributing the proof tasks (i.e. the nodes of the OR-tree) over the network of processors. The execution of a task involves checking if a proof could be found and, in case the task is not a leaf node, generating new tasks. PARTHEO's mechanism for distributing the load is that of "task stealing" (receiver-initiated policy): as soon as a processor runs out of proof-tasks in its local memory ("task store") it asks its direct neighbours in the network for proof tasks. If a neighbouring processor has enough tasks, it sends some tasks (e.g. half of them).

2 The Model

In order to study the behaviour of the parallel system without additional hard-

[*] This work was supported by the Deutsche Forschungsgemeinschaft within the Sonderforschungsbereich 342, Subproject A5.

ware, the approach of modelling the parallel system, based on its actual implementation has been used. The model built is a *queueing network model* extended with explicit process interaction and communication and is evaluated by simulation using the software tool MAOS [2]. The investigations would not be possible within a framework applying pure queueing network models. MAOS's object-based principles of modelling allowed to map the parallel executable proof tasks of PARTHEO directly into tasks of the corresponding MAOS model.

In order to avoid problems wrt. to the notion of speedup for OR-parallelism, for measurements and simulations we use a representative of a (synthetic) class of formulae showing a regular n-ary OR-tree ($n \in \{7, 9, 11, 15, 21, 24\}$), resulting in between 400 proof tasks ($n = 7$) to 14425 tasks ($n = 24$). As the execution time we take the time needed to process all possible proof tasks up to a given depth of the OR-tree[2]. We study the scalability of PARTHEO wrt. the topology of a square torus with $p = m \times m$ processors, and vary m from 2 to 16 resulting in networks of 4 to 256 processors.

The values of all parameters for the PARTHEO model have been obtained from the actual PARTHEO implementation (16$\times$T800). With these values, the simulation model has been validated (cf. [5, 2])[3].

3 Results

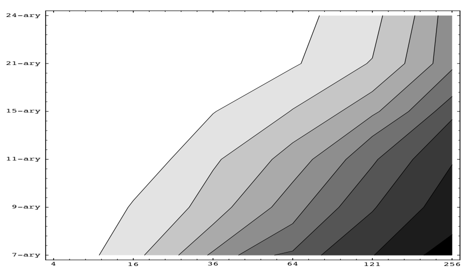
The scalability of the PARTHEO model is analysed by investigating the efficiency η wrt. variations of the number of processors and the problem size. The iso-efficiency curves as shown in Fig. 1A relate the problem size (in our case, directly corresponding to the the number of generated proof-tasks) and the number of processors to the efficiency obtainable (cf. e.g.,[3]). An efficiency of almost 1, being equal to linear speedup, can be obtained in all cases with a small number of processors. When the number of processors increases, the efficiency decreases drastically, as indicated by the steep slope in the iso-efficiency curves. The effect is stronger for smaller problems, although the number of proof-tasks is, even for these problems, large compared to the number of processors. This indicates a severe load imbalance due to the start-up phase and improper migration of proof tasks.

Fig. 1B shows the mean number of active processors for time intervals of one second over the entire execution time. With a comparatively large number of processors, we can also detect several intervals during the run-time in which the mean number of active processors decreases substantially, often to only about 60% or less of the available processors.

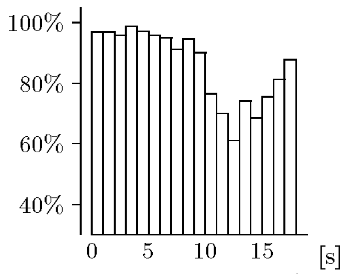
During those times, one expects a good utilisation, since enough proof tasks should be available in the system. However, the vast majority of proof-tasks are tasks at the leaf nodes of the OR-tree which do not generate new tasks and have long execution times. This leads to the conclusion that a task-stealing mechanism should take the type of proof task into account. Experiments with variations of the appropriate parameters, however, yielded only little variations in the overall

[2] This corresponds to the situation of searching for all proofs of a formula.

[3] For each different OR-tree and number of processors, 10 independent simulation runs have been made from which mean values and confidence intervals (at level 0.9) have been calculated. There is only little fluctuation. Hence, no confidence intervals are shown.



A) Iso-efficiency curves for 4 to 256 processors and 7-ary to 24-ary OR-trees. Solid lines represent efficiency values of 0.9, 0.8, . . . , 0.1 (from left to right).



B) Percentage of mean number of active processors in time intervals of 1 second as a function of time (121 processors, 24-ary OR-tree)

Fig. 1. Iso-efficiency curves (A) and processor utilisation (B)

behaviour. Therefore, we focus on ways to increase the balance of load during the start-up phase of PARTHEO. Starting with one proof task on a single processor is a simple and convenient scheme for implementation, but as the experiments revealed, results in a severe load-imbalance.

In the following experiment we use a different scheme of start-up. Instead of starting with only one proof task, we first generate n proof tasks (for an n-ary OR-tree) on one processor. In most cases, the number of processors is larger than the number of initial proof tasks. Then, we place proof tasks onto processors (one per processor) which roughly have the same distance to each other in the network before activating the task-stealing mechanism as usual.

Fig. 2 shows the efficiency curves for this experiment revealing a much better efficiency, especially in the case of large networks. This last task distribution scheme

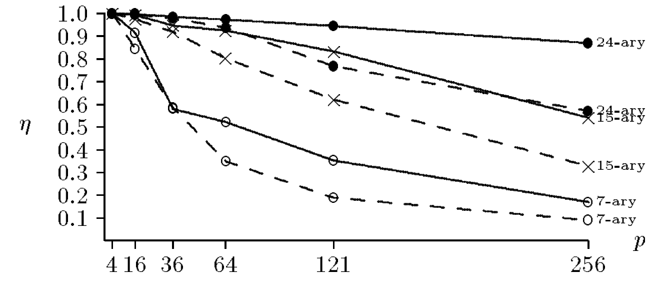


Fig. 2. Efficiency values for different numbers of processors p. Dotted lines are for the original start-up mechanism, solid lines for the modified one.

may be characterised as a two-level distribution of load as described in [1]: tasks in the first level are created at one (master-)processor and attracted by designated (master-)processors at the second level, which again create (sub-)tasks according to a specified tree depth. But there is a significant difference: in our scheme there doesn't exist a particular fixed group partition of the processors, but all processors (initially starting with the neighbours of the designated processors) may attain tasks from each other. Therefore our group partitioning may be viewed as a dynamic one depending on the arity of the OR-tree and without fixed group borders. This scheme is easier to implement, e.g., there is no need for group merging in order to balance load across groups. On the other hand, we cannot adapt this simpler scheme to more than two levels of task creation, which would probably be needed for a larger number of processors and a relative small number of terminal tasks. But, if the number of terminal tasks is large compared to the number of processors a two level scheme will suffice, since we could choose an appropriate tree depth to create enough tasks in the first level to utilise enough designated (master-)processors in the second level.

4 Conclusions

Starting from a specification of the parallel system and its implementation, an extended queueing network model was developed, modeling the communication principles and the mechanism for distributing work. Details of the system which are not of interest for evaluating the performance (e.g., sequential calculations), are hidden by abstraction.

This model was used to analyse the scalability of the parallel system for the OR-parallel theorem prover PARTHEO. Although an almost linear speedup can be obtained with a comparatively small number of processors, the efficiency decreases substantially with an increasing number of processors, due to a strong load-imbalance. A closer look revealed that a major reason for this imbalance can be found in the start-up of the system, since the system starts with one proof task on one designated processor.

A comparatively small change in this start-up behaviour – instead of one proof task, we initially distribute n tasks (for an n-ary OR-tree) – resulted in a substantial increase of efficiency. This result is worth to be a basis for future experiments, where the number of created tasks – or the corresponding tree depth – in the initially designated processor may be adapted to the number of processors and to the needs of a good first level distribution of these initially created tasks.

References

[1] M. Furuichi, K. Taki, and N. Ichiyoshi. A multi-level load balancing scheme for or-parallel exhaustive search programs on the multi-psi. In *Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 1990.

[2] M. R. Jobmann. *Leistungsanalyse von Rechen- und Kommunikationssystemen – Konzepte der Modellauswertung und Definition einer Modellierungssprache.* PhD thesis, Universität Hamburg, Hamburg, Feb 1991.

[3] V. Kumar and A. Gupta. Analyzing Scalability of parallel Algorithms and Architectures. Technical Report TR-91-18, Univerity of Minnesota, June 1991.

[4] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning*, 8(2):183–212, 1992.

[5] J. Schumann and R. Letz. PARTHEO: a High Performance Parallel Theorem Prover. In *CADE10.* Springer, 1990.

This article was processed using the LaTeX macro package with LLNCS style