Distributed HW/SW-Partitioning for Embedded Reconfigurable Networks *

Thilo Streichert, Christian Haubelt, Jürgen Teich
University of Erlangen-Nuremberg, Germany
{streichert,haubelt,teich}@cs.fau.de

Abstract

In this paper, we propose a distributed online HW/SW-partitioning strategy for increasing fault tolerance in HW/SW-reconfigurable networked systems.

1. Introduction

Distributed and adaptive embedded hardware platforms are becoming more and more important for applications in the area of automotive, body area networks, ambient intelligence, etc. In order to be able to cope with different time-variant application demands or node defects, these systems must be adaptive and be able to react to unforeseen changing requirements. Due to the possibilities provided by the currently available technology, functionality implemented in hardware (HW) or software (SW) can be dynamically assigned to resources in the network.

We therefore propose a distributed two-step strategy for online HW/SW-partitioning, that consists of a HW/SW bi-partitioning heuristic and a dynamic load balancing algorithm.

All related work discusses either offline HW/SW partitioning algorithms [5] or load balancing approaches only but do not consider the online HW/SW partitioning problem. One publication [4] in this field describes load balancing on a platform consisting of a microprocessor and reconfigurable HW, but without an extension for networked systems.

2. Problem Definition

There are two possible scenarios where a (re)partitioning of processes in a reconfigurable network becomes important or even necessary: The first is the possible failure of a computational node at a certain time. The second is for optimality reasons in case of changes of the computational demands of the running application due to either finishing tasks or yet unknown arriving tasks.

The HW/SW partitioning problem is defined as an assignment of each task to a resource as well as an indication whether the task is implemented in HW or SW. Here, we propose a distributed algorithm which solves this problem at runtime, leading to an online HW/SW partitioning problem. In the following, we will call the set of active resources at time t the *allocation* $\alpha(t)$ and the assignment of tasks to resources the *binding* $\beta(t)$.

Our approach to online HW/SW partitioning tries to find a binding that optimizes the current binding $\beta(t)$ with respect to the following objectives:

1. load balance in the network: With this objective to be minimized, the load in the network is balanced between the nodes, whereas HW- and SW-tasks are treated separately: $max(max(w_i^S) - min(w_i^S), max(w_i^H) - min(w_i^H)) \forall i = 1..|N|$ with w_i^S being the SW load on node i and w_i^H being the HW load on node i. $|N|$ is the number of resource nodes.

2. hardware/software load balance: The load between the HW- and SW-resources has to be balanced, i.e., we minimize $\left| \sum_{i=1}^{|N|} w_i^{\mathrm{S}} - \sum_{i=1}^{|N|} w_i^{\mathrm{H}} \right|$.

3. minimization of total load: To avoid unnecessary high computational load in the network, we minimize $\sum_{i=1}^{|N|} w_i^S + w_i^H$.

With these objectives, our online HW/SW-partitioning algorithm will then create most likely task assignments that enable a good *load reserve* on each active node which is important for achieving fast repair times in case of unknown future node failures. In this paper, we assume constant workload demands on each node, i.e., $w^S(p) = w_i^S(p) \forall i = 1..|N|$ for a given process p.

The SW workload $w_i^S(p)$ on node n_i of process p is the fraction of execution time to its period and the HW workload $w_i^H(p)$ is defined as the fraction of required area and maximal available area, resp. configurable logic elements in case of FPGA implementations.

3 Online Hardware/Software Partitioning

In figure 1 we present our methodology for handling unexpected events. The goal of the fast repair phase is to reestablish the functionality of the network. This part is not covered in this paper. The second phase, the online HW/SW partitioning tries to find an optimal binding of processes to nodes so to optimize the above three objectives. This is achieved by an online HW/SW partitioning strategy consisting of a load balancing and a bi-partitioning phase. While the load balancing step tries to find good solutions with respect to the first objective, the local bi-partitioning tries to optimize the second and third objective. Note that this partitioning will run in a distributed and decentralized manner in the network for reasons of fault tolerance.

3.1 Load Balancing

For the load balancing, we apply a diffusion based algorithm which moves load entities along the links in the network to other nodes. Characteristic to a diffusion-based algorithm, introduced first by Cybenko [3], is that iteratively,

*Supported in part by the German Science Foundation (DFG) under contract TE 163/10-1, SPP 1148 (Rekonfigurierbare Rechensysteme)

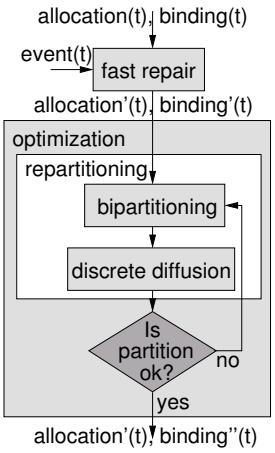

Figure 1. Phases of HW/SW partitioning

each node is allowed to move any size of load to each of its neighbors. Communication is only allowed along edges in the network. Although diffusion algorithms have received a considerable amount of attention throughout the last couple of years, see, e.g., [1], they have two negative characteristica: They work with real-valued portions of load and they have alternating behaviour. Therefore, we proposed a diffusion scheme that migrates only full tasks between nodes. For this discrete diffusion algorithm we have proven that it does not exceed optimality constraints concerning the amount of migrated tasks in the continuous case and we are able to show theoretically maximal deviations with respect to the quality of the load balance, cf. [2].

3.2 Local Bi-Partitioning

The bi-partitioning algorithm first determines the load ratio between a HW and a SW implementation for each process: $w^H(p_i)/w^S(p_i)$. According to this ratio, the algorithm selects one task and implements it either in HW or SW. Due to such a local strategy, we can guarantee that the total load will be minimized, but to reach an optimal HW/SW balance, we calculate the total SW load and the total HW load on each node. If the total HW load is less than the total SW load, the algorithm selects a task which will be implemented in HW, and the other way round.

Due to these competing objectives, tasks with a ratio larger than one can be assigned to hardware and tasks with a ratio less than one are assigned to software. In order to undo this suboptimal task assignment, we migrate these tasks at first in the diffusion step.

The run time complexity of this algorithm is $\mathcal{O}(|P_j|log_2(|P_i|+|P_j|))$, where $|P_i|$ is the number of processes bound onto node n_i and $|P_j|$ is the number of new processes on the same node.

4 Experimental Evaluation

In the following evaluation, we are comparing the results obtained with our distributed approach with the results of a methodology which is based on an Evolutionary Algorithm

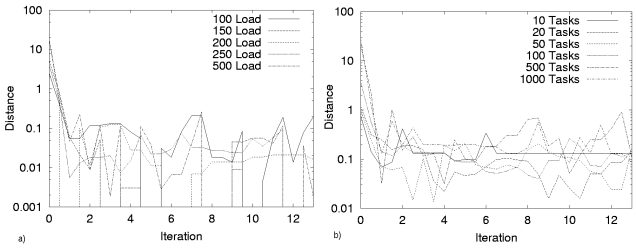

Figure 2. Shown is the distance d(s) to a reference solution over the iteration of the HW/SW-partitioner. a) Optimal partitioned tasks with a certain load on a node. b) Randomly distributed tasks in a network

(EA) and possesses global knowledge, i.e., a centralized optimization strategy is used. Using the EA, we determine a set R of reference solutions and calculate the shortest normalized distance $d(s)$ from the solution s found by the online algorithm to any reference solution $r \in R$.

In the first experiment, we are starting from a network which is in an optimal state, such that all tasks are implemented optimally according to all objectives. Now, we assume that new SW-tasks arrive on one node. Starting from this state, Figure 2a) shows how the algorithm performs for different load values. In the second experiment, the initial binding of tasks and load sizes were determined randomly. For this case, which is comparable to an initialization phase of a network, we generated process sets with 10 to 1000 processes, see Figure 2b). In this figure, we can clearly see that the algorithm improves the distribution of tasks already with the first iteration leading to the best improvement. Note that Figure 2 presents the distance after each bi-partition and diffusion step. We can see in Figure 2 that the failure of one node causes a high normalized error. Interestingly, the algorithm finds global optima but due to local information our online algorithm cannot decide when it finds a global optimum.

5 Conclusions and Future Work

In this paper, we have presented a first approach to online HW/SW partitioning in networked embedded systems. The proposed two-level algorithm increases the load reserves on each node in order to compensate node defects.

In the future work, we will try to find a sufficient termination criteria and will investigate real-time capabilities.

References

[1] R. Elsässer and A. Frommer and B. Monien and R. Preis, "Optimal and Alternating-Direction Loadbalancing Schemes," *Proc. of Euro-Par 99, Parallel Processing*, 1999

[2] T. Streichert and Ch. Haubelt and J. Teich, "Online HW/SW-Partitioning in Networked Embedded Systems," *Proc. of ASP-DAC 05*, 2005

[3] G. Cybenko, "Dynamic Load Balancing for Distributed Memory Multiprocessors," *Jour. of Parallel and Distributed Computing*, 1989

[4] R. Lysecky and F. Vahid, "A Configurable Logic Architecture for Dynamic Hardware/Software Partitioning", *Proc. of DATE'04*, 2004

[5] M. López-Vallejo and J. C. López, "On the Hardware-Software Partitioning Problem: System Modeling and Partitioning Techniques", *Trans. on Design Automation of Electronic Systems*, 2003