HARDWARE/SOFTWARE PARTITIONING
Research abstract

Zoltán Ádám Mann, András Orbán

In embedded systems, most functionality can be implemented either by special-purpose hardware or by a software program running on a general-purpose processor. Hardware/software partitioning is concerned with deciding which functions should be implemented in hardware and which ones in software. It aims at finding an optimal trade-off between conflicting requirements on performance, power, costs, etc. Therefore, partitioning is a tough but very important problem.

The main characteristics of our partitioning model are the following. We consider one software context (i.e. one general-purpose processor) and one hardware context (e.g. one FPGA). Software implementation of a component is associated with a *software cost*, which may be e.g. the running time of the component if implemented in software. Hardware implementation of a component is associated with a *hardware cost*. If two communicating components are mapped to different contexts, this incurs a *communication cost*. If two components are mapped to the same context, then the communication between them is neglected.

The software cost of a partition is the sum of the software costs of the components that are mapped to software. Similarly, its hardware cost is the sum of the hardware costs of the components that are mapped to hardware. Its communication cost is the sum of the communication costs between those components that are mapped to different contexts. While this model is simplified and constrained in several aspects, it captures the essence of hardware/software partitioning, which lies in the simultaneous optimization of several conflicting cost factors.

We have defined a formal graph-theoretic model for this problem, in which the behavioral components of the system are modeled with the vertices, and their communication with the edges of a graph, and proven its NP-completeness in the general case. We also managed to identify some efficiently solvable special cases. In particular, if communication is the only significant cost factor, then the problem can be solved optimally in polynomial time. On the other hand, if communication can be neglected, then the problem is still NP-complete, but pseudo-polynomial and approximation algorithms are known for this special case.

We have developed several algorithms for solving the partitioning problem. We formulated the partitioning problem as an integer linear program (ILP) which can be solved by standard ILP solvers. This method yields the exact optimum, and works in acceptable time for graphs with up to 300 nodes. We also developed an algorithm based on branch-and-bound that can solve the partitioning problem optimally, and is usually faster than the ILP-based algorithm.

In order to partition even more complex systems, i.e. graphs with several thousands of nodes, we also developed a number of heuristic algorithms. In particular, we developed a genetic algorithm (GA) which turned out to be effective for very large graphs as well. Another approach is based on clustering: the vertices of the original graph are clustered, so that the number of vertices is reduced until the resulting graph is small enough to be solved by the exact methods described above.

We also considered a slightly modified version of the partitioning problem, in which the total cost of a partition is defined as the *weighted sum* of the corresponding hardware cost, software cost, and communication cost. We presented a *polynomial-time exact algorithm* for this problem. This algorithm also forms the basis for another heuristic algorithm for the original partitioning problem, which works by running the polynomial-time algorithm for several different weights and selecting the best partition satisfying the constraints.

One of the most important advantages of our partitioning model is that it can also accommodate *data objects*. Most previous approaches to hardware/software partitioning considered only functional objects; however, data objects (i.e. variables) also have to be mapped to either hardware or software. What is more, their mapping also heavily affects how the functional objects should be partitioned and vice versa. For this reason, our approach handles data objects and functional objects together in a combined optimization problem. The same partitioning algorithms can be used for this extended problem as well, and thus we can find partitions of much higher quality with significantly more realistic costs.

The applicability of the whole partitioning methodology depends heavily on how the necessary cost values (hardware costs and software costs of the nodes and communication costs of the edges) can be obtained from a given system specification. For this reason we developed a partitioning framework, which takes as input a C program, and automatically creates the corresponding graph on which partitioning is performed. Static and dynamic analysis is applied to the C program in order to obtain the necessary cost values. In particular, we use a proprietary memory profiling method to obtain the cost of communication between functional and data objects. The whole process is automated by our partitioning tool.