Template Mini-Project Proposal

Robert Dick

This document gives an example of the project proposal structure. The topic was taken from an actual project in a previous related, but not identical, course. Although the structure is appropriate for IDES, the topic is not ideal for IDES.

1 Goal

Enable designers of embedded systems to automatically extract execution and communication graphs from software implementation for use in developing high-performance application-specific hardware.

2 Formal Problem Statement

Given an application written in ANSI-C, automatically extract a graph in which nodes represent computation within a thread for a particular duration and edges indicate the quantities of data communicated among tasks. The use of dynamic graph extraction has the benefit of permitting internal loop iterations and conditional evaluations to approximate those expected in normal use, but it does imply that the input data set(s) used for execution during extraction are important.

We plan to instrument the Simics [1] full-system simulator to capture data flow among threads, which was recently purchased by Intel. It will be necessary to identify context switches, which we will achieve by extending the simulator with operating system introspection capabilities based on specific program counter values.

3 Most Closely Related Work

There have been projects on automatically generating graphs for debugging and evaluation of synthesis algorithms [2]. Such synthetic graphs are not derived from real applications and therefore have major drawbacks for use in validating design tools and cannot be used to design hardware accelerators for real applications.

Compilers typically build instruction-level data dependency graphs [3] of abstraction is far below the proposed task-level graph and this analysis is static.

Manually constructed graph-based system specifications have been published in the past [4, 5]. These specifications required painstaking manual construction or extraction.

4 Schedule

Checkpoint 1: Install Simics, get it simulating multithreaded applications under a Linux operating system image.

Checkpoint 2: Develop a Simics extension to identify context switches, and modify memory reads and writes to allow tracking of data flow among threads.

End of course: Develop a graphical display utility to simplify graph browsing and to analyze multiple multimedia applications.

References

[1] P. S. Magnusson, F. Dahlgren, H. Grahn, M. Karlsson, F. Larsson, F. Lundholm, A. Moestedt, J. Nilsson, P. Stenström, and B. Werner, "SimICS/sun4m: A virtual workstation," in *Proc. USENIX Conf.*, June 1998.

[2] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: task graphs for free," in *Proc. Int. Wkshp. Hardware/Software Co-Design*, Mar. 1998, pp. 97–101.

[3] K. S. Vallerio and N. K. Jha, "Task graph transformation to aid system synthesis," in *Proc. Int. Conf. on Circuits & Systems*, May 2002, pp. 695–698.

[4] S. Prakash and A. Parker, "SOS: Synthesis of application-specific heterogeneous multiprocessor systems," *J. Parallel & Distributed Computing*, vol. 16, pp. 338–351, Dec. 1992.

[5] J. Hou and W. Wolf, "Process partitioning for distributed embedded systems," in *Proc. Int. Wkshp. Hardware/Software Co-Design*, Mar. 1996, pp. 70–76.