doi:10.1093/comjnl/bxl005

Capsule Reviews

Hardware-Oriented Algorithms for Rendering Order-Independent Transparency. M. AMOR, M. BÓO, E. J. PADRÓN AND D. BARTZ

This paper starts from the goal of increasing visual realism in scenes with high depth complexity. This goal requires sorting algorithms that render transparent objects properly into a scene (after having rendered opaque objects). In real time applications, the available hardware resources in graphics cards make the correct sorting a difficult architectural and scheduling problem and hence existing algorithms seem to suffer from some drawbacks. This paper proposes two new algorithms that reduce the time requirements of pervious proposals while at the same time, only requiring minor modifications to the general structure of current graphics cards. The first proposal is a hardware-oriented algorithm based on weight factor computations and processes the transparencies without requiring their previous sorting. This algorithm only requires a comparator, two adders and a multiplicator and can be performed in any order (unlike earlier algorithms which used back-to-front sorted order). Moreover, this algorithm is sequential and is directly associated to the reduced number of hardware resources available. The second proposal is a hardware-oriented algorithm based on a pre-sorting stage (based on the transitivity of the distance relation), which aids the reduction of the time requirements of previous proposals. Both algorithms are evaluated in terms of the time and hardware requirements. It is shown that both algorithms reduce the time requirements per pixel with respect to previous proposals at the cost of including an organized storage scheme in contrast with the traditional random storage. Moreover, both algorithms reduce the memory bandwidth requirements of previous hardware proposals.

Instruction-Level Parallelism through Microthreading—A Scalable Approach to Chip Multiprocessors. K. BOUSIAS, N. HASASNEH AND C. JESSHOPE

Chip multiprocessors (CMPs) aid to achieve major increases in the computational power of computers. Although CMPs scale well, programming them without explicit use of concurrency remains a major problem. Moreover, in the presence of a number of processors used in a CMP, it remains difficult to split the code into a number of independent threads, to schedule these on many processors and to do this with a low and scalable overhead. This paper addresses this problem by aiming to define a feasible architecture for a scalable CMP which minimizes the communication and synchronization overheads between different threads. The presented solution in the paper is based on the decomposition of a sequential program into small fragments of code called microthreads which are scheduled dynamically and which can communicate and synchronize together quite efficiently. Furthermore, the paper presents an analysis of microthreaded register file ports in terms of the average number of accesses to each port of the register file which illustrates some advantages of the method. Moreover, preliminary simulation results are given which again establish some advantages related to scalability.

Using Control Dependencies for Space-Aware Bytecode Verification. C. BERNARDESCHI, G. LETTIERI, L. MARTINI AND P. MASCI

The Java Virtual Machine checks security using the bytecode verifier which helps ensure that bytecode execution cannot compromise the memory. A Java card is a smart card which runs on a Java Card Virtual Machine (JCVM). Java cards present the challenge of requiring high security features while only having limited resources. Major challenges arise because of the large memory space requirements of the verification process which make it difficult to embed the implementation of a bytecode verifier in the JCVM. As a matter of fact, on-card bytecode verification requires special optimizations since cards have limited memory resources and furthermore, bytecode verification is expensive in space. Even the dictionary size of not very complex methods is too large to fit in the RAM available on Java cards. This paper is concerned with this on-card verification process. The paper proposes a verification algorithm which has the potential of saving space through the optimization of the use of memory by associating a lifetime to the data structures. The algorithm is given in two variants (an If-based algorithm which saves execution states of conditional instructions and a Target-based algorithm which saves execution states of jump targets). The two algorithms are compared and are used in experimental settings related to developing prototype verifiers in the so-called Bytecode Engineering Library. These experimental results are most encouraging.