Compressing Address Trace Data for Cache Simulations

Alexander Fox, Thomas Grün (afox, gruen@cs.uni-sb.de)
Fachbereich 14 Informatik, Universität des Saarlandes,
Postfach 151150, Geb. 45, Zi. 3.20, 66041 Saarbrücken

Computer architects who investigate memory systems try to identify the best cache architecture by simulating different cache variants. They usually feed prerecorded memory reference patterns or address traces into a cache simulator and compare the different cache architectures. The address traces have been extracted from a variety of workloads and they consist of tens of millions of memory references each. Thus, they require plenty of disk space and I/O time when being processed. There has been little research on how to compress these data efficiently. The most common format is DINERO format which stores the reference type (fetch, store, load) and memory address in plain ASCII format. Generic compression programs like compress or gzip yield a compression factor of 3 to 7. Up to now, the best format is PDATS which exploits the fact, that most instructions are fetched from successive addresses. It uses a sort of run length encoding and achieves a compression factor of 5 to 12. Applying gzip to the output results in another factor of 5 to 30.

Our new approach of storing address traces, the RPS format (recovered program structure), is based on two ideas: first, the structure of the underlying program is reconstructed from the address trace, and second, the output is decomposed in multiple files such that gzip can take advantage of repeated input patterns. In a first step, the control flow of the program is determined by identifying the basic blocks, i.e., the straight segments of code with no jumps in and out. Then, the invocation sequence of basic blocks is written to a file, which can be compressed by a factor of more than 35, since gzip can easily detect patterns in it. The basic block data contains information on the length of a basic block and on the position of load and store instructions among all instructions. Their addresses are stored in separate files. In a second step, the load and store references are partitioned in global, local and unassigned variable classes. Global variables have the same value for all invocations of a basic block, local variables can be represented as base + offset, where offset is a constant and base only changes between invocations of a basic block. All other variables are "unassigned" and their addresses are stored in separate files as a difference to the previous value.

The 5 files composing the RPS format contain all information needed to reconstruct the original address trace. On large traces, RPS compresses better by a factor of 4 than the gzipped PDATS format, which was the best format up to now. Detailed information on this work is available on http://www-csi.cs.uni-sb.de/~tgr/atrace.