Summary Part 1
The Performance of Runtime Data Cache Prefetching in a Dynamic Optimization System

Mrinal Nath (ID: 3307043)
February 28, 2005

What are the problems solved by this paper?

Traditional software controlled data prefetching is very effective for certain applications like dense matrix oriented numerical applications. But for other applications that include indirect memory references, complicated data structures and recursive data structures, the performance of the software technique is limited. This is because runtime information about cache misses and the miss addresses is not available to the compiler.

Also, various parameters affect the compilers ability to insert useful prefetch instructions. For e.g. program structure, runtime memory behavior, and microarchitectural features all affect the usefulness of the prefetch instructions inserted by the compiler.

In this paper, dynamic optimization techniques are being used to generate runtime prefetch optimizations (by inserting instructions).

What are the approaches attempted by this paper?

Runtime profiling is used to collect information about the various phases of the program. Once a stable phase has been detected, traces are built by the trace selector. Then control is transferred to the dynamic optimizer (which runs as a separate thread), which optimizes these traces (based on data reference pattern information). The optimized trace is written into the trace pool for use in future repetitions of the trace. The prefetch code inserted itself can be optimized and it has to be scheduled appropriately so that it performs useful fetches. (It should not pollute the cache unnecessarily and should fetch the data just in time.)

Further, the profiling results can be fed back to the compiler so that it avoids prefetching data that has good cache hit rates.

What are the main conclusions of this paper?

The technique can produce good speedup in some applications (those in which prefetching is beneficial). In the applications where prefetching is not of much use, the overheads associated with the technique are negligible, causing only up to 2% performance degradation. The authors examined the traces and found that for these applications, the dynamic optimizer detects the correct delinquent loads. But the technique does not help in improving the performance of the applications due to various reasons. (1) The cache misses may be evenly distributed and so instruction scheduling itself is able to hide the latency of these misses. (2) Some delinquent loads have complex address calculations, due to which the stride could not be computed accurately. (3) Sometimes the optimizer does not insert prefetchs early enough to avoid the cache miss.