

Aaron Kobayashi

9/10/2002

Performance Limits of Trace Caches

Assumptions:

There are a few basic assumptions that affect the plausibility of this paper. The experimental results in this paper are based on heavily on the results yielded from sim-outorder simulator created by SimpleScalar tools. While it is likely that the simulator has been written to exactly model a real world implementation of a trace cache, software bugs do exist and they could affect the test results. The authors also mention that the simulator was modified in some way. These modifications could unknowingly skew results or introduce flaws in the simulation algorithms.

Summary:

Trace caching is important to the future of microarchitecture. It is an idea that promises great results over the more traditional I-Cache method. While trace caching has many possible benefits, it is both appropriate and necessary to examine the limits of trace caches and how they perform relative to the ideal case.

In order to understand the benefits (and limitations) of trace caches, we first need to define some metrics that we can test our ideas against. These metrics are hit rate, IPC, fragmentation, duplication, indexability, and retirement rate.

The hit rate and IPC are the most common metrics and can be applied to both I caches and trace caches. The hit rate measures the effectiveness of a cache in providing instructions to the front end of the processor. If a processor has a high hit rate, it is more effective because it does not have to wait as long to go get instructions from main memory. The IPC or instructions per cycle is the ultimate measure of a processor as it allows us to see how much work a processor is doing. IPC can be greatly affected by the efficiency of other processor systems such as the fetch and cache sub-systems.

Fragmentation gives us an idea of how efficiently the trace cache stores instructions by indicating the amount of unused space in the cache. A trace cache defines an amount of space for each trace (typically 16 instructions). When a trace is shorter than 16 instructions due to a trace terminating instruction such as a branching instruction, the extra space allocated for that trace is lost and this unused space increases the fragmentation of the cache.

Duplication is a measure of how efficiently the unfragmented cache is utilized. Duplication is caused by the fact that two or more different traces may contain the same physical instructions. Each trace stores those instructions and consequently the cache stores is not fully utilized due to duplication.

Efficiency is essentially a convenient way for us to talk about a cache trace's combined fragmentation and duplication. Efficiency indicates the amount of the total cache space that we are utilizing for unique instructions. It is interesting to note that a traditional I-cache design always has an efficiency of 100% because fragmentation and duplication are inherent in the design of trace caches but do not exist in I-Caches.

Indexability is the rate at which a cache miss is performed because the trace starting address is not in the trace cache at all, including interior blocks. In this paper, indexability is presented as a limit because it is not possible to look at the entire contents of the trace cache simultaneously. The indexability metric allows us to measure and compare proper trace-cache indexing algorithms.

Retirement Rate is the ratio of the number of instructions fetched into the pipeline to the number retired. This rate gives us a metric we can use to measure the effects of using a trace cache on the rest of the processor resources. The retirement rate is affected by the branch predictions accuracy, pipeline depth, and the processor issue width.

Limitations, Weaknesses, and Fallacies:

This paper was slightly limited in scope. The stated purpose of this paper was to “study the limits of trace cache performance and from where the limits arise.” Later in the paper, they describe the upper bound of trace cache performance as the being equivalent to an idealized three-ported instruction cache with a perfect alignment network, however they do not elaborate on this in any way. In order to strengthen the paper, a greater explanation of this mechanism would be helpful. This problem is exacerbated by the results of the compress and go benchmarks which seem to exceed their performance bounds. These results raise questions about the validity of both the simulation environment and the proposed performance bounds of a trace cache.

Experimental Results:

The single most important result of this research is the discovery that the branch predictor accuracy heavily impacts the IPC of a trace cache based processor. As branch prediction accuracy increases, the IPC could increase by on average 60% or more! This was shown to be the factor that most limits the processor and therefore has the greatest potential gains.

Even though the branch prediction can be greatly improved, the paper further shows that as a predictor approaches perfect prediction, the trace cache continues to fall short of the ideal case by more than 20%. Fragmentation duplication, and indexability seem to be the culprits. It was shown that duplication of instructions increases dramatically as the size of the trace cache is increased. Simultaneously the fragmentation increases with larger trace caches due to the limited number of starting addresses available to utilize the trace cache. These two factors combine result in a loss of 20%-40% loss in capacity of the cache.

Finally it was shown that the trace cache hit rate ranges from 60%-90%, but if a perfectly indexable trace cache is used, that rate increase to 95%-99%. These results suggest that the main reason a trace cache suffers a high miss rate is because of simplistic cache line allocation policies.

References:

- M. Postiff, G. Tyson, and T. Mudge, "Performance Limits of Trace Caches," Journal of Instruction-Level Parallelism, Vol. 1 (October 1999).