

Using Prediction to Accelerate Coherence Protocols

Shubhendu S. Mukherjee and Mark D. Hill

Computer Sciences Department, University of Wisconsin-Madison

1210 West Dayton Street, Madison WI 53706-1685, USA

{shubu,markhill}@cs.wisc.edu

URL: <http://www.cs.wisc.edu/~{shubu,markhill}>

Presenter: **Naresh Sukumar**

Wednesday 11/5/2008

Summary: **Nicolas Tjioe**

Most large shared memory multiprocessors use directory protocols to support their cache coherence protocol. One of the drawbacks of directory protocol is long latency to access the remote cache blocks. This paper describes the new approach with general prediction logic that monitors coherence activity and triggers appropriate coherence action.

Cosmos is a two level adaptive predictor in which the first level table is called as Message History Table (MHT) and the second level table is called as Pattern History Table (PHT). This paper accelerates coherence protocols by developing and evaluating the Cosmos coherence message predictor. The various strengths and weaknesses of the paper are described below:

Strengths of the paper:

- Dynamic pattern instead of static pattern –cosmos can discover and track application-specific patterns not known a priori. This remains the chief contribution of the paper given that a general predictor overcome the dependence of predictor on application type making it adapt to new application patterns.
- The prediction is immune to order of arrival of messages and has multiple predictions to choose from compared to PAp.
- Cosmos tries to separates the predictor from the protocol which, if implemented correctly will enable predictor to be designed and evaluated fairly independent of coherence protocol
- The mechanisms presented might have applications to uniprocessor cache designs. Perhaps the designs could be applied to predict cache operations for a single high-performance complex non-CMP processor.
- The prediction mechanisms are based on a directory protocol design and thus might be more useful in a large CMP design where a directory protocol is necessary to overcome the inter-core wire delays that might limit the scale of a mechanism like “Cooperative Caching”.

- This novel application of branch prediction to cache coherency and its operations produces obvious speedup enhancements based on the given simulation and benchmark results.
- Unlike other optimizations using prediction, the proposed prediction logic is separate from standard coherence logic. This makes it easier to debug. If the logic is integrated with the protocol, it may undergo state explosion.
- The predictor can discover and track application specific patterns and dynamically adapt to the conditions. This gives improved performance in all applications. This feature is not present in other implementations of prediction logic.

Weakness of the paper:

- Since the paper deals with coherence protocol message prediction in isolation as opposed to integrating it into a coherence protocol, it may be making certain simplifying assumptions. While the justification for the same has been given in terms of readiness of tool and implementation concerns that might obscure initial studies, the efficacy of the scheme in its entirety compared to existing static pattern predictors cannot be fully understood.
- While the paper repeatedly dwells on the accuracy of prediction, the discussion in section 4.4 on the performance benefits in the form of speedup are not adequate. While the paper mentions that its motive is not to investigate impact on runtime, this would be a vital factor to establish the success of cosmos irrespective of its high prediction accuracy.
- While cosmos is an attempt at designing a general predictor scheme, its reliance on the slow changing patterns/signatures as illustrated by low accuracy of Cosmos for “barnes” and the authors’ confession that the cosmos may function worse than directed predictors do not make a strong case for adopting cosmos. This is all the more significant given that cosmos is likely to require more states than directed optimization. Hence, the argument validating higher investment based on the claim of higher prediction accuracy is not strong since the higher prediction accuracy is dependent on nature of application.
- More memory to accommodate History tables for Message and History – the overhead discussed in the paper is for small cache blocks (128 bytes), so the extension to larger cache blocks in use today is not known.
- The study in this paper used five different benchmarking programs to evaluate the effectiveness and performance of the prediction methodologies in the cache schemes. “Appbt”, “Barnes”, “Dsmc”, “Moldyn”, and “Unstructured” are scientific applications suitable for execution on parallel architectures due to the large numbers of discrete cell units used to model a particular natural process. This is a weakness of the study, because the scope of the benchmarks used is

relatively narrow. The application domain is mainly focused on the use of shared data between the neighboring blocks comprising the discrete cells of the simulations. Thus, a reader has to be skeptical of the applicability of the prediction techniques to multiprocessor or CMP environments outside of the purely scientific benchmarks they were tested on since multiprocessors are also often used in server environments as well for things like webpage serving, transaction processing, etc.

- The Authors state in the beginning that: “Cosmo’s high prediction accuracy is a result of predictable coherence message signatures that arise from stable sharing patterns of cache blocks”. In terms of the benchmarks used, this is expected since the benchmark domain is relatively homogenous. However, it is possible that such “stable patterns” are not the normal case in today’s CMP implementations and applications. For example, how likely is it that a high-throughput data center transaction processing environment will exhibit similar behavior?
- Note that the publish date of the paper is 1998. Thus, this predates the ongoing wave of CMP products currently being released. The use of the history tables to make the actual predictions requires an expansion of the overall memory footprint used to implement the entire cache hierarchy. The authors note that the memory overhead requirements are expected to be less than 22% in most cases. In today’s CMP designs however, this might limit the usefulness of such a design if die space is at a premium and must be reserved for individual tiles or cores. Note that since the publication of this paper, others have suggested various directory protocols and cache enhancements (We have studied these in class earlier in the semester).
- The prediction logic helps to reduce the latencies of remote accesses. In the current CMPs, as opposed to shared memory multiprocessors, remote access latencies are not as critical as the on-chip capacity. This is because all remote accesses are on-chip. The prediction logic further increases the capacity pressure while reducing latency. Thus, it is not practical for CMPs.
- The authors have not discussed anything about the power budget. Considering the complex logic, the implementation will be power hungry as it involves directory based protocol and two-step prediction. Thus it may not be feasible at all in today's CMPs where power budget is critical.
- The way to handle mis-predictions is discussed in the paper. However, the mis-prediction penalty is not quantified.
- The results quantify the number of accurate predictions but do not give any relation between the number of accurate predictions and performance.
- The implementation has a large memory overhead. In some applications, memory requirement is as high as 63%. This may be feasible in SMPs with large off-chip

memories but is infeasible in modern CMPs where capacity is more critical than on-chip latency.

- As the prediction logic is not integrated with the coherence protocol, it requires more hardware resources to store the Message and pattern history tables. Also it is less cost effective as compared to the integrated optimizations.
- Unlike the branch prediction, where the state machine has 1-2 bits for encoding, Cosmos has multiple bits to handle prediction. Thus the logic is highly complex.