Composable Cache Model Evaluation

Vlastimil Babka

DEPARTMENT OF DISTRIBUTED AND DEPENDABLE SYSTEMS
FACULTY OF MATHEMATICS AND PHYSICS
CHARLES UNIVERSITY IN PRAGUE

http://d3s.mff.cuni.cz
Context: Composable Cache Model (CCM)

• Scenario: (Two) workloads sharing an L2 cache
• Goal: predict the increase of cache misses due to limited shared cache capacity, and the resulting slowdown of the workloads
• Input: for each workload
  ▪ Stack distance profile (averaged over all cache sets)
  ▪ Accesses per instruction (API)
  ▪ Instructions per cycle (IPC) when run in isolation
  ▪ L2 cache miss penalty (in processor cycles)
• Output: modified stack distance profile and IPC, due to cache capacity sharing
  ▪ Potential for further stack profile composition
CCM Accuracy Evaluation

- On workloads from the RPG framework
  - SPEC CPU2006 benchmarks, FFT, LZW...
- Model inputs obtained using stressmark approach (Xu et. al, ISPASS'10)
  - Reconstructs stack profile by observing cache miss rates while varying effective cache size
  - Also determines cache miss penalty by regression
- Predictions compared with measurements, running the pairs of workloads in parallel via RIP framework
  - Implementation helped by Tomáš Martinec
- Initial simplification: disabled L2 prefetching
CCM performance misprediction
(depending on measured slowdown due to sharing)
CCM miss rate (miss per access - MPA) misprediction
(depending on miss rate increase due to sharing)
Possible Sources of Bad MPA Prediction

• Imprecise model input
  ▪ Stack profiles obtained from stressmark
  ▪ All cache sets averaged in one profile

• Wrong IPC prediction (due to other factors than cache capacity)
  ▪ IPC used in the iteration phase of the model
  ▪ If strongly biased against one workload in the pair, it would become relatively faster in each iteration, occupying more and more cache and making the other workload slower and slower

• Fundamentally wrong / insufficient model ...
Obtaining Stack Profiles from Valgrind

• Valgrind is a tool for dynamic binary instrumentation
  ▪ Can capture each memory access
  ▪ Built-in LRU cache simulator (cachegrind)
    • Determines cache hits and misses without perf. counters
    • Extended to gather also stack distance profiles in a controlled way

• Despite heavy instrumentation faster than stressmark (less repetitions needed)
  ▪ ~5 hours vs ~2.5 days
  ▪ Stressmark results still used to estimate cache miss penalty
Stack profiles obtained by stressmark (black) and valgrind (white)
CCM (valgrind profiles) miss rate (miss per access - MPA) misprediction (depending on miss rate increase due to sharing)
CCM (valgrind profiles) performance misprediction
(depending on measured slowdown due to sharing)

Difference in predicted / measured throughput [%]

-20  0  20  40  60

Measured slowdown compared to isolated execution [%]

-20  0  20  40  60

Workload
- o astar
- △ bzip2
- + fft
- x lbm
- ♦ libquantum
- ▽ lzq
- ▽ mcf
- * namd
- ♦ sjeng
CCM predicted slowdown due to sharing
(depending on measured slowdown due to sharing)
Prediction with Valgrind-based Profiles

- With valgrind-based profiles, the miss rate prediction is significantly better
  - Worse only for workloads where it does not matter anyway (small cache access rate)
  - Other sources of MPA misprediction less important
- Throughput (IPC) prediction also much better
  - Further plots are based on these profiles unless noted otherwise
  - Stressmark results still used to determine miss penalty
    - It could be possible to reduce the number of stressmark settings measured
Possible Sources of Tput Misprediction

• Miss rate misprediction?
  ▪ Does IPC misprediction depend on MPA misprediction?
  ▪ Under-predicted MPA could lead to over-predicted IPC (and vice versa)
CCM performance misprediction
(depending on miss rate misprediction)
CCM predicted performance
(depending on miss rate misprediction)
Possible Sources of Tput Misprediction

• Miss rate misprediction?
  ▪ Yes, but responsible only for the smaller errors

• Wrong miss penalty?
  ▪ Does IPC misprediction depend on the difference between measured shared and isolated MPA?
    • Linear relationship would indicate wrongly determined, but still constant miss penalty
    • Superlinear relationship would indicate non-constant miss penalty (e.g. due to memory bus sharing)
    • No relationship could indicate cache request handling capacity sharing
CCM performance misprediction
(depending on miss rate increase due to sharing)

Difference in predicted / measured throughput [%]

Measured increase in miss rate compared to isolated execution [MPA_shr - MPA_isol]
CCM performance misprediction
(depending on miss rate increase due to sharing)

Difference in predicted / measured throughput [%]

Measured increase in miss rate compared to isolated execution [MPA_shr - MPA_isol]
Possible Sources of Tput Misprediction

• Wrong miss penalty seems to cause many cases of mispredicted throughput
• Superlinear dependency on increasing miss rate suggests memory bus sharing
• This can be tested by measuring slowdown when workloads share the memory bus, but not the L2 cache (by picking different cores)
  ▪ See if this slowdown is related to misprediction
CCM performance misprediction for shared cache
(depending on measured slowdown due to interference on NOT shared cache)
Comparison with the CAMP Model

• CAMP Model (Xu et. al, ISPASS'10)
  ▪ Same inputs, different approach
  ▪ Also iterative, but to solve system of equations
  ▪ Outputs: effective cache occupations, IPC

• Our implementation in R (with T.Martinec)
  ▪ Some simplifications on not-so-clear details
  ▪ Different stack profiles interpolation

• Some pairs of workloads problematic
  ▪ Especially (but not only) those with low MPA
  ▪ Solutions not found, exceptions or infinite cycles
CCM (black) and CAMP (red) miss rate misprediction (stressmark profiles) (depending on miss rate increase due to sharing)
CCM (black) and CAMP (red) miss rate misprediction (valgrind profiles) (depending on miss rate increase due to sharing)
CCM (black) and CAMP (red) performance misprediction (valgrind profiles) (depending on measured slowdown due to sharing)

Difference in predicted / measured throughput [%]

Measured slowdown compared to isolated execution [%]
Possible Sources of Tput Misprediction

• Miss rate prediction is similar to CCM
• IPC prediction is quite pessimistic
  ▪ Compared to both CCM prediction and actual measurements
• The difference could be in the approach of converting miss rate changes to performance
  ▪ CCM estimates miss penalty from the difference in MPA (w.r.t. isolated execution) and times
  ▪ CAMP models time as $A + B \times \text{cache\_misses}$
    • Estimates $A$ and $B$ from stressmark results
    • This mixes together both misses that are part of isolated execution, and due to the sharing
CCM (black) and CAMP with IPC modification (red) miss rate misprediction (depending on miss rate increase due to sharing)
CCM (black) and CAMP with IPC modification (red) performance misprediction (depending on measured slowdown due to sharing)
Better Look at the Differences

• The change allowed more pairs of workloads to be successfully predicted with CAMP
• On previous plots it was not clear which points were new and which changed prediction error
• Let's compare only the results where we have data for both modified and unmodified CAMP
CAMP with IPC modification miss rate misprediction
(depending on unmodified CAMP miss rate misprediction)
CAMP with IPC modification performance misprediction
(depending on unmodified CAMP performance misprediction)
Conclusion

• CCM probably a promising direction
  ▪ Good miss rate prediction using valgrind profiles
    • Stressmark-based profiles yield bad results
  ▪ Throughput prediction not always precise
    • Need to incorporate (probably) memory bus sharing
  ▪ Prefetching could be also a problem
    • Measurements with it enabled are running...
  ▪ Not worse than CAMP (in our implementation)
    • Convergence problems in CAMP
    • Similar miss rate prediction accuracy
    • Better IPC prediction accuracy
      ▪ Only partially caused by different miss penalty handling
Very-Near-Future Work

• Evaluation with prefetching enabled
• Predicting each cache set separately
  ▪ Could improve the cases with wrong MPA
  ▪ Already implemented, but very slow
• Improving the IPC accuracy
  ▪ Modeling miss penalty depending on combined miss rate?
• Better comparison with CAMP (?)
• Use the model for performance prediction
  ▪ RPG generated systems + SimQPN
• Evaluate compositionability
Thank you.