R1-5

Dynamic Data Migration to Eliminate Bank-level Interference for Stencil Applications in Multicore Systems

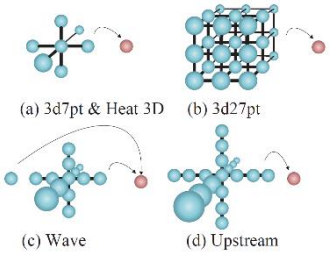
Wei Hen Lo, Yen Hao Chen, and TingTing Hwang

Department of Computer Science, National Tsing Hua University, R.O.C

Turtleevil_1@hotmail.com, yhchen@cs.nthu.edu.tw, tingting@cs.nthu.edu.tw

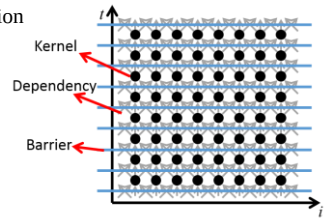
Stencil Applications (Stencils)

A stencil computation repeatedly updates each point of a d-dimensional grid as a function of itself and its near neighbors. It is one of the most fundamental computational patterns in lots areas.



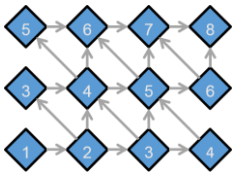
Barrier Synchronization (Sync.)

Modern automatic transformation compiler framework can generate efficient tiling parallel stencil codes by increasing data locality and thread parallelism.



Dynamic Scheduling (Dyn.) [1]

Dynamically scheduling parallel stencils improves system performance. But, it also exacerbates memory contention problem because of more requests set to the DRAM memory.

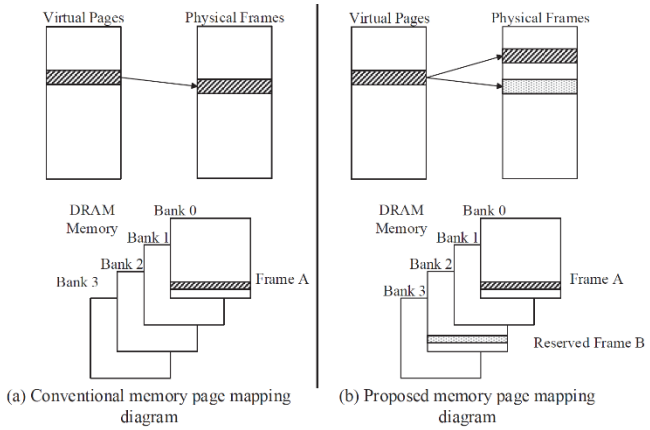


OS Page Coloring (OSC.) [2]

Traditional OS page coloring method which partitions the memory pages in advance cannot alleviate the memory contention in dynamic scheduling parallel stencils. It replaces the bank bits with CPU core ID to avoid to consecutive memory requests sent to a same DRAM bank (DRAM band conflict). But by our experiment, stencils still experience lots of DRAM bank conflicts because of data reusing by other CPU cores.

Proposed Data Migration Method

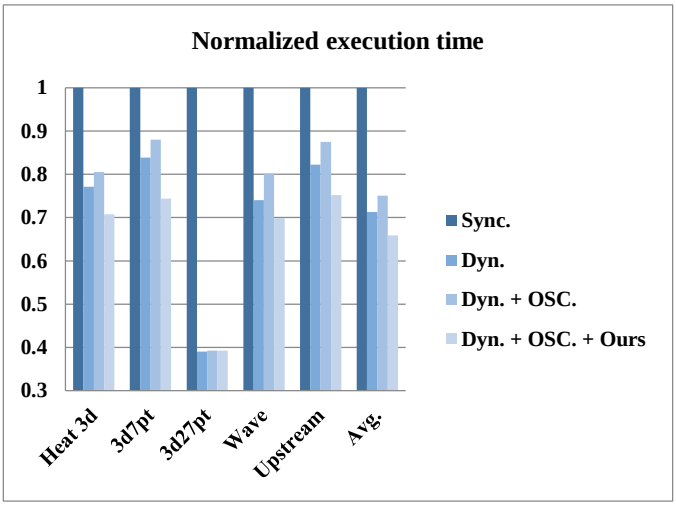
A new software/hardware cooperative dynamic data migration method is proposed to avoid DRAM bank conflicts by exploiting the update-and-reuse property of stencils. When the OS receives a page fault, it tries to find a group of free frames whose physical address is only different in the bank bits. And reserve those frames for the virtual page. The memory controller migrates a data block to other bank only when a write request is sent from the last level cache, so called Migrate-On-Write. By this technique, DRAM bank conflicts can be avoided, and therefore, the system performance can be improved.



(a) Conventional memory page mapping diagram

(b) Proposed memory page mapping diagram

Experimental Results



Conclusion

We have designed and evaluated a new dynamic data migration method to address the DRAM bank conflict problem of dynamic scheduling parallel stencils. By the experimental results, our methodology improves performance by 7% on the 8-core system and 9.3% on the 16-core system.

Reference

[1] M. M. Baskaran, N. Vydyanathan, U. K. Bondhugula, J. Ramanujam, A. Rountev, P. Sadayappan, "The Compiler-Assisted Dynamic Scheduling for Effective Parallelization of Loop Nests on Multicore Processors," PPoPP'09, pp. 219-228, April 2009.

[2] L. Liu, Z. Cui, M. Xing, Y. Bao, M. Chen, C. Wu, "A software memory partition approach for eliminating bank-level interference in multicore," PACT'12, pp. 367-376, September 2012.