Regularities Considered Harmful: Forcing Randomness to Memory Accesses to Reduce Row Buffer Conflicts for Multi-Core, Multi-Bank Systems

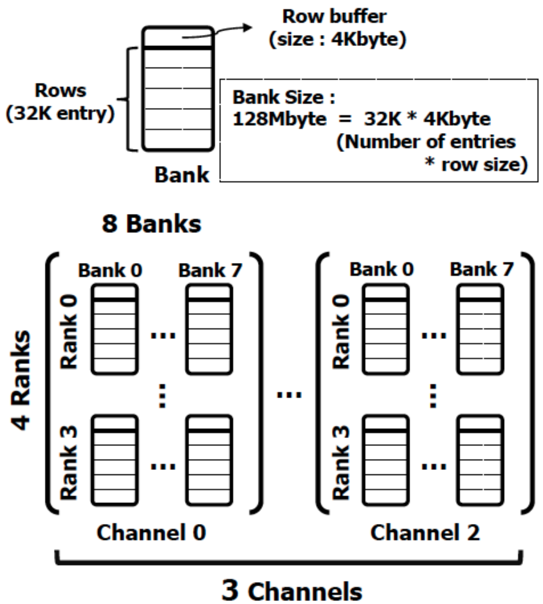
Heekwon Park, Seungjae Baek, Jongmoo Choi
Donghee Lee and Sam H. Noh

72131715 Neo Kim
phoenixise@gmail.com

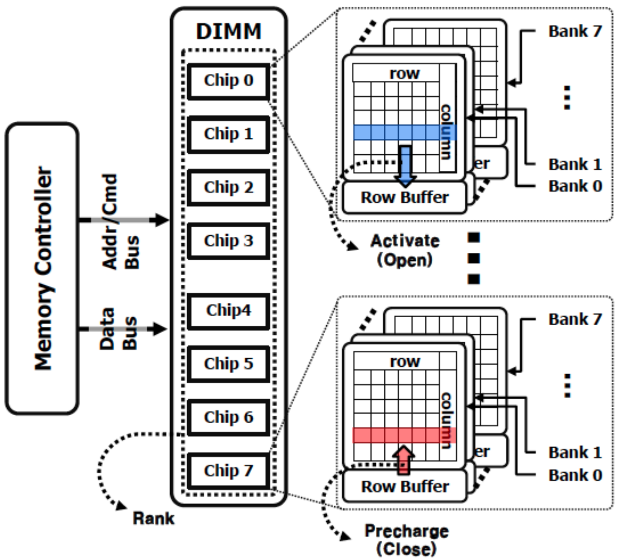
Contents

Background

- ## Memory Organization
  - Memory organization consists of multiple channels.
  - A channel is divided into multiple ranks.
  - A rank is divided into multiple banks.
  - A bank consists of a set of rows and a row-buffer.



(a) Conceptual memory organization



(b) Physical memory module

Background

- Row-buffer
  - There is a row-buffer for each bank. It is a cache area of a bank.
  - It is intended to exploit spatial locality.
  - If data requested by a core is cached in the row-buffer, it can be served immediately. It is called a **row-buffer hit**.
  - Otherwise, a row-buffer needs two operations of 'precharge' and 'active'.
    - Precharge operation is to write back data.
    - Active operation is to load the entire data from a row.
  - It is called a **row-buffer conflict**.

Q : Is it necessary to precharge even if there is no change in the cache?

Background

● Row-buffer conflict

- It eliminates the caching effect.
- More delays and energy consumption.
- Degrades memory access latency by 2 to 5 times

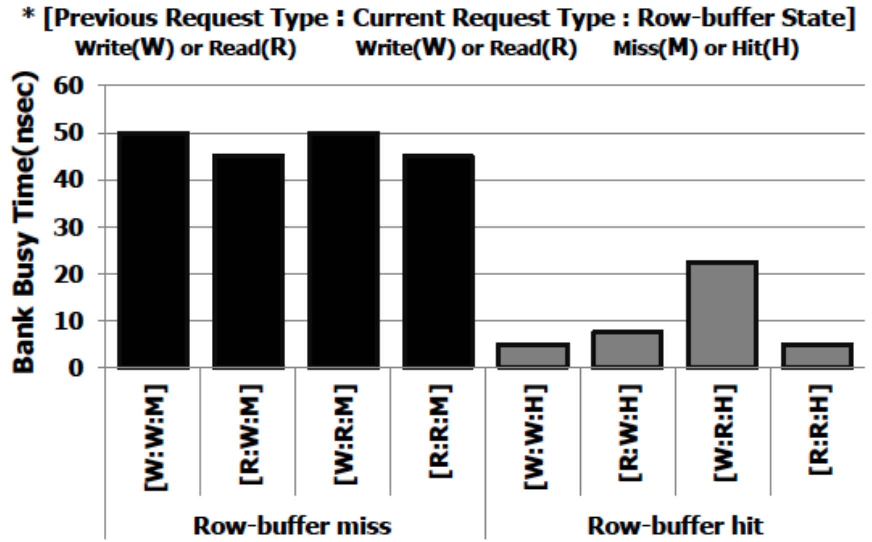


Figure 2. Row-buffer hit and conflict overhead

Background

- Row-buffer conflict
  - It eliminates the caching effect.
  - More de
  - Degrade

Figure 2. Row-buffer hit and conflict overhead

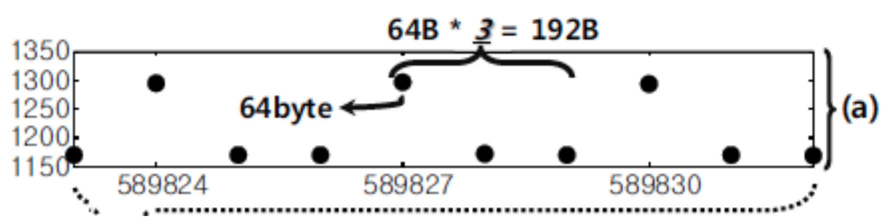
Memory Organization Analysis (1)

- ## Analysis Tool
  - To explore the internal structure of the memory organization
  - Employed three techniques
    - Set the CPU cache mode as uncacheable to guarantee that each access is serviced at the memory module.
    - One iteration has numerous times of accessing two variables to cancel out measurement noise
    - Mutually dependent two variables to avoid optimizations

Memory Organization Analysis (2)

Analysis Results

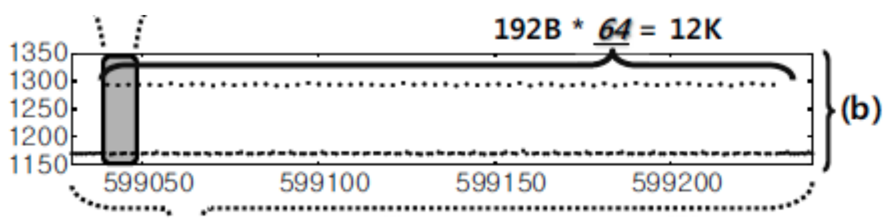
- Figure 4(a) shows a pattern repeating every 3 addresses.
- It means that a row-buffer conflict occurs every 3 addresses.
- And there are 3 channels.
- Channel interleaving is performed with the cache line unit.



Memory Organization Analysis (3)

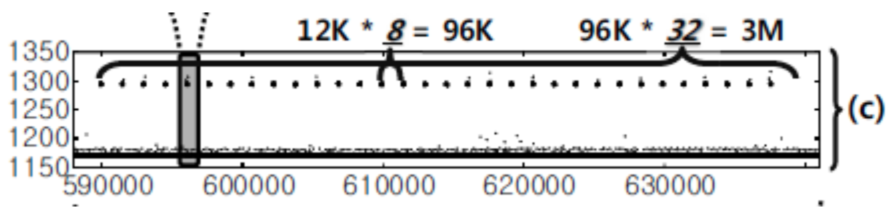
Analysis Results

- Figure 4(b) shows that the pattern in (a) occurs 64 times continuously.
- It is 12KB (64B * 3 * 64) and the row size is 4KB, so three consecutive rows are managed in the same manner.



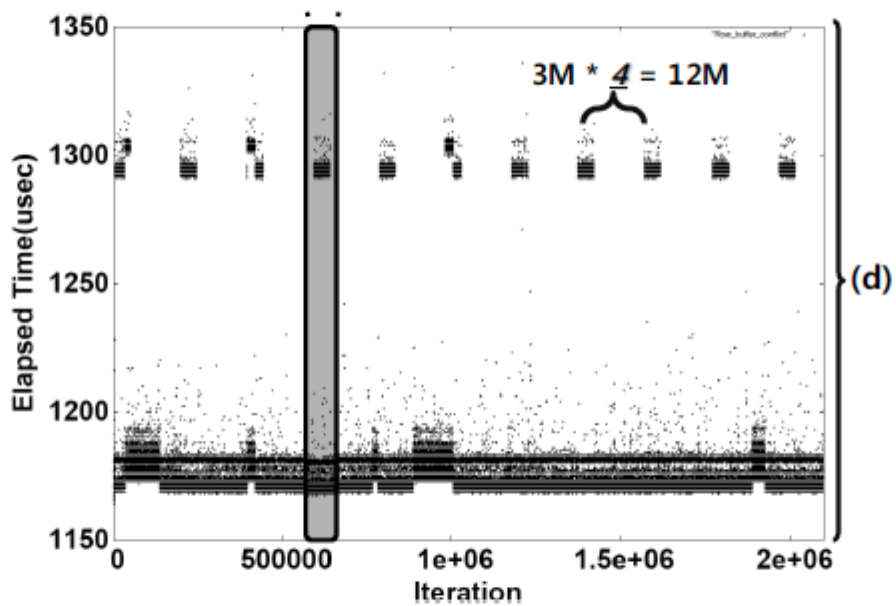
Memory Organization Analysis (4)

- ## Analysis Results
  - Figure 4(c) shows that the pattern in (b) occurs in 96KB (12KB * 8) interval.
  - And the system has 8 banks.
  - Bank interleaving is configured in three row units.
  - The 96KB pattern is repeated 32 times of 3MB(96KB * 32) memory size.



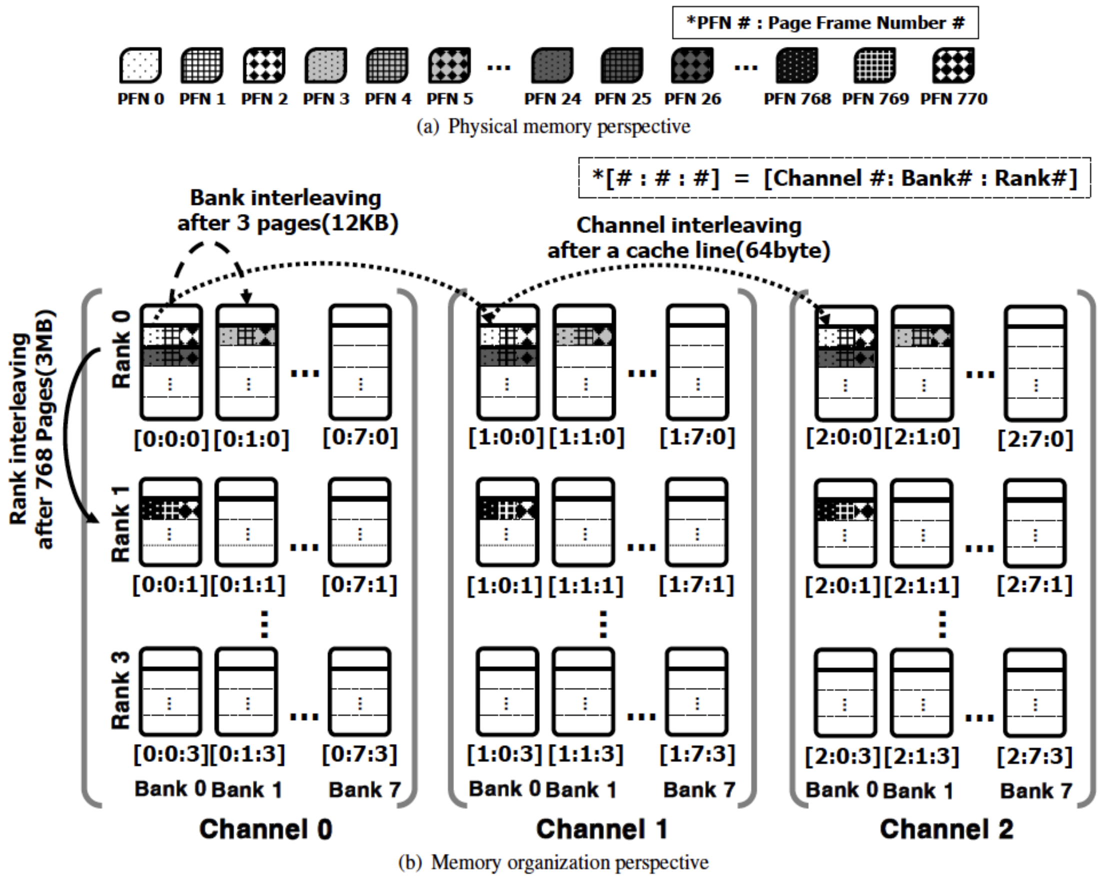
Memory Organization Analysis (5)

- ## Analysis Results
  - Figure 4(d) shows that the pattern in (c) occurs in 12MB interval.
  - And the system has 4 ranks.
  - Rank interleaving is performed in 3MB units.



Memory Organization Analysis (6)

- Implication



Implementation

- **Observations**
  - Row-buffer conflict is even worse in a multi-core system.
  - Memory partitioning is one of feasible solution, but also has some issues.
    - Reduced memory parallelism
    - Causes a scalability issue
    - Unavailable of allocating large consecutive page frames
    - More serious conflicts when an application is migrated to another core
  - Random memory access

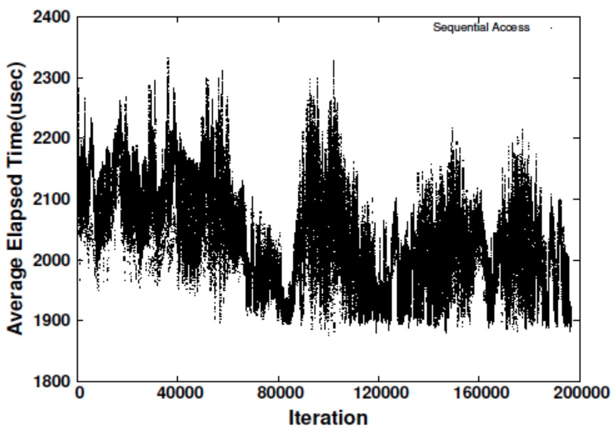


Figure 6. Row-buffer conflicts with 4 cores: sequential access pattern

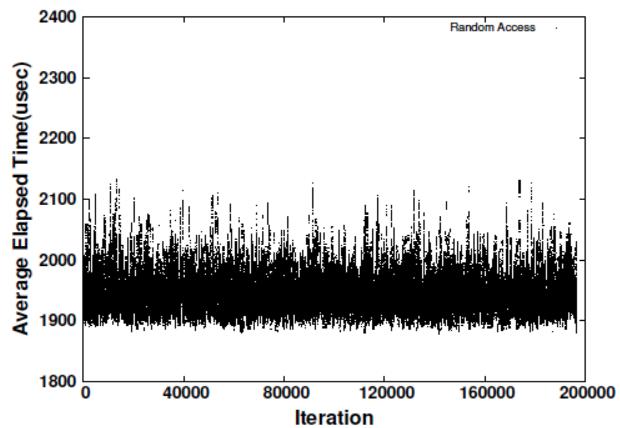


Figure 7. Row-buffer conflicts with 4 cores: random access pattern

Implementation

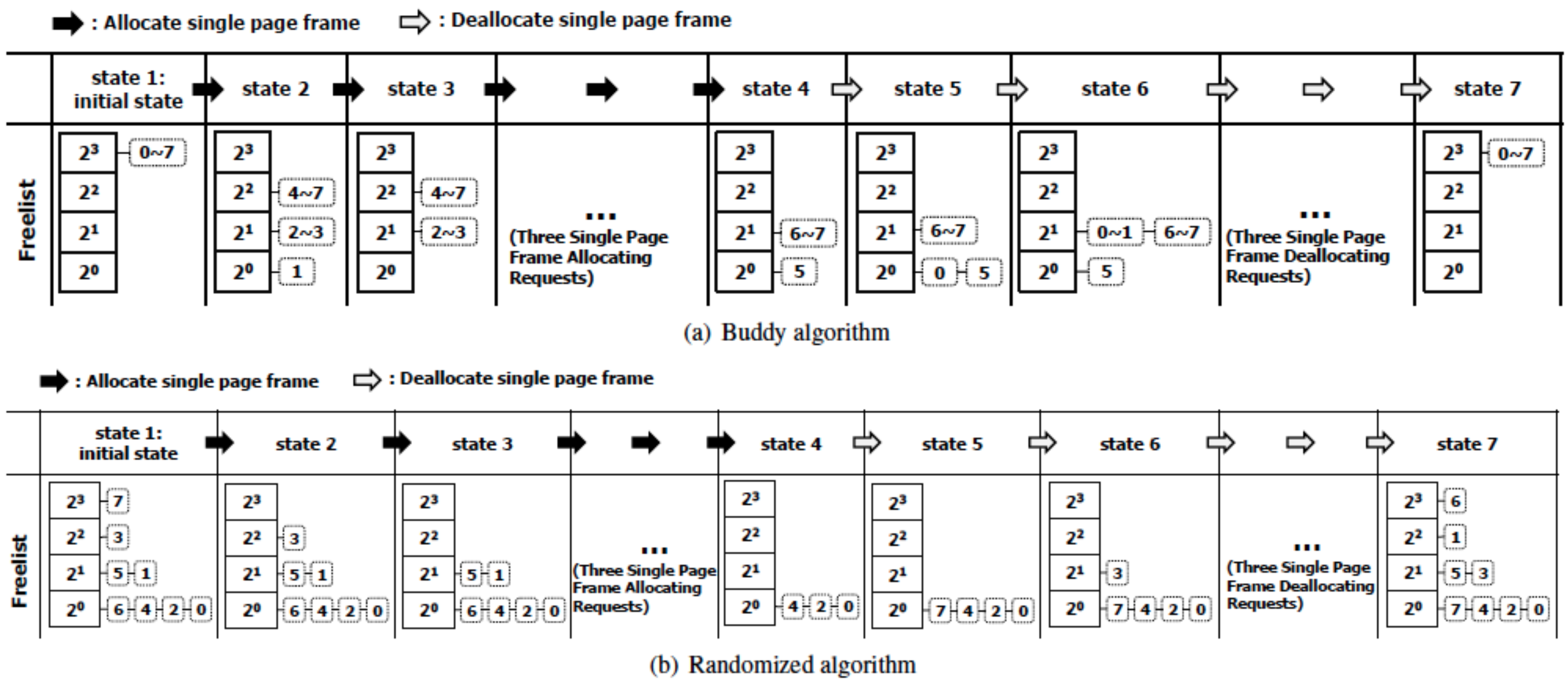
- ## Memory Container
  - Page frames allocated to each individual thread should come from distinct banks as much as possible
  - For that, a memory container is introduced, which is a unit of memory that comprises the minimum number of page frames that can cover all the banks of the memory organization.
  - According to the previous analysis, it is 12MB. But for being generic, it is set to 4MB.

Implementation

- ## Randomized Algorithm
  - Replaces the buddy algorithm.
  - Two new concepts are introduced into the buddy algorithm
    - Individual page frame management
    - Downward search
  - The location of the page has a double meaning.
  - Allocation by the buddy algorithm has a strong tendency to be regular.
  - But the randomized algorithm, the arrangement of page frames are getting different in the consecutive allocations and deallocations.

Implementation

- ## Randomized Algorithm



(a) Buddy algorithm

(b) Randomized algorithm

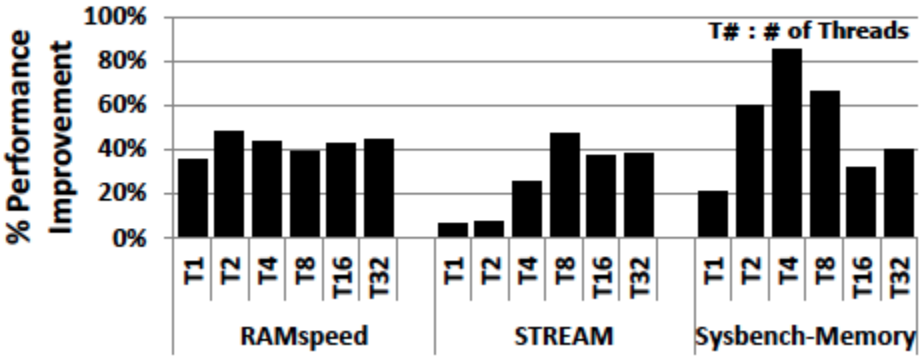
Figure 8. Comparison between buddy and our randomized algorithm

Evaluation

- ## Benchmarks
  - Seven benchmarks which are categorized into 3 groups.
  - The first group consists of memory intensive benchmarks.
    - Styream, Sysbench-memory and Ramspeed
  - The second group consists of CPU or I/O intensive benchmarks.
    - Kernel compile, Dbench and Unixbench
  - The third group is the PARSEC benchmark for diverse application domains.

Evaluation

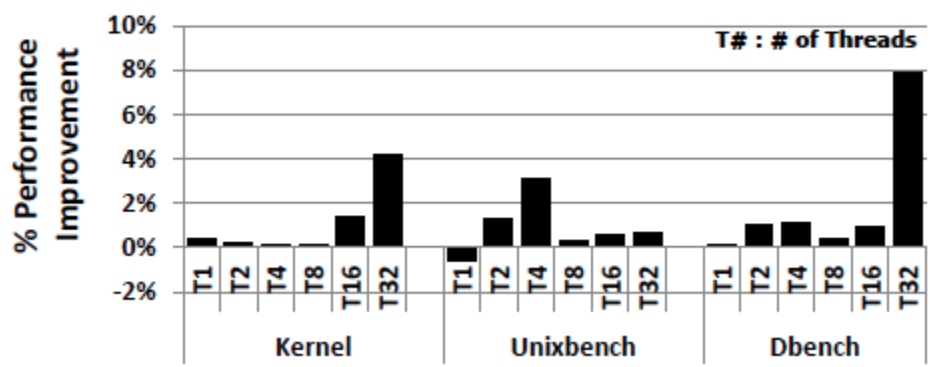
- ## Memory Intensive Benchmarks
  - Performance is improved by as little as 6.5% to as much as 85.2%



(a) Memory intensive benchmark results

Evaluation

- CPU or I/O Intensive Benchmarks
  - The improvements are not that considerable since memory references are only a small portion.



(b) CPU or I/O intensive benchmark results

Evaluation

- PARSEC Benchmark
  - According to the characteristics of the programs, performance improvements show difference aspects.
  - Significant improvement (avr. 10.4%)
    - Programs accessing a large memory range like canneal, facesim, fluidanimate and streamcluster.
  - Unnoticeable difference
    - Programs which have irregular acess patterns like blackscholes, freqmine, swaptions and x264.
  - Remaining programs shows improvements (avr. 2.2%)

Conclusion

- Two goals of M^3
  - To dedicate multiple banks to a core to maximize memory parallelism
    - New notion of a memory container is devised.
  - To reduce cases where multiple cores access the same bank
    - Randomizing memory allocation algorithm is introduced.

- Future work
  - Single page frame allocation can be done in O(1).
  - M^3 issues such as fragmentation and lock contention.

References

- Memory bank, http://en.wikipedia.org/wiki/Memory_bank
- Row-buffer precharge, http://www.ece.eng.wayne.edu/~sjiang/ECE7995-winter-09/lecture-7.pdf
- Channel Interleaving, http://en.wikipedia.org/wiki/Interleaved_memory
- Parallelism Issues on Modern Memory Architecture, http://dcslab.hanyang.ac.kr/nvramos12/presentation/[NVRAM]Choi_Dankook.pdf
- Out-of-order Execution, http://en.wikipedia.org/wiki/Out-of-order_execution