

Parallelism Issues on Modern Memory Architecture (OS Perspective)

2012. 10. 16

Jongmoo Choi

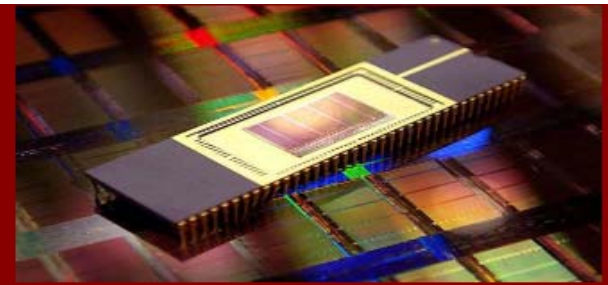
(In cooperation with Sam H. Noh, Donghee Lee)

<http://embedded.dankook.ac.kr/~choijm>

NVRAMOS 2012 Fall

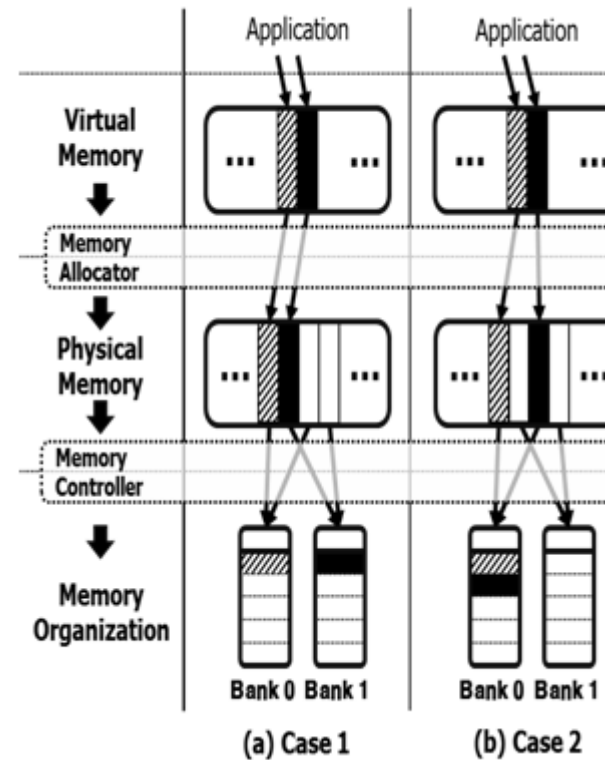
Operating System Support for
Next Generation Large Scale NVRAM

Organized by KIISE, October 15 - 17, 2012, Jeju, Korea



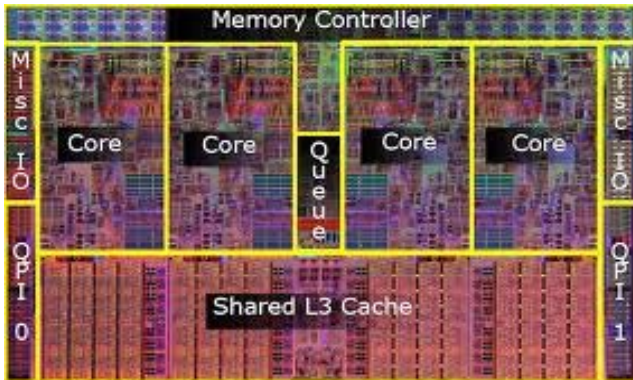
Contents

- Introduction
- Observation
- Reasoning
 - ✓ Low-buffer conflict
 - ✓ Page frames to banks relation
 - ✓ Sequential vs. random allocation
- Lessons
- Conclusion



Introduction

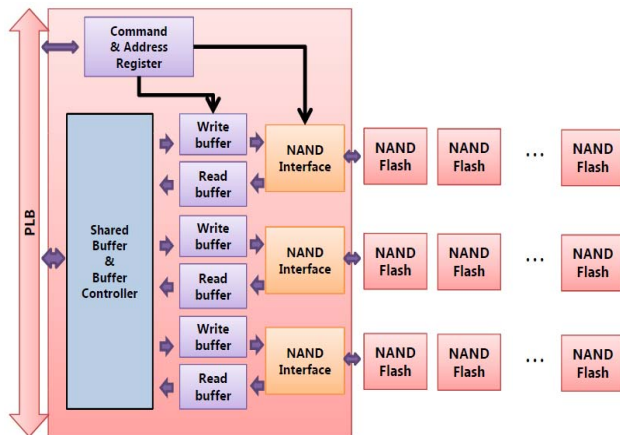
■ Parallelism everywhere



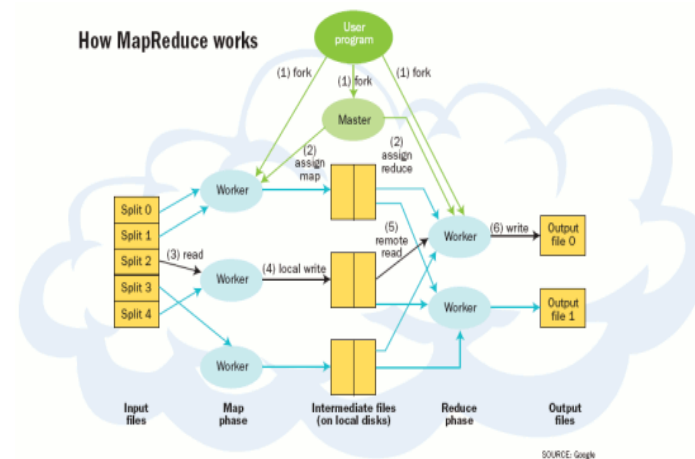
Multicore (Manycore)



CUDA on GPGPU



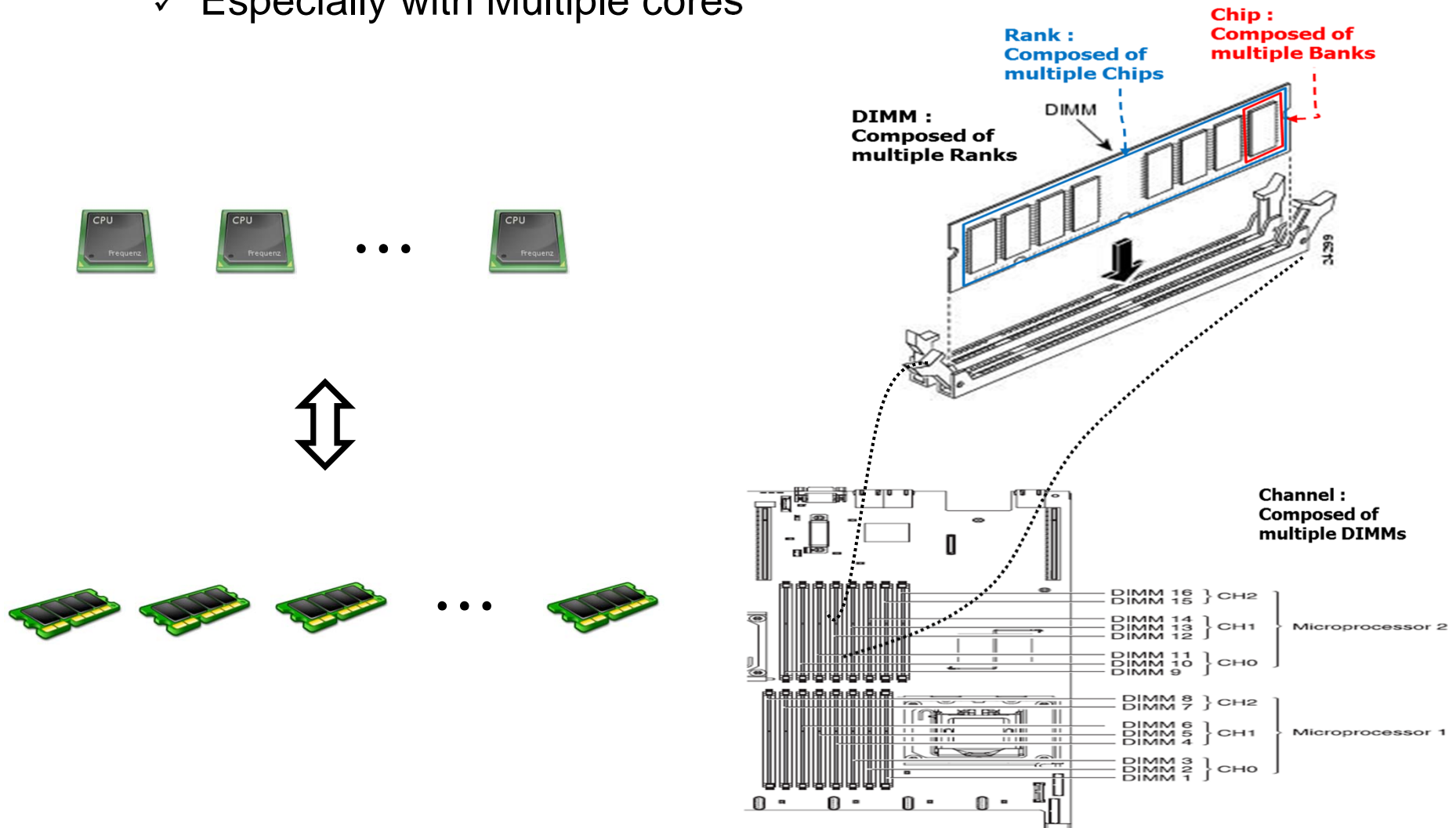
Multi-channel/way in SSD



MapReduce for Bigdata

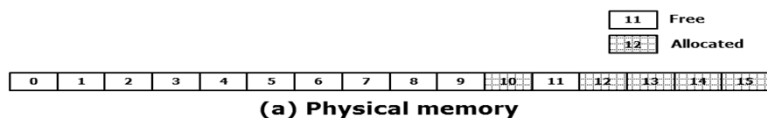
Introduction

- Parallelism on DRAM
 - ✓ Especially with Multiple cores

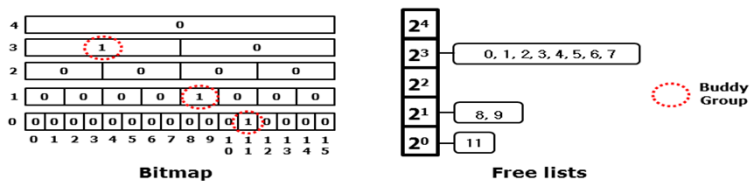


Observation

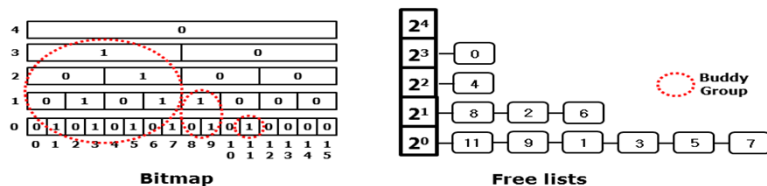
- New buddy system: iBuddy (inverse Buddy)
 - ✓ Buddy: allocation of page frames
 - ✓ iBuddy
 - Individual page frame management (vs. Group management)
 - Top-down search (vs. Bottom up search)
 - Split/coalescing at free/allocation time (vs. vice versa in Buddy)



(a) Physical memory



(b) Buddy System



(c) iBuddy System

iBuddy Concept

NVRAMOS 2011 Spring
Operating System Support for Next Generation Large Scale NVRAM
Organized by KIISE, April 18 - 20, 2011, Jeju, Korea

Home Committee **Program** Venue Photos Registration Past Events

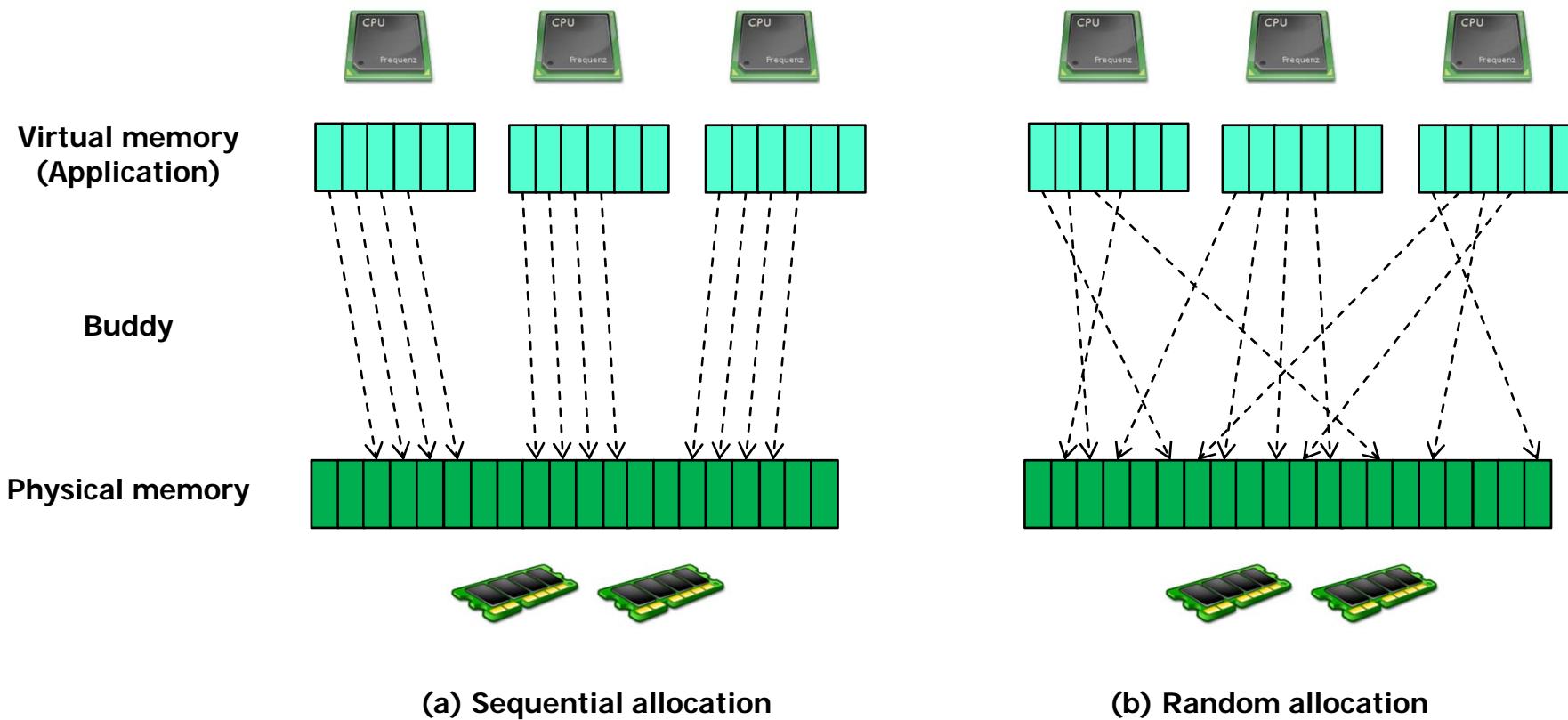
Program

Time	Subject	Speaker
11:00 - 18:00	Registration	
	DAMO Session:	
	• <i>SSD Market Trends</i> , Daehye Lee (PaxDisk)	
	• <i>SSD Future Trends and Research Issues</i> , Sooyong Kang (Hanyang Univ.)	
	• <i>Research Issues in Flash Memory-based Mobile Storage</i> , Dongjun Shin (Samsung)	
	• <i>System-Wide Issues for Efficient Use of E-SSD</i> , Kyung Ho Kim (Samsung)	
13:00 - 14:30		
14:30 - 15:10	<i>Smart SSD Controller Design for Improving Random Read Performance</i>	Yongsoo Joo (Ewha Womans Univ.)
15:10 - 15:40	Coffee Break	
15:40 - 16:20	<i>Object-based Solid State Drives</i>	Jinsoo Kim (SKKU)
16:20 - 17:00	<i>Flash Memory Reliability Model Based on Operations and Faults</i>	JiHyuck Yun (SNU)
19th, April: Lotus Hall		
Time	Subject	Speaker
9:00 - 14:00	Registration	
9:00 - 9:40	<i>Operating system supports for SCM as main memory systems</i>	Jongmoo Choi (Dankook Univ.)
9:40 - 10:20	<i>Write reference characteristics and SCM-based memory management</i>	Hyekyung Bahn (Ewha Womans Univ.)
10:20 - 10:40	Coffee Break	

NVRAMOS'11

Observation

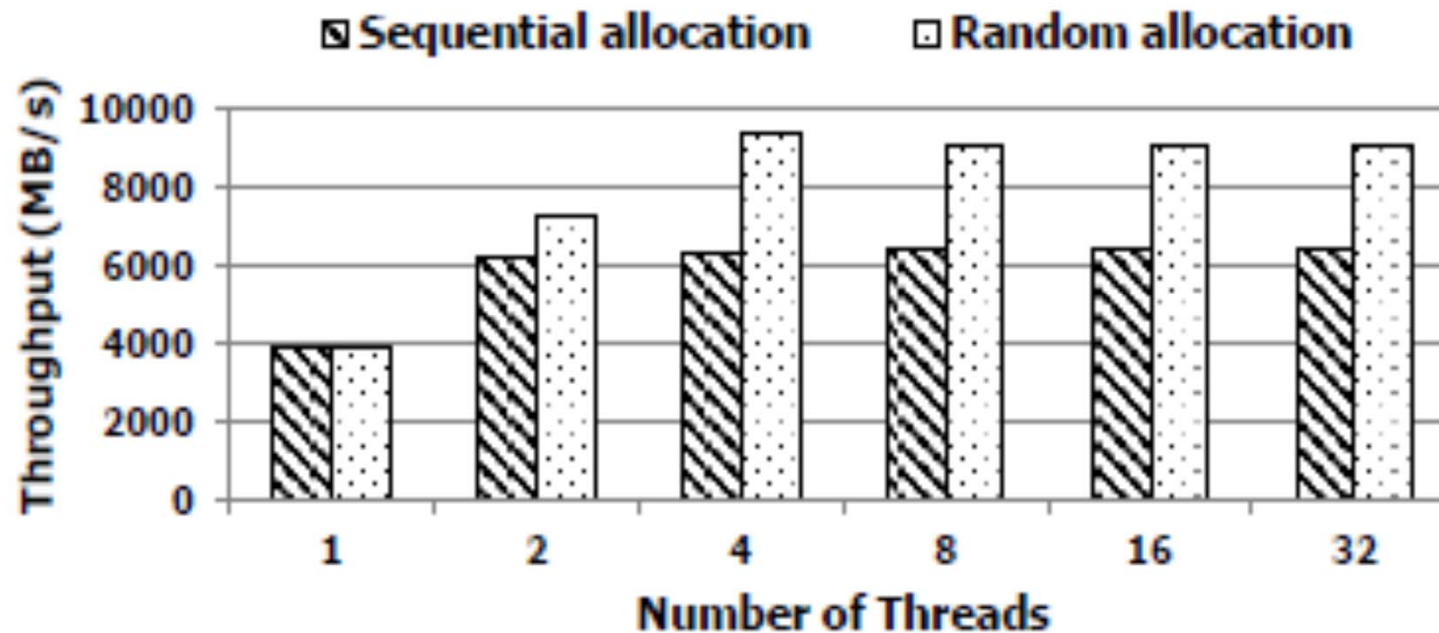
- Allocation of page frames
 - ✓ Buddy can control allocation sequences



Observation

■ Experimental results

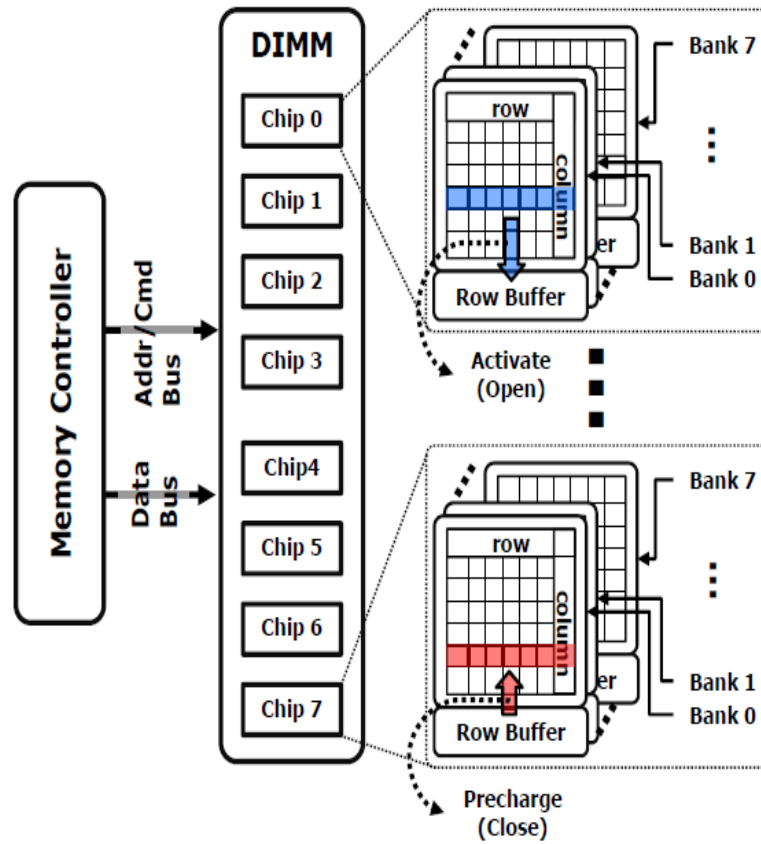
- ✓ Application: Stream benchmark
- ✓ System: Intel Xeon (quad-core), 32GB DRAM



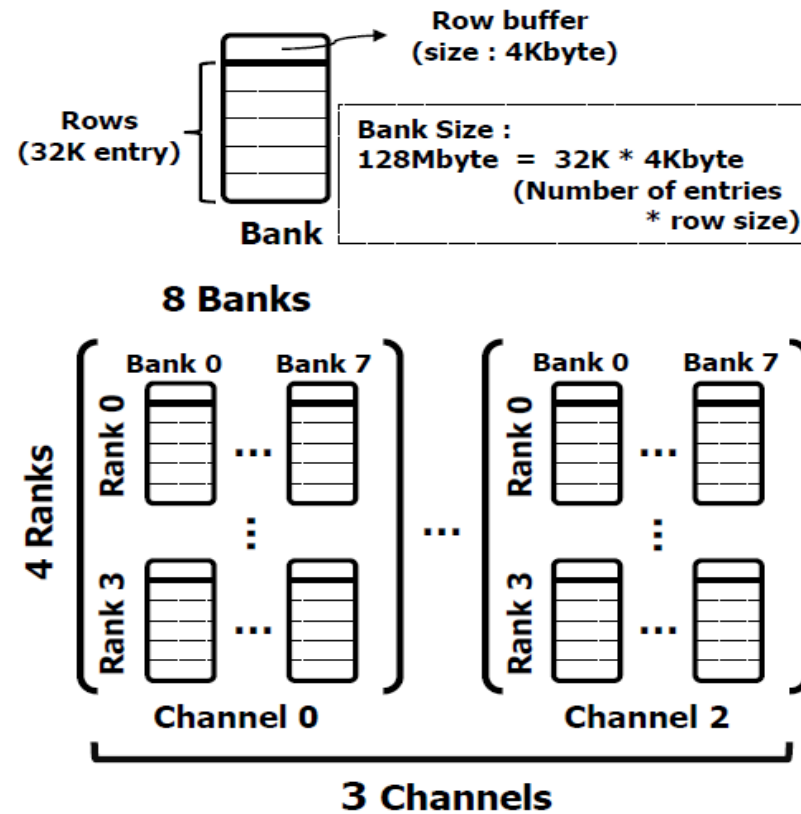
👉 Page frame allocation policy affects performance especially on Multicore

Reasoning 1: Row-buffer conflict

- Memory organization
 - ✓ Channel, Rank, Bank
 - ✓ Row and Row-buffer



(a) Physical memory module

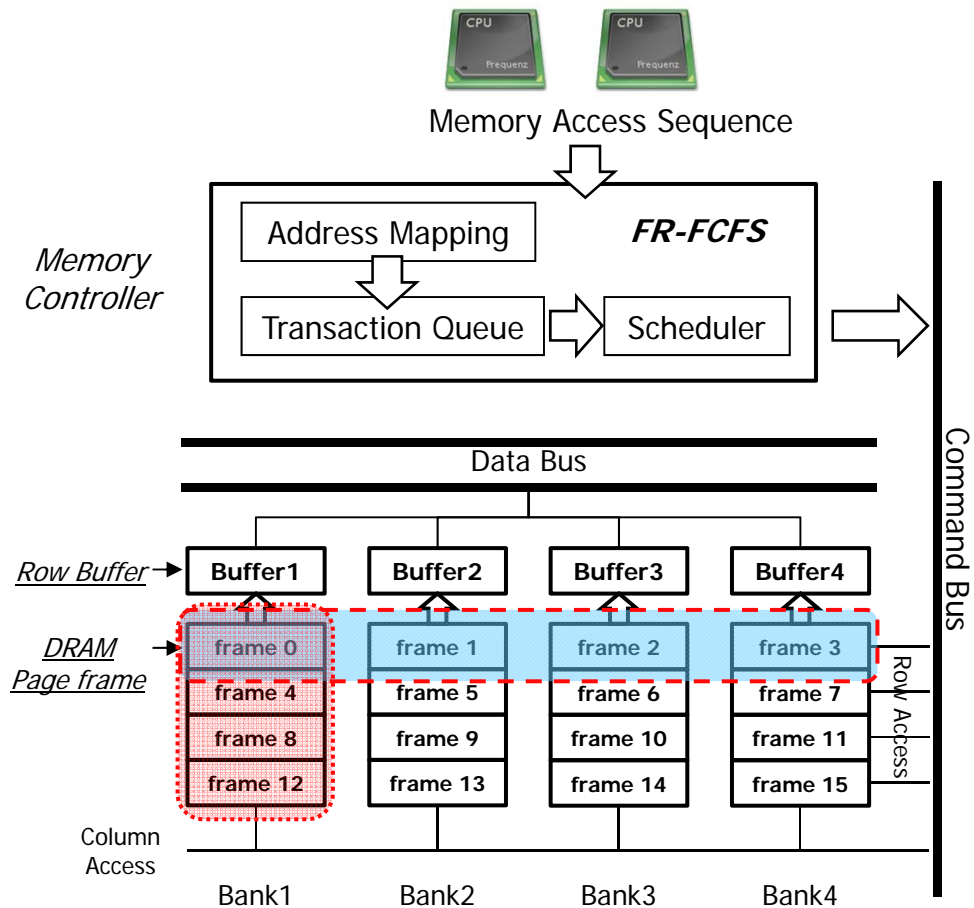


(b) Conceptual memory organization

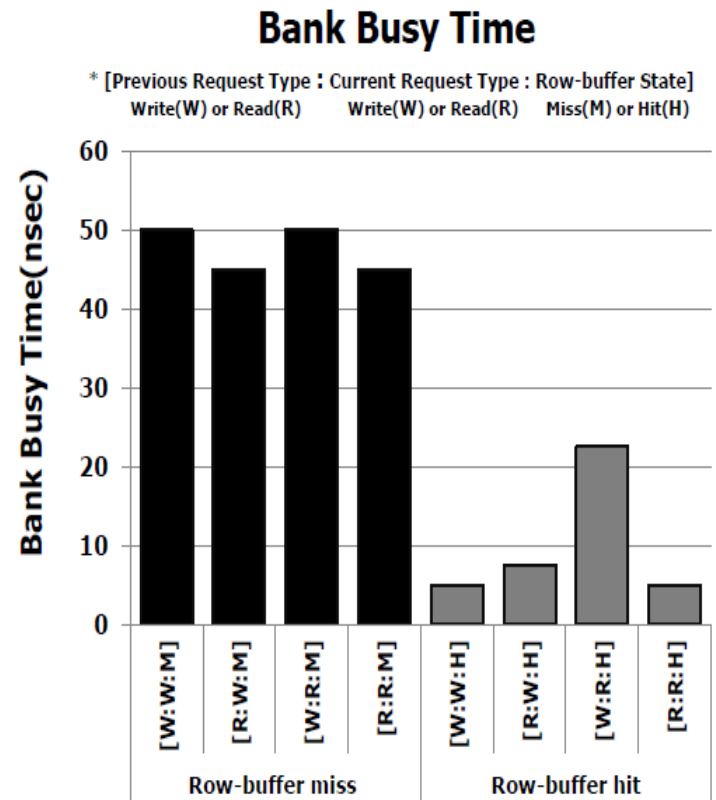
Reasoning 1: Row-buffer conflict

■ Row-buffer

- ✓ A kind of cache area: hit and miss (conflict)
- ✓ Conflict cost: activation, Precharge



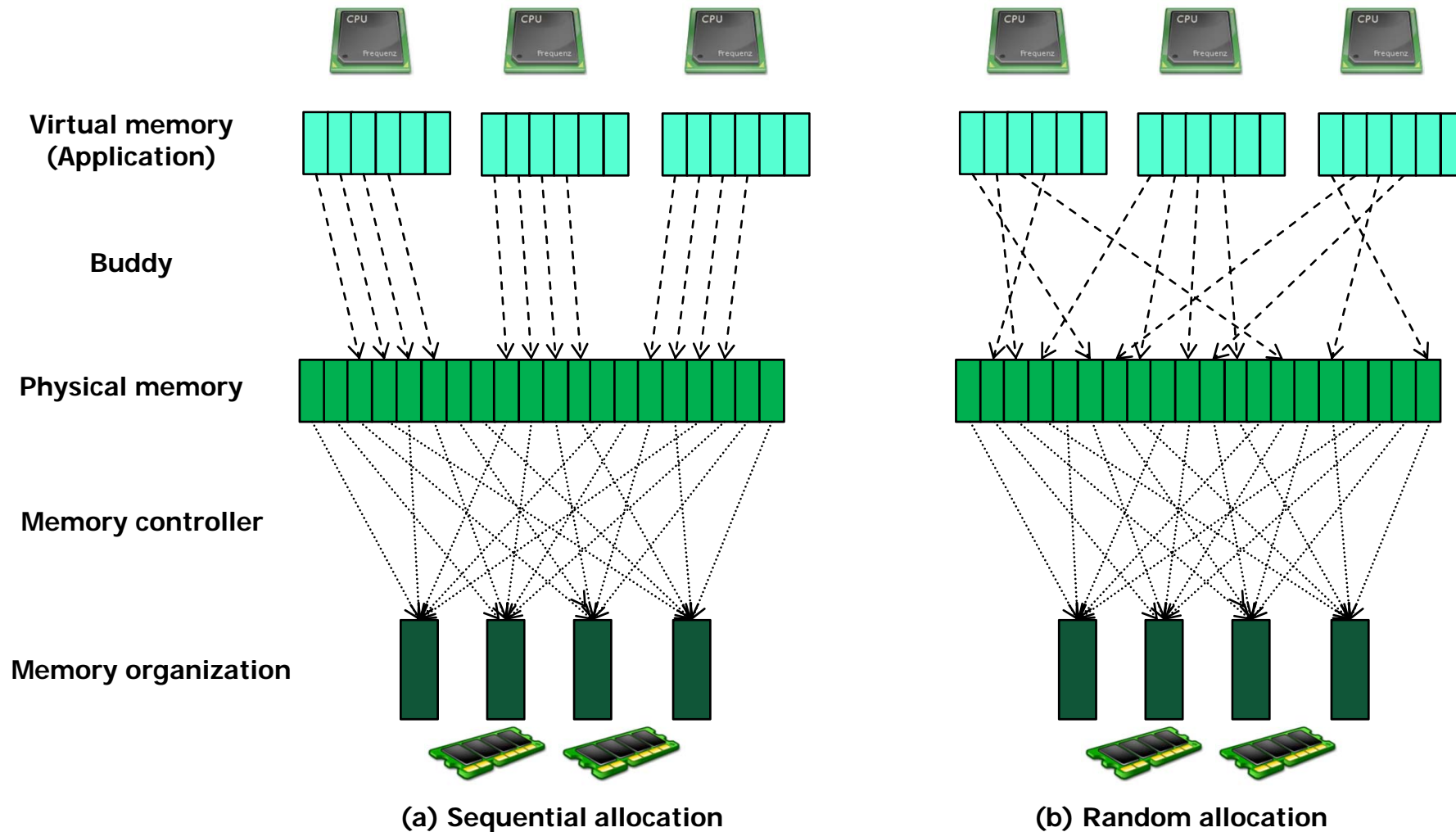
(a) Role of Row Buffer



(b) Row Buffer conflict cost

Reasoning 1: Row-buffer conflict

■ Memory hierarchy reconsideration



☞ Performance difference is due to the row-buffer conflict → Need new layer?

☞ frames-to-banks relation, why random?

Reasoning 2: Page frames to banks relation

■ Analyzing program

- ✓ To identify how page frames are mapped into banks
 - Access physical memory sequentially with the cache-line unit
 - Using data dependency with two variables to prevent the out-of-order execution in CPU and scheduling effect of memory controller
 - Measure the access latency

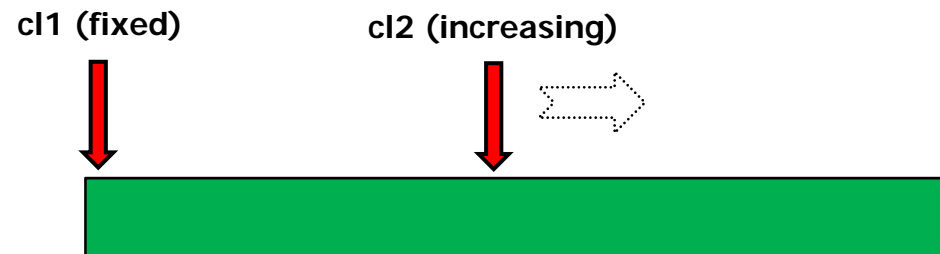
```
1: procedure ANALYZER_CORE(start_of_mem, end_of_mem)
2:   *cl_1 ← First cache line from start_of_mem
3:   *cl_2 ← Second cache line from start_of_mem
4:   Set uncached mode from start_of_mem to end_of_mem

5: repeat // iterate through all test memory area
6:   Start stopwatch
7:   i = 0;

8:   while i++ < 500 do
9:     *cl_1 ← *cl_2 + magic_number_1;
10:    *cl_2 ← *cl_1 + magic_number_2;
11:    //To preventing out-of-order execution
    in CPU and Memory controller
12:   end while

13:   Stop stopwatch
14:   cl_2 ← Next cache line
15:   until cl_2 > end_of_memory
16: end procedure
```

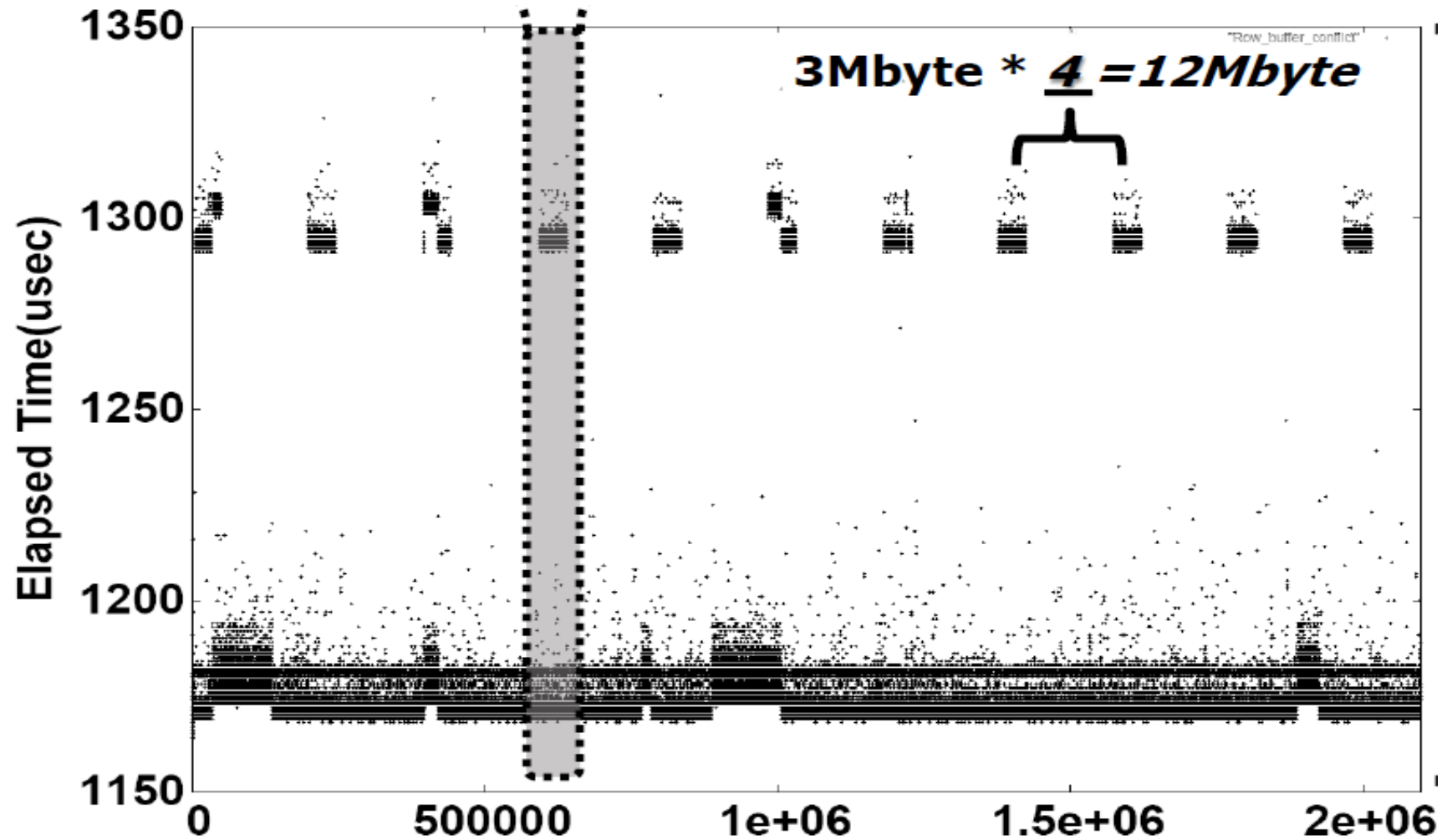
Pseudo code



Reference pattern

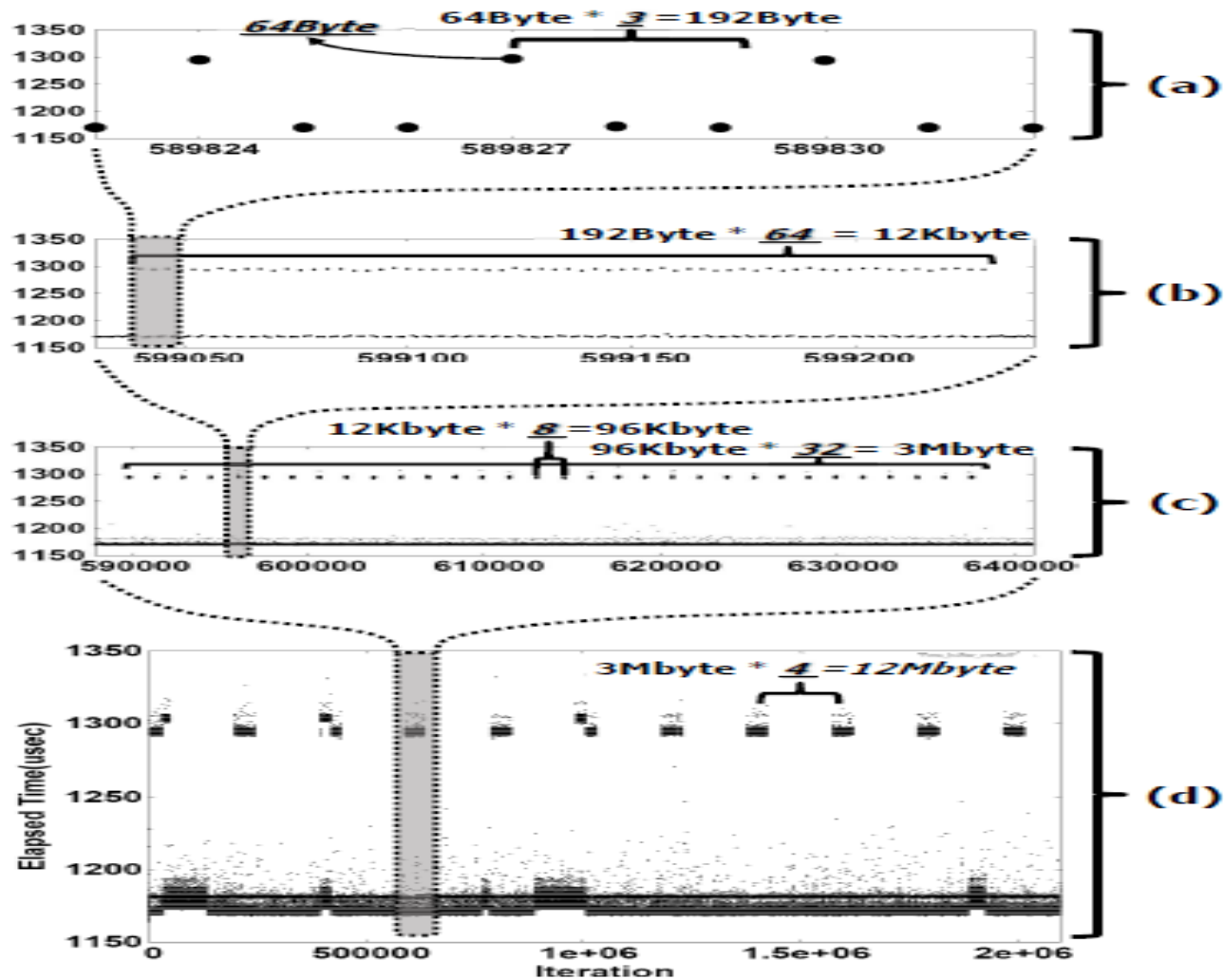
Reasoning 2: Page frames to banks relation

- Measurement



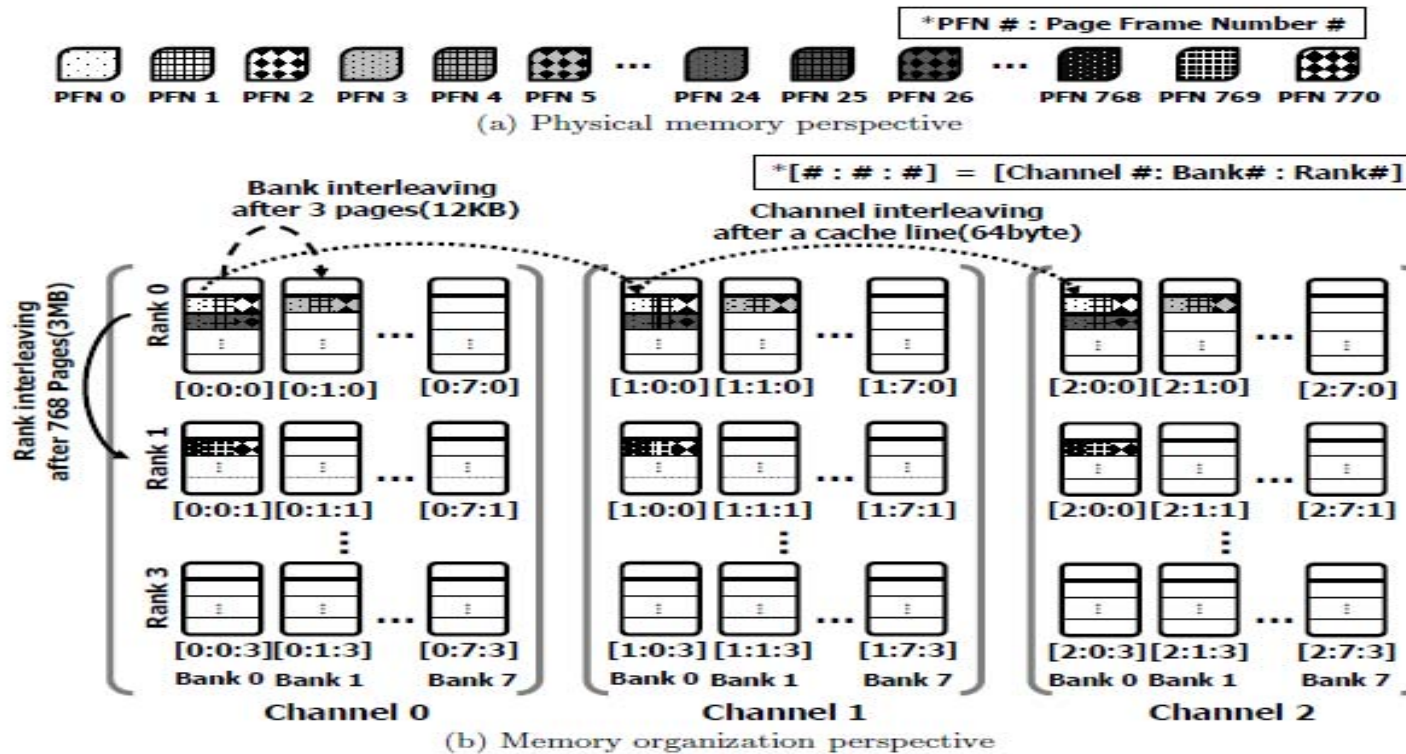
Reasoning 2: Page frames to banks relation

- Analysis results



Reasoning 2: Page frames to banks relation

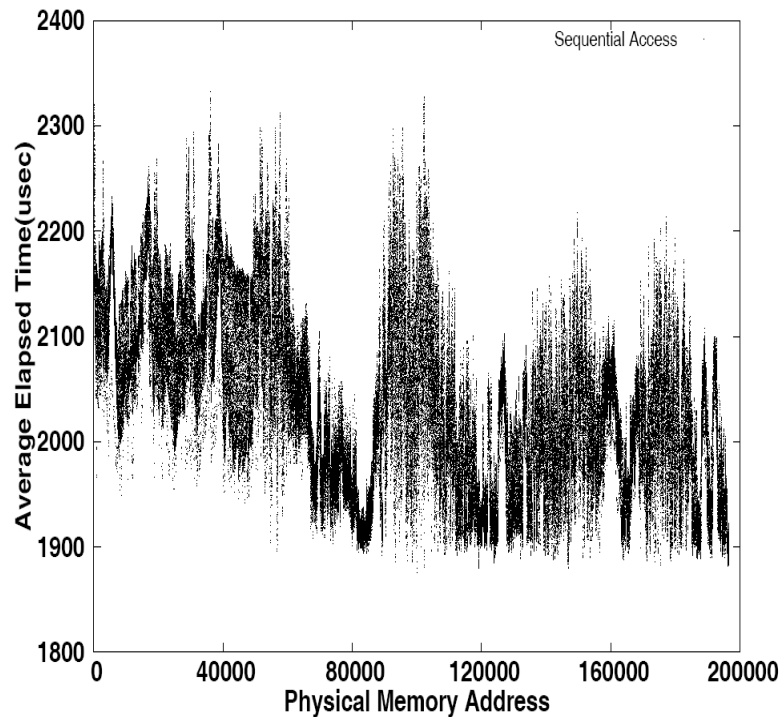
■ Analysis results



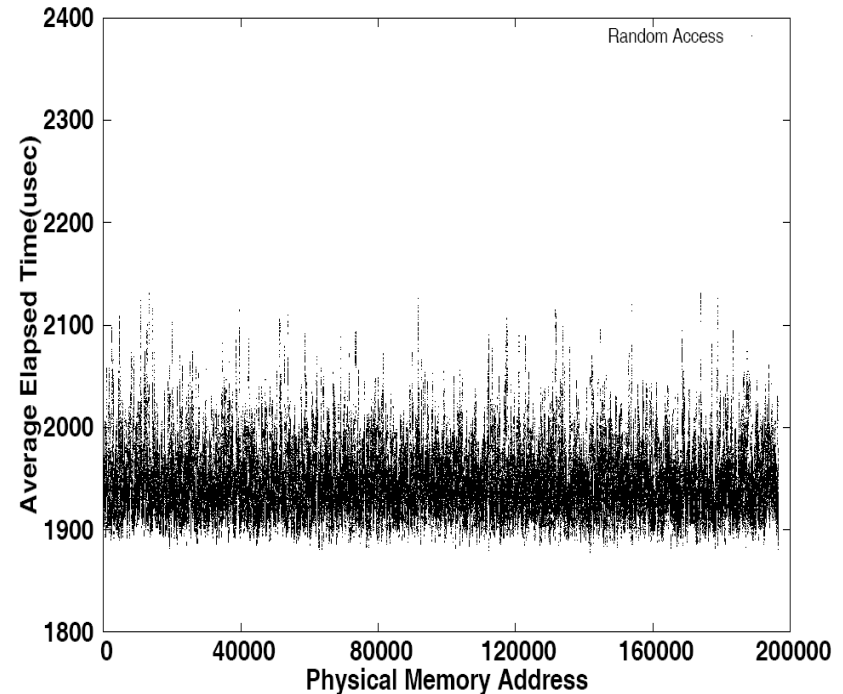
- ✓ A page frame is mapped into 3 banks (channel interleaving with CPU cache line size)
- ✓ Bank interleaving with 12KB size ($12\text{KB} * 8 = 96\text{KB}$)
- ✓ Rank interleaving with 3MB size ($96\text{KB} * 32 = 3\text{MB}$)
- ✓ Round-robin fashion with 12MB ($3\text{MB} * 4$)

Reasoning 3: Allocation sequence

- Sequential or random reference pattern test
 - ✓ Modify the analyzing program
 - Access one variable (cl1) only with sequentially or randomly
 - Run on 4 cores at the same time (different 12MB memory area)



(a) Sequential access



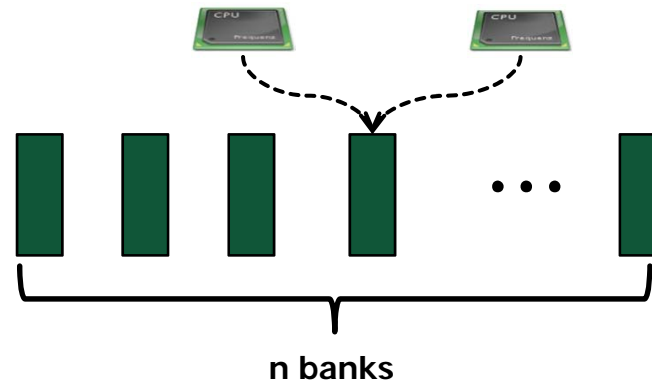
(b) Random access

Reasoning 3: Allocation sequence

■ Modeling (in progress)

✓ Conflict probability

- Random: $1/n$
- Sequential: $\{1/n * n + (n-1)/n * 0\} / n = 1/n$



✓ Correlated Conflict

- A set of successive conflicts
- Incur different progress ratio (theory of relativity)
- Amplify conflicts among multiple cores



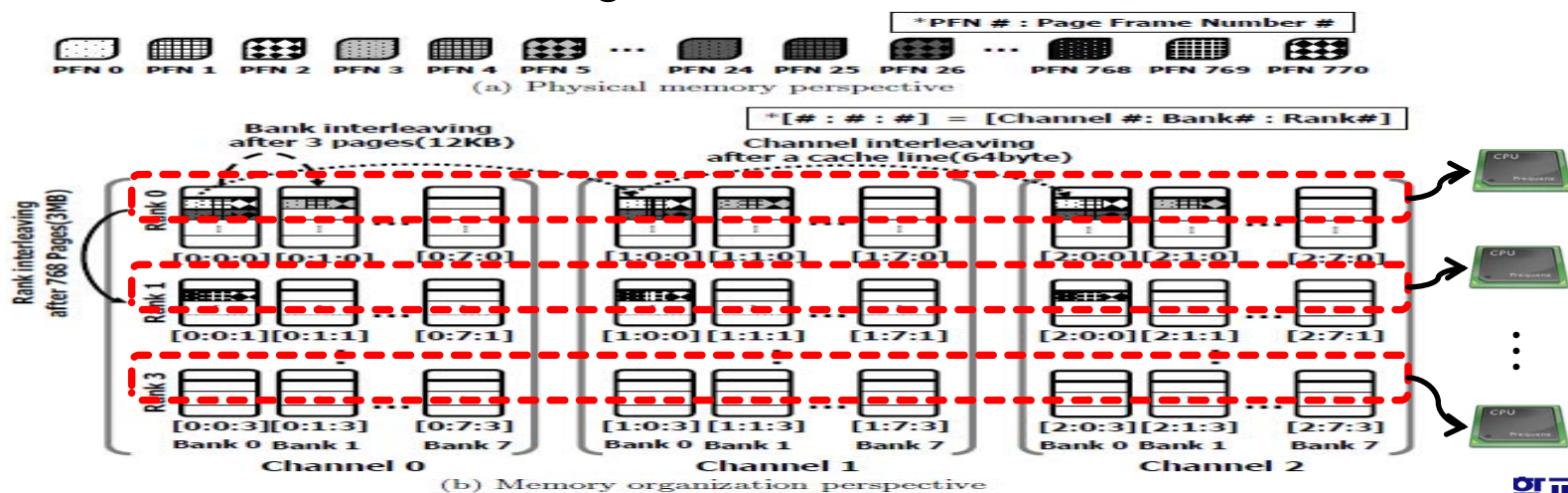
(a) Random access



(b) Sequential access

Lessons

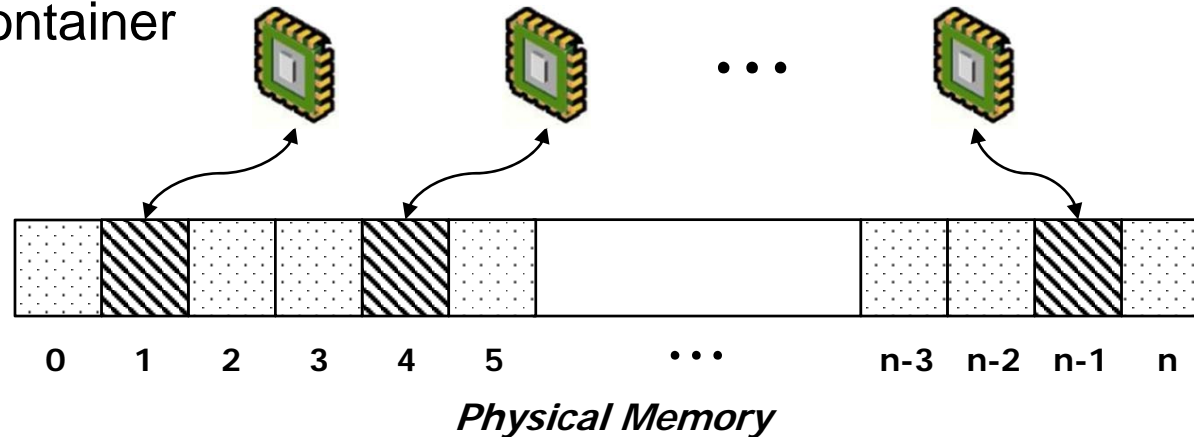
- Page frame access latency is not same (even in UMA)
 - ✓ Row-buffer conflict
- OS page frame allocation policy
 - ✓ Can affect the row-buffer conflict in a bank
 - ✓ Regularities (such as sequential) considered harmful
- In our experimental system
 - ✓ Allocating three successive frames obtains prefetching effects
 - ✓ 12MB (or 3MB) is a good candidate for partitioning to reduce the row-buffer conflicts among cores



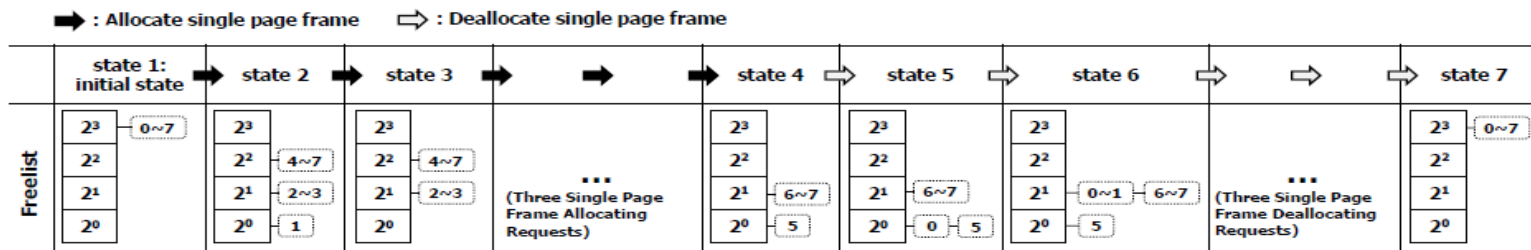
Lessons

- OS-level Bank-aware allocator (M^3 allocator)

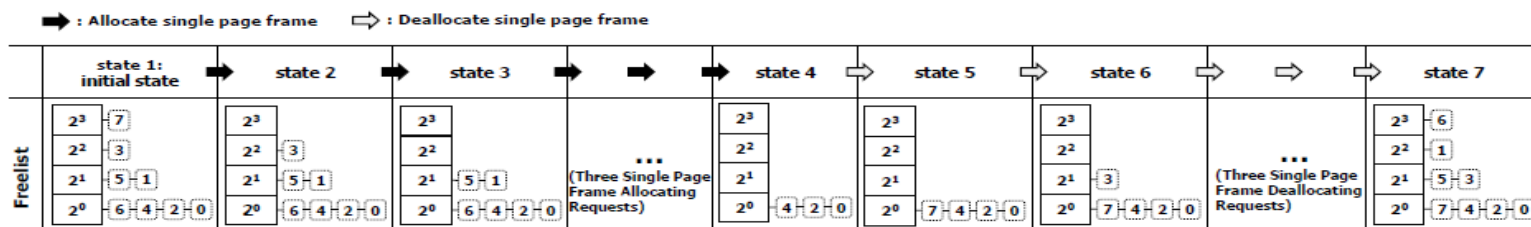
- ✓ Memory container



- ✓ Randomizing allocation



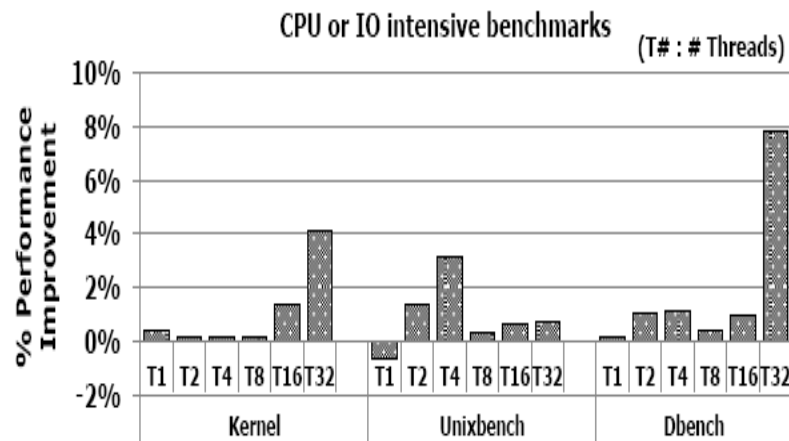
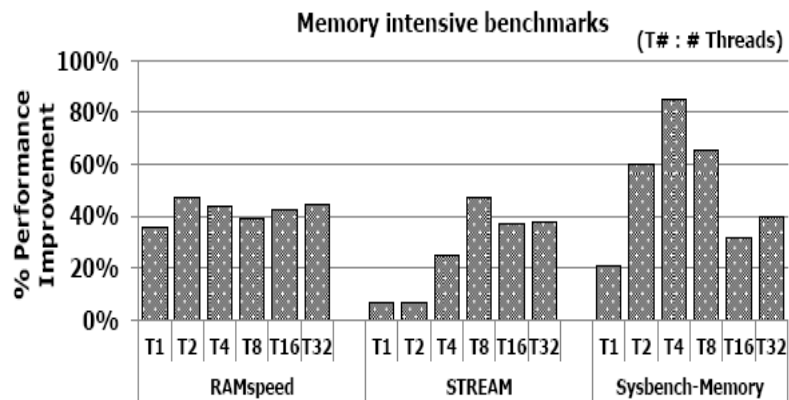
(a) Buddy algorithm



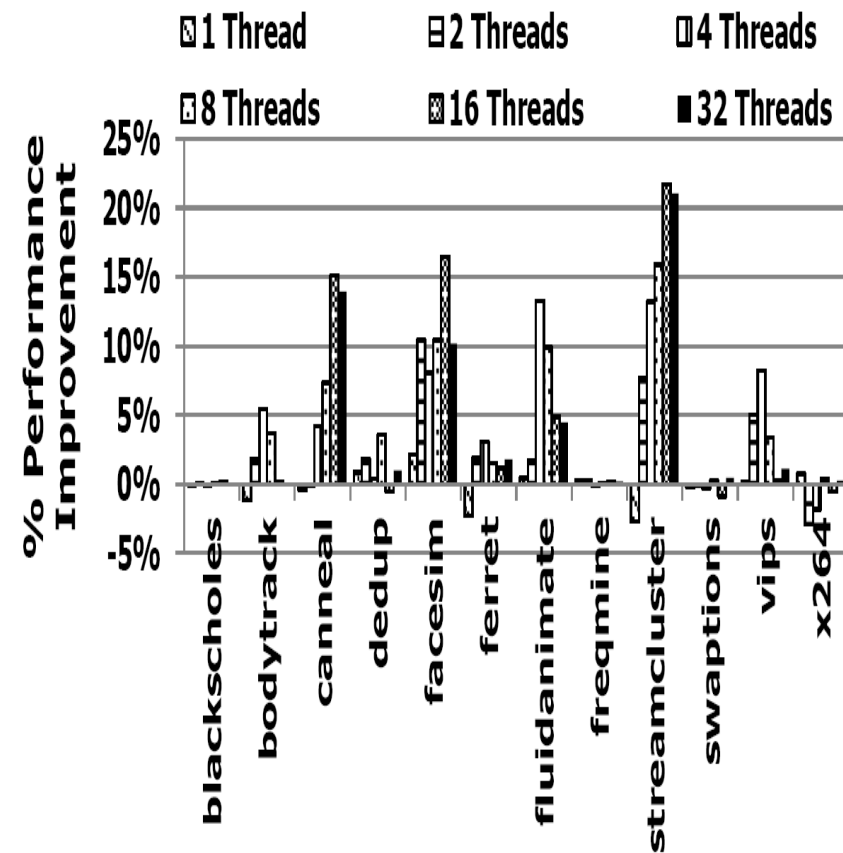
(b) Randomized algorithm

Lessons

■ Experiment results



Parsec Benchmark Suite Results



Conclusion

- Bank-Level Parallelism (BLP)
 - ✓ Low-buffer conflict
- OS-level approach
 - ✓ Page frames to banks relation
 - ✓ Sequential vs. random allocation
- New consideration
 - ✓ Virtual memory
 - ✓ Physical memory
 - ✓ Memory organization

