Watch Memory System on Real Systems

by Hardware/Software Hybrid Methods

Yungang Bao

Princeton University

Institute of **C**omputing **T**echnology (**ICT**),
Chinese Academy of Sciences

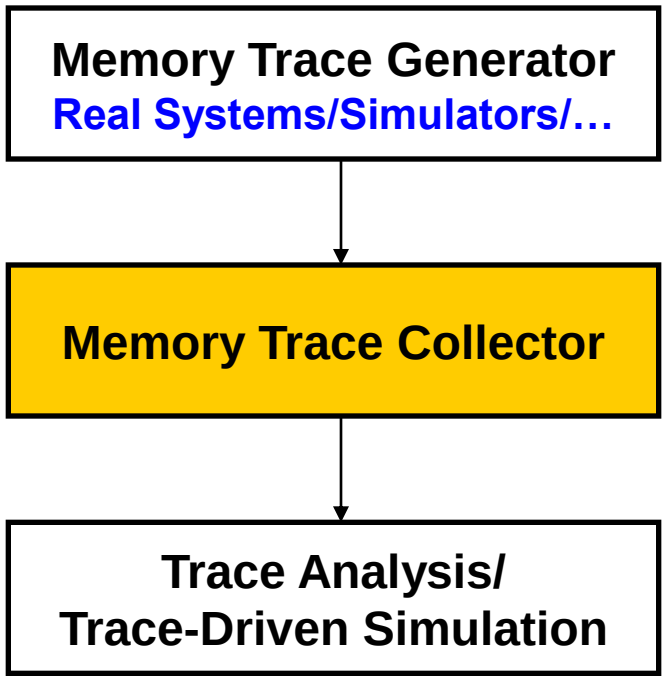
Joint work with
Mingyu Chen, Licheng Chen, Zehan Cui, Yongbing Huang, Lei Liu, Yuan Ruan and Dan Tang

Memory System: Past and Present

- Memory is the key component of computer systems, drawing lots of research efforts
- Before 1990, memory capacity was a big issue for whole system performance
- In the 1990s, memory wall was raised due to the disparity of CPU and memory speed
- We are in the multicore era where scalability, power and QoS are the new challenges

Memory System Research Process

- **Trace-driven Approaches**



- Thermal Modeling and Management of DRAM Memory Systems (ISCA'07)
  - M5 → Trace → MEMSpot
- Memory Prefetching Using Adaptive Stream Detection (Micro'06)
  - Power5+ Simulator → Trace → MemSim

Software simulation is still the dominated effective method for collecting memory trace.

However,
Voice of researchers

- We collected memory reference traces from a detailed full-system simulator. … (Each simulation still took 1-2 weeks to complete.) …

  -- Disaggregated Memory for Expansion and Sharing in Blade Servers, ISCA'09

- We used COTSon, a relatively fast simulator [2]. However, even with COTSon, tracing at this level is slow, which limits the amount of simulated execution we can achieve. For example, one of our runs, covering 213 traced seconds, took 10 days. …

  -- Operating System Support for NVM+DRAM Hybrid Main Memory, HotOS'09

- We need to evaluate the DRAM DTM schemes for at least thousands of seconds. Direct cycle-accurate simulation for studying DRAM thermal management is almost infeasible at this time length ….

  -- Thermal Modeling and Management of DRAM Memory Systems, ISCA'07

An Ideal Memory Trace Collector

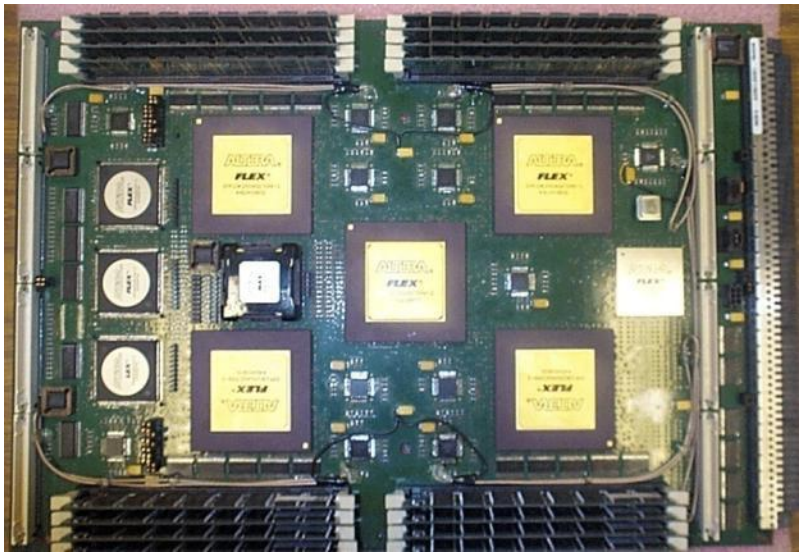
- Richard Uhlig (Intel), Trevor Mudge (UMich).

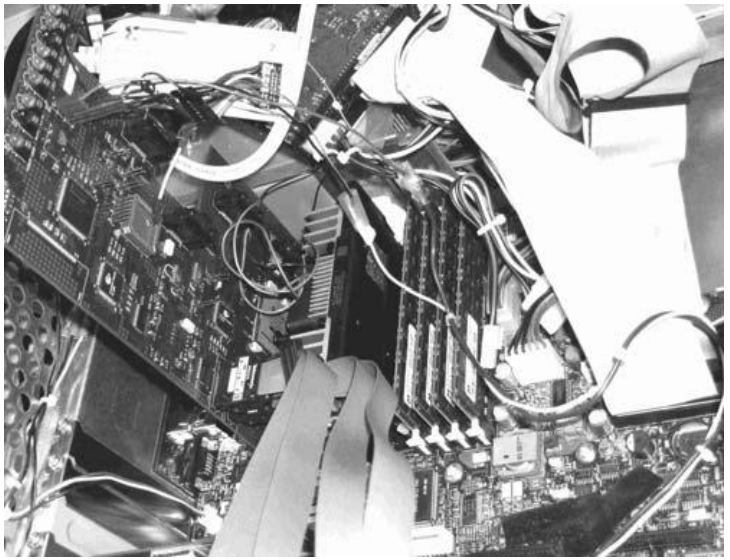
  Trace-driven memory simulation: A Survey. *ACM Computing Surveys*, Jun., 1997

- An Ideal Memory Trace Collector should be
  - **Fast** (without waiting weeks)
  - **Complete** (App, OS, Lib, etc.)
  - **Detail** (high-level information, e.g., Virt, Pid)
  - **Undistorted** (No Time Dilation or Memory Dilation)
  - Others
    - **Portability, Inexpensive, Easy to operate**

Hardware-based Method

IBM's MemorIES@ASPLOS'00

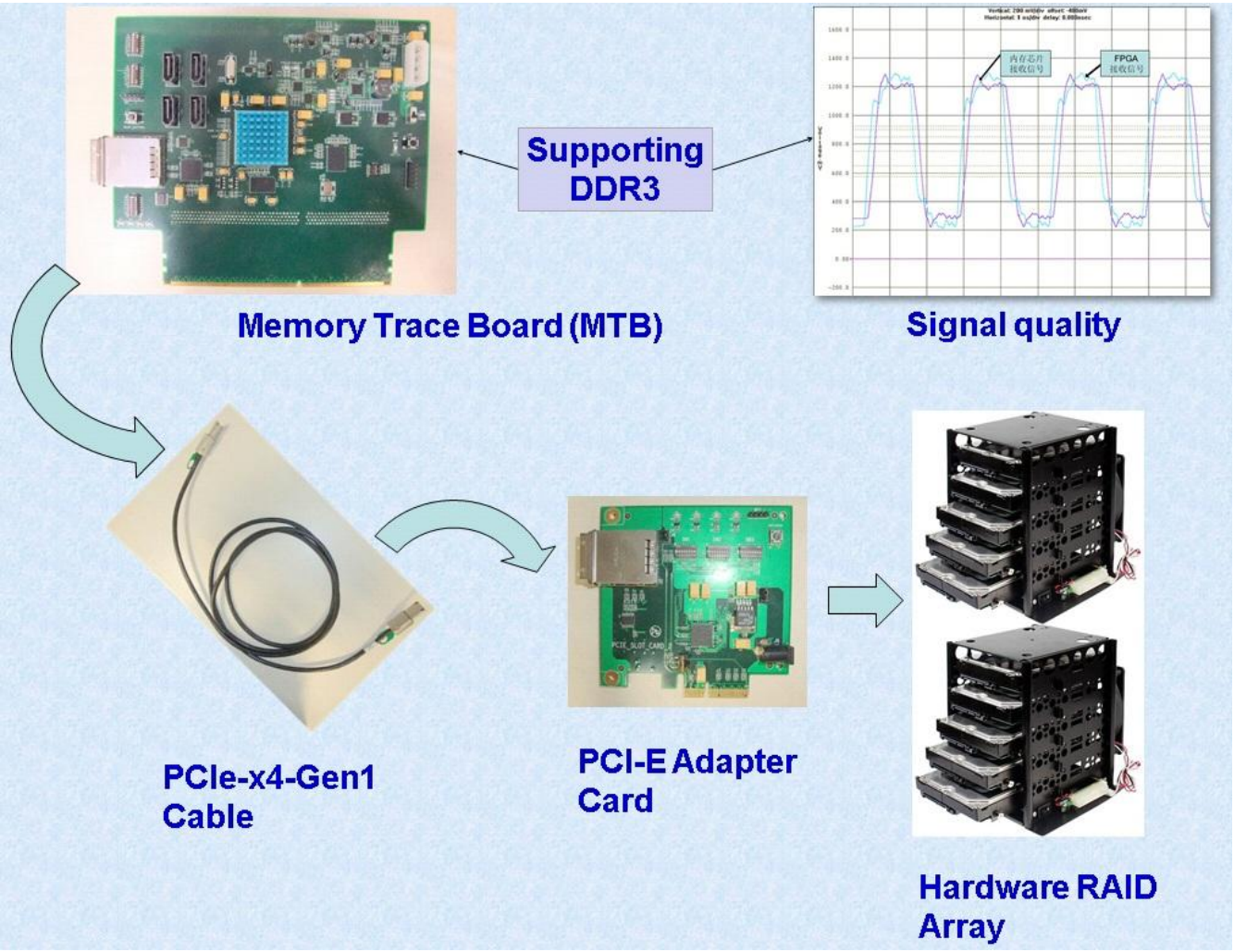




Intel's ACE@FPGA'06

- Collect memory traces of whole systems, but most have no idea of high-level information
- Allow programs to run at native speed, but difficult to deal with huge memory trace
- Some old tools (20 years ago) can get high-level information, but with slow memory speed and poor portability.

HMTT: A Hardware/Software Hybrid Memory Trace Tool



- ✓ Fast
- ✓ Complete
- ✓ Detailed
- ✓ Undistorted
- ✓ Portable
- ✓ Inexpensive

Studying Memory System on Real Systems by HMTT

- **Principle/Design of HMTT** (*Sigmetrics'08*)
- **Derivatives Tools of HMTT**
  - Object-Relative Memory Profiling (*ISPASS'12*)
  - Low-overhead Lock Profiling (*PACT'12*)
  - Fine-Granularity Memory Power Profiling (*PMP'11*)
- **Leveraging HMTT to Optimize Memory System**
  - DMA Characteristics and DMA Cache (*HPCA'10*)
  - Page-Coloring based Bank-level Partition (*PACT'12*)

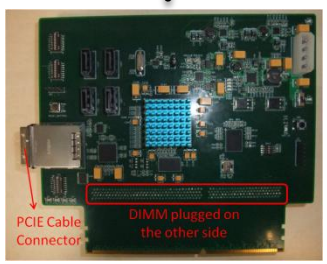
Agenda

- **Principle/Design of HMTT**
- **Derivatives Tools of HMTT**
  - Object-Relative Memory Profiling
- **Leveraging HMTT to Optimize Memory System**
  - Page-Coloring based Bank-level Partition
- **Conclusion**

An Overview of HMTT

Physical Address Trace

0x398f24a
0x398f24b
0x398f24c
……
0x1af4aa
0x1af4a6
0x1af4a8
……
0x38d2cfc
0x38d2cfd
……

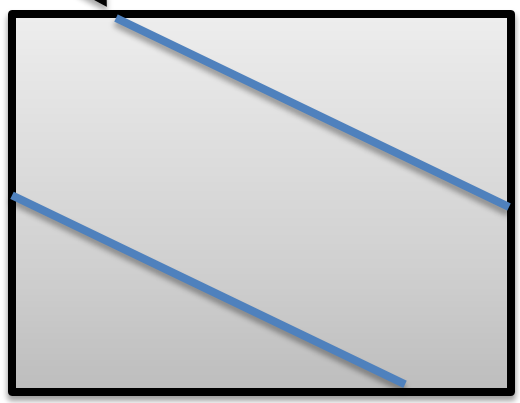


Virtual Address Trace

0x1f05000
0x1f06000
0x1f07000
……
0x1f15000
0x1f16000
0x1f17000
……
0x1f25000
0x1f26000
……

Object Access Pattern

Matrix (VA: 0x1f05000)



Linux

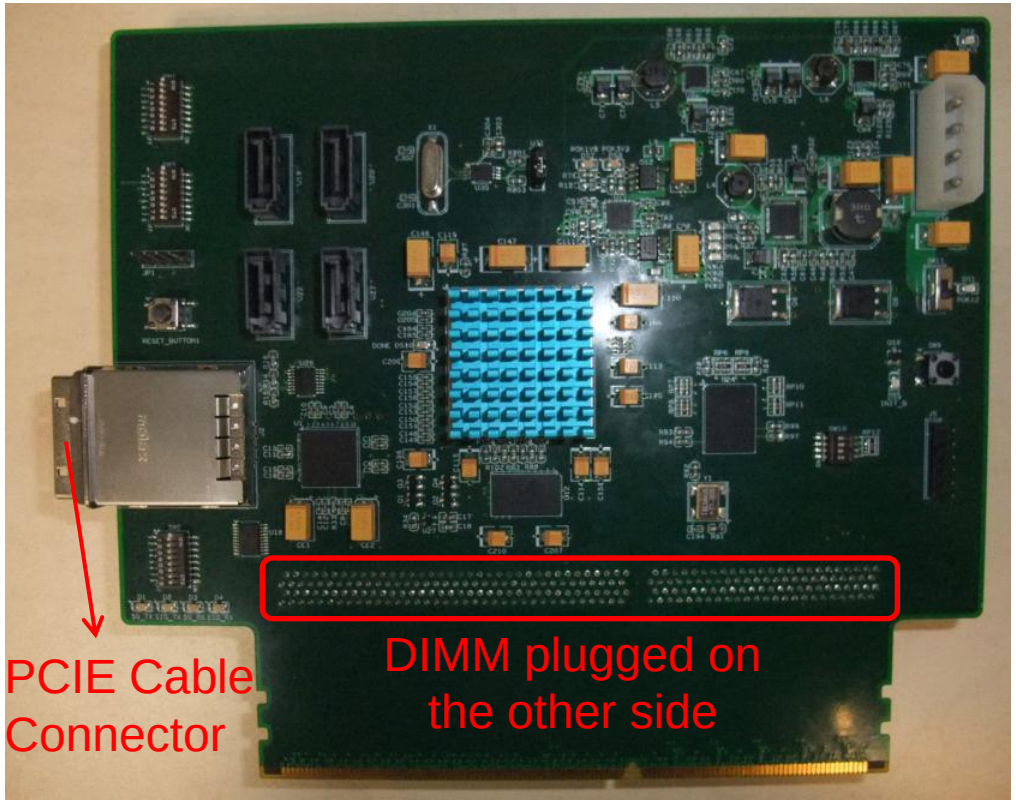
Hardware Component

- Plugged in memory DIMM slots
  - Snooping signals on memory bus
- Support DDR3-800 DRAM compatible
- Uses PCI-e to transfer memory trace to remote machines

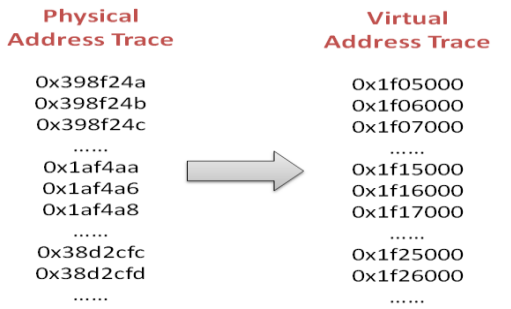
Memory Trace:
<time_stamp, r/w, phy_addr>

Advantages:

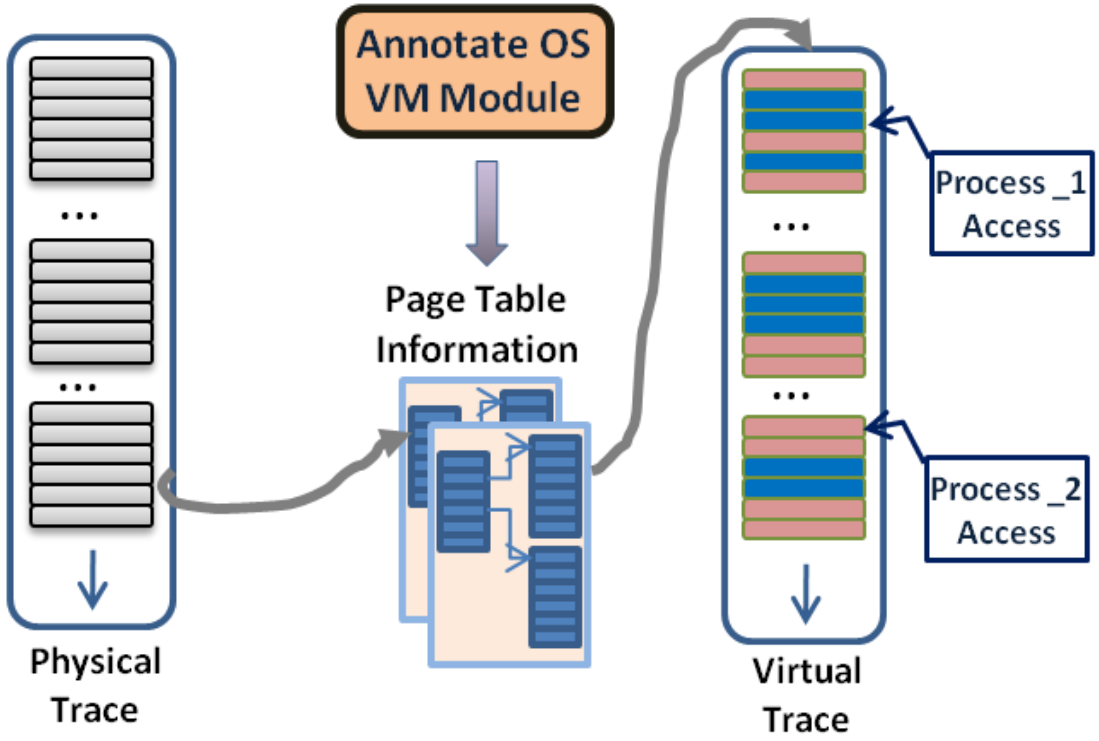
- Platform independent
- Negligible overhead
- Full-system memory traces, including OS, page table walks
- Support storing large-scale trace



Software Challenges (1)



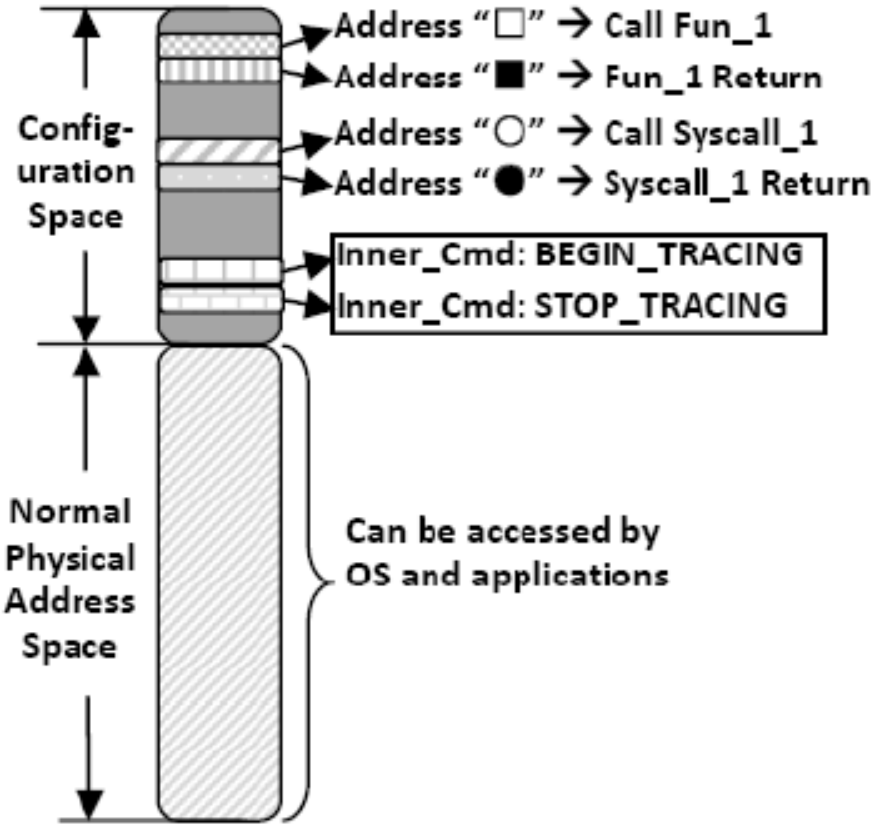
- **Physical address → High-Level Event**
- How to translate physical address to virtual address of a specific process?



- Modify OS kernel to obtain page table
- Lookup a phy_addr in the dumped page table
- Generate virtual trace of each process

Synchronization Challenge (2)

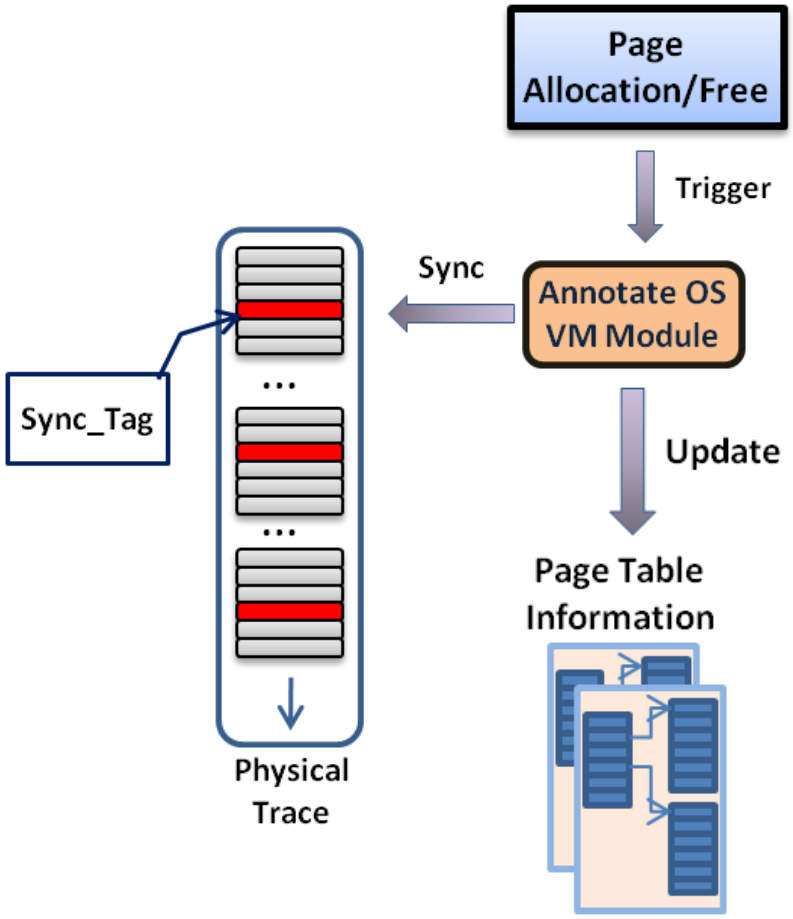
- How to synchronize hardware and software when a page table update occurs in OS kernel?



- **High-Level Event Encoding Mechanism (HLE2M)**

1. Reserve memory space
2. Map HL-events to the space
3. Collect the accesses to the space
4. Translate the accesses into the events offline

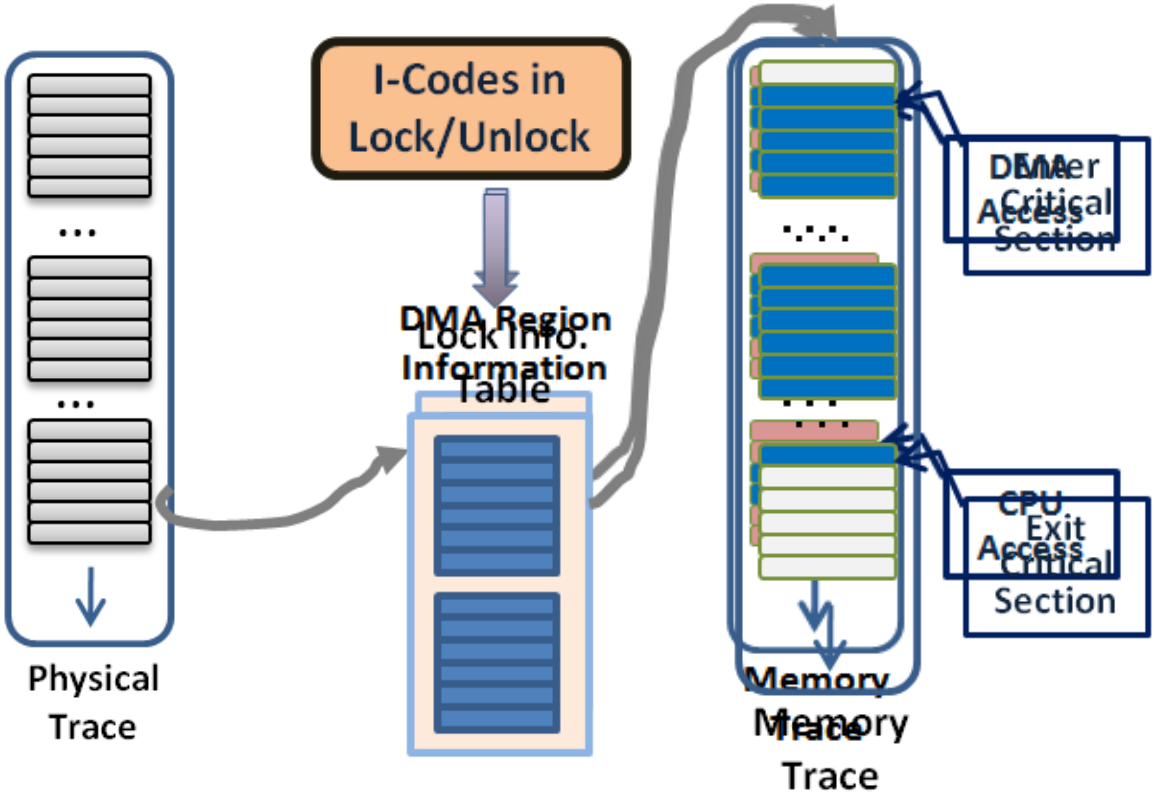
Detail: Page Table Update Event



- Upon each physical page allocation/Free in kernel
  - Trigger annotated codes in OS VM module
  - Update dumped page table
  - Send a sync_tag to hardware by issuing a specific access to the red address

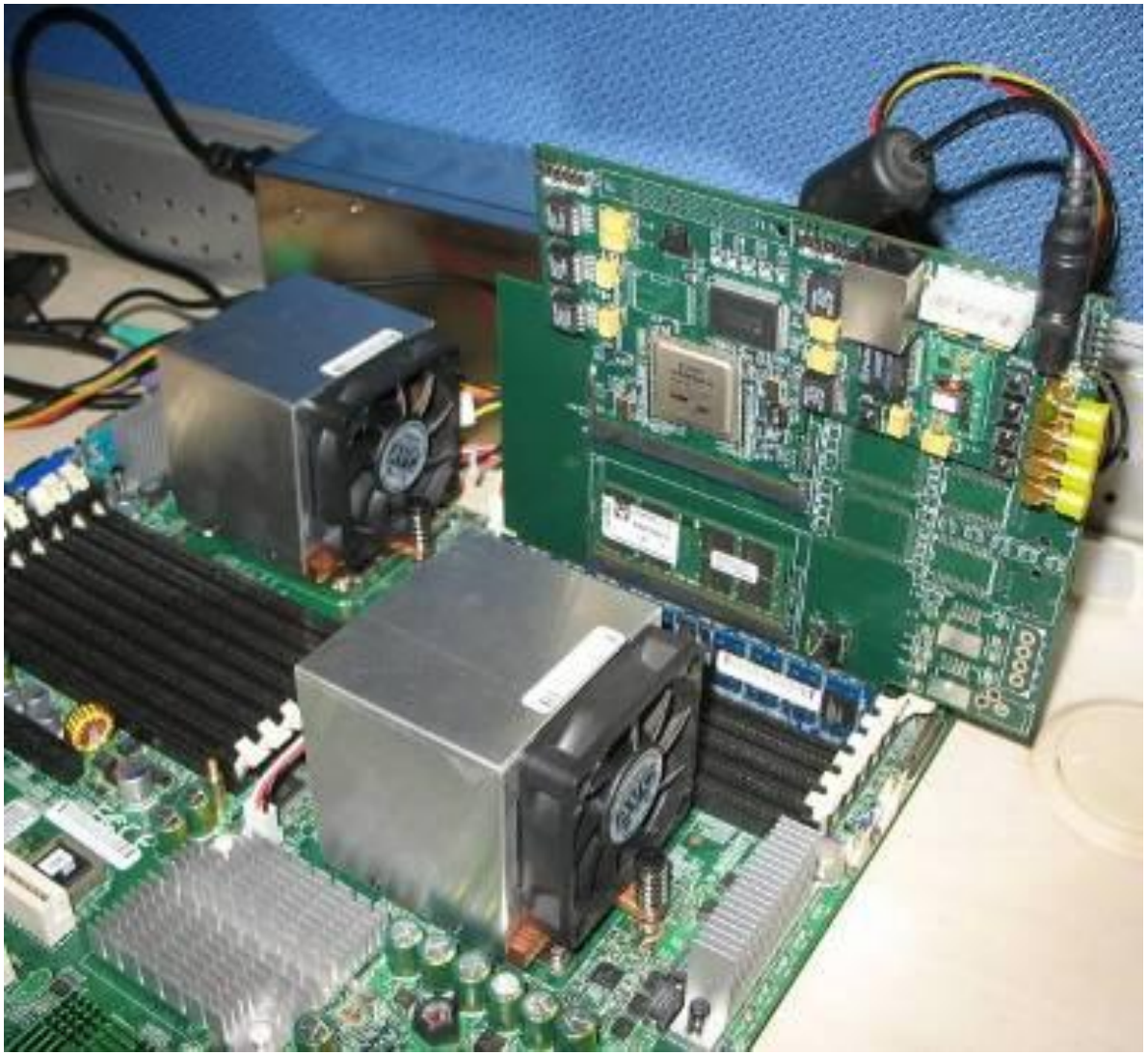
More High-Level Information

High-Level Event Encoding Mechanism (**HLE2M**) enables more useful scenarios.



- CPU/DMA Access
- Lock Profiling

HMTT Prototype



Agenda

- **Principle/Design of HMTT**
- **Derivatives Tools of HMTT**
  - Object-Relative Memory Profiling
- **Leveraging HMTT to Optimize Memory System**
  - Page-Coloring based Bank-level Partition
- **Conclusion**

Memory Profiling

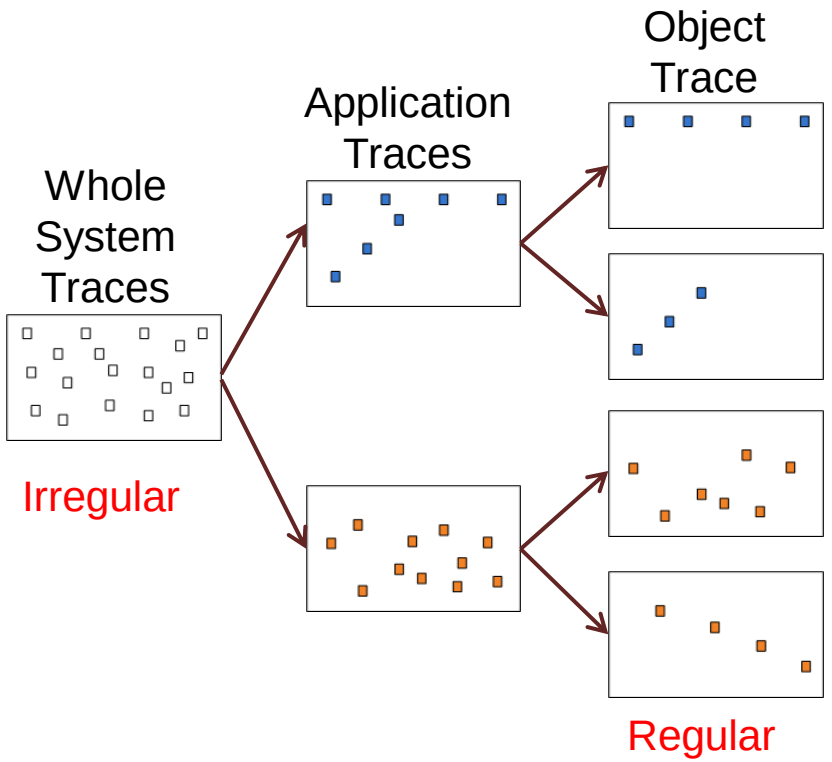
- Memory profiling is to collect memory behavior information during the execution of programs.
- Profiling can be performed for
  - different hardware components
  - different software levels

Function
Application
Whole System

Objects (Array, List etc.)

Object Memory Profiling

- Object refers to a group of data stored as a unit [Wu'04]
  - Distinguish regular patterns from mixed and irregular traces
- Valuable for optimization
  - Memory trace compression
  - Data layout
  - Object-level prefetching
  - Cache partition [**Soft-OLP, PACT 2009**]

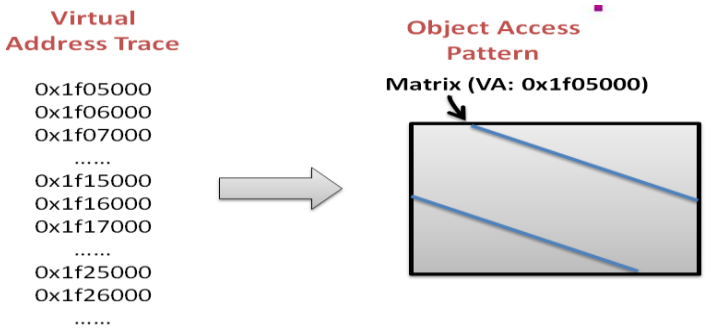


Current Profiling Approaches

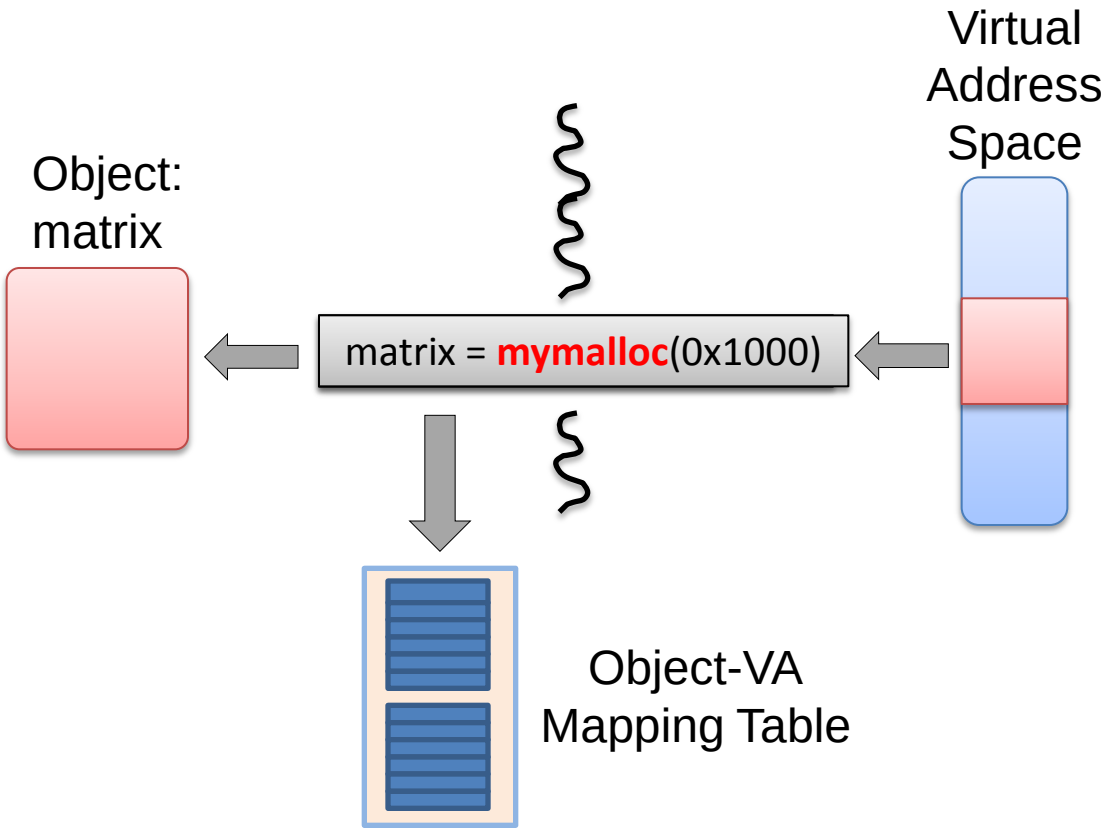
- Existing approaches
  - Compiler-driven: re-compile/re-link, source code
  - Instrumentation: heavy overhead
  - Simulation: accuracy problem, slow
  - Performance Counter: lack of detailed traces

- We enhance HMTT to support object memory profiling
  - Accurate: real application & real system
  - Lightweight
  - Collect page table walks at object-level

Translation Challenge



- How to translate virtual address to objects?



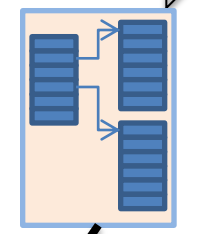
- The role of malloc() is to map VA to object
- Use dynamic library overwrite to replace malloc()

Put them all together

Physical Address Trace

0x398f24a
0x398f24b
0x398f24c
sync_tag
page walk
0x1af4a6
0x1af4a8
sync_tag
0x38d2cfc
0x38d2cfd
page walk

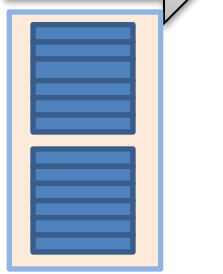
Dumped Page Table



Virtual Address Trace

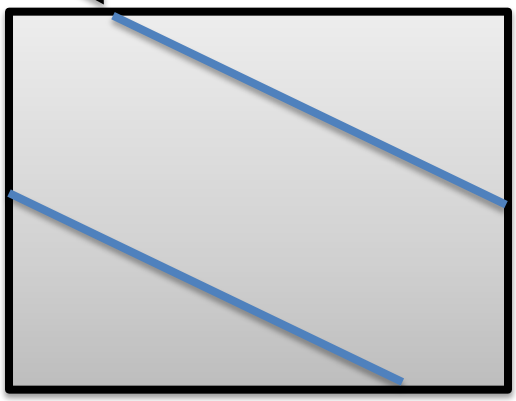
0x1f05000
0x1f06000
0x1f07000
……
0x1f15000
0x1f16000
0x1f17000
……
0x1f25000
0x1f26000
……

Object-VA Mapping Table



Object Access Pattern

Matrix (VA: 0x1f05000)



Use page table to distinguish three types of memory access

- Sync_tag → update page table
- Access page table itself → page table walk due to TLB miss
- Other memory access → virtual address

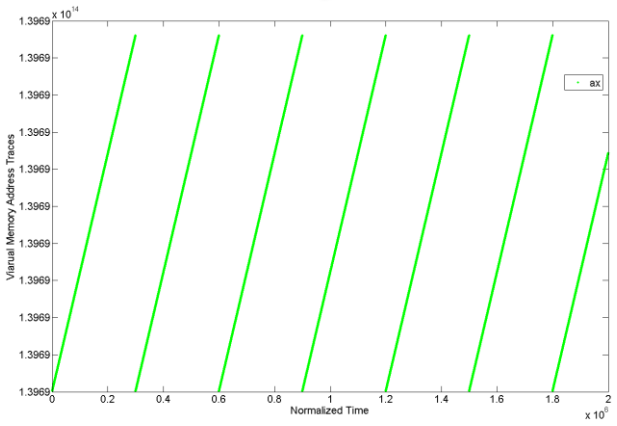
Evaluation Methodology

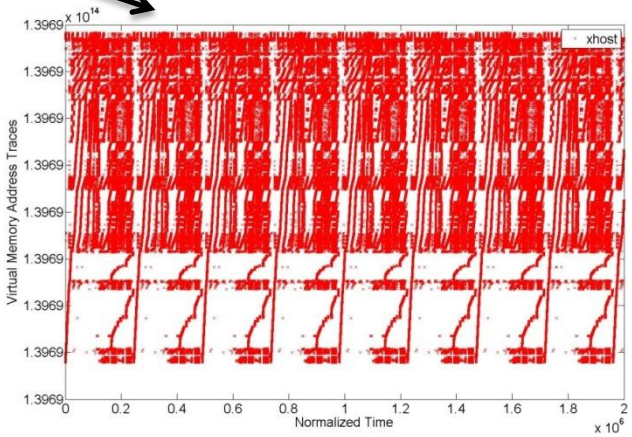
Processor		Intel Xeon E5504, 2.0GHz, 2 Sockets, 4 Cores per Socket **(8 core in total)**
Private Cache	L1	D-Cache: 32KB, 8-way, 64Byte/line I-Cache: 32KB, 4-way, 64Byte/Line
	L2	256KB, 8-way, 64Byte/line
Shared Cache	L3	4MB, 16-way, 64Byte/line
TLB (private)	DTLB0	64 entries for 4-KByte pages 32 entries for huge pages (2MByte)
	TLB1	512 entries for 4-KByte pages
Memory		**DDR3-800** RDIMM, dual-rank, plugged into Socket 0, 4GB 0.25GB reserved for HMTT configuration and buffer 3.75GB system available
Operating System		CentOS 5.3, Linux kernel 2.6.32.18
Benchmarks		Multithreaded PARSEC 2.1 A custom hybrid MPI/pthread implemented BFS of Graph500-1.2

Validation

- For SpMV benchmark (CSR) :

y = ax * xhost



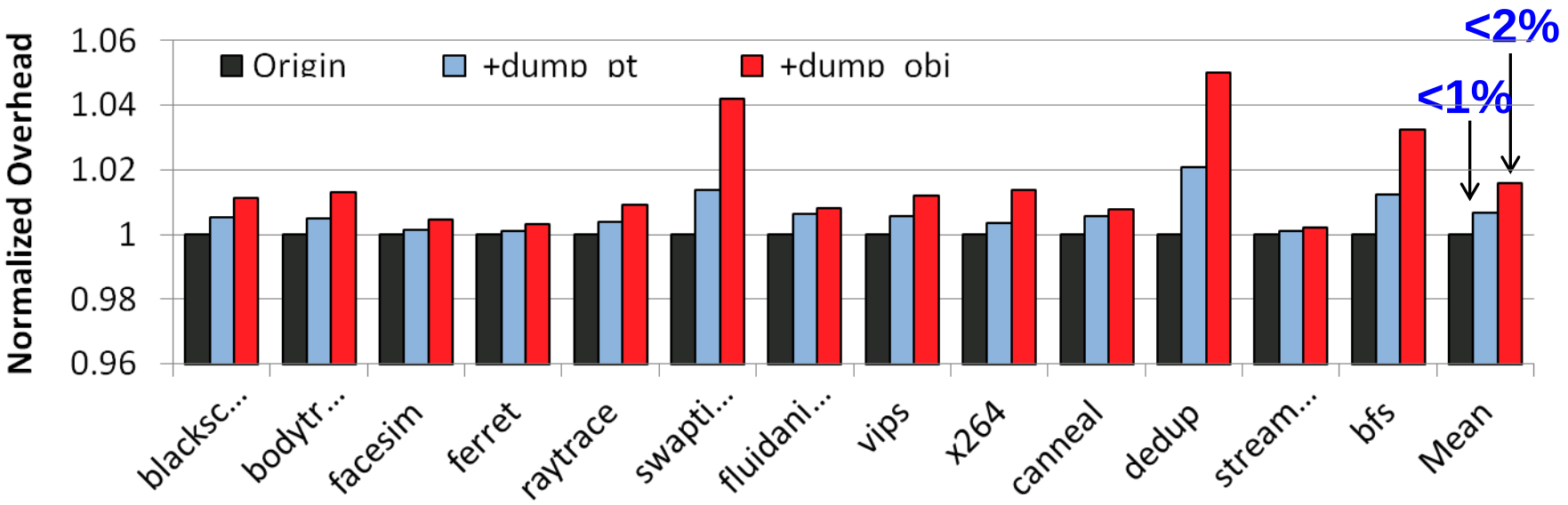


Our system is able to distinguish regular access pattern from irregular pattern

- Micro-benchmark:
  - —The error is less than 2%

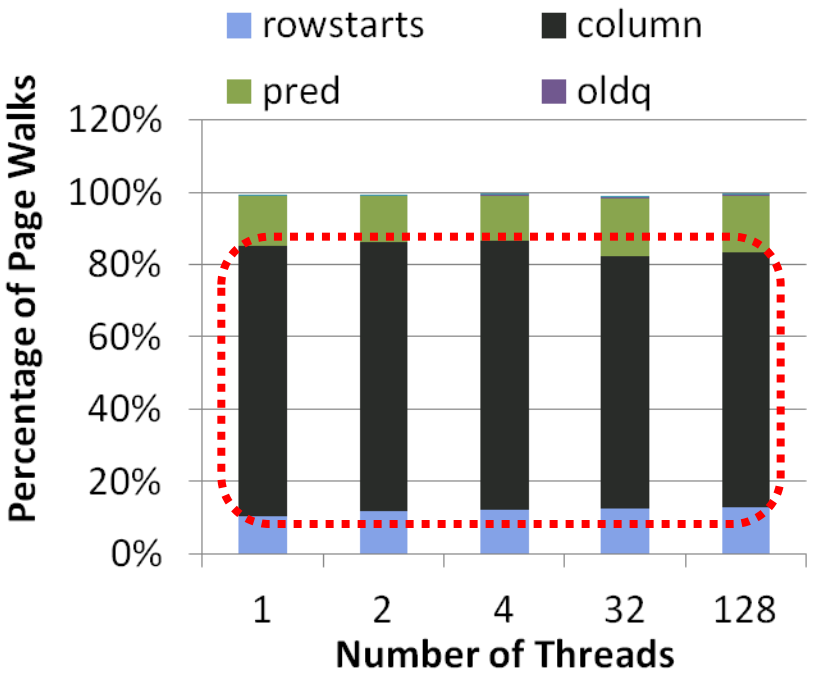
Overhead

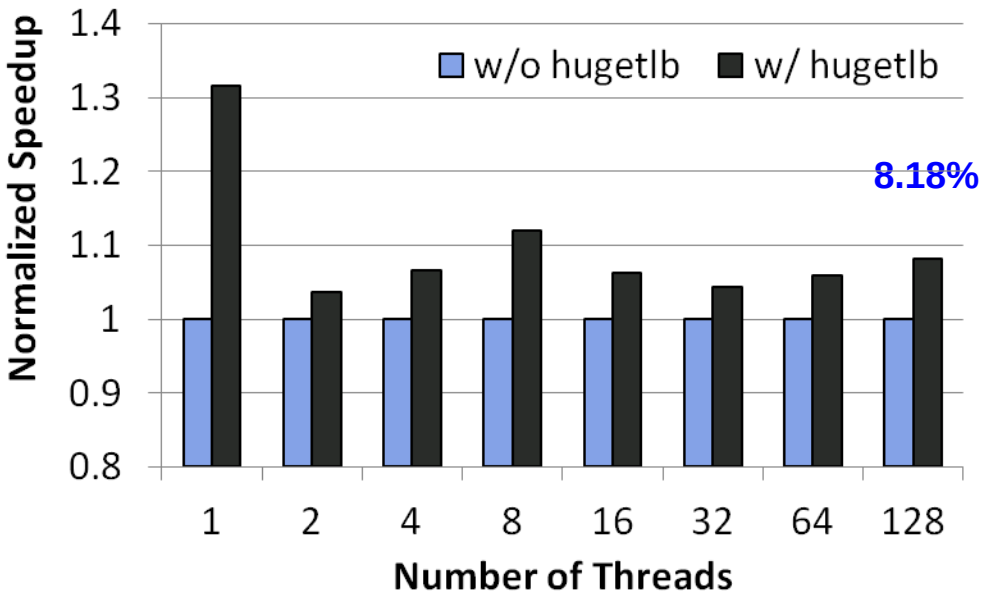
- Two main overhead:
  - Dumping page table traces: + dump_pt
  - Dumping object-VA mapping: + dump_obj
    - Monitoring objects >= 4KB: result in most memory references



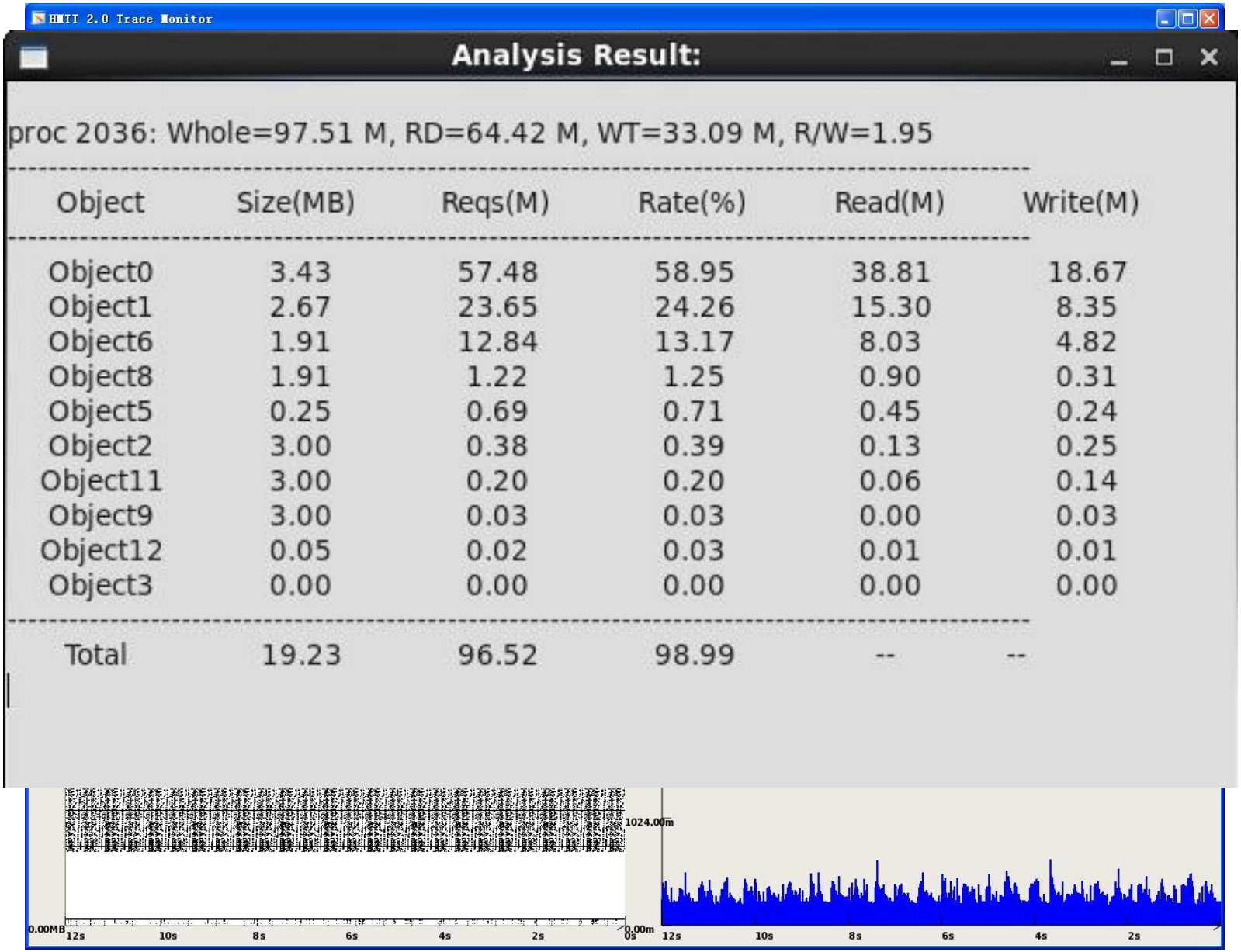
Case Study: BFS (Breadth-First Search)

- *column* object : about 71% of page walks → key object
- Optimization: use huge page for *column* object
  - Speedup: about 12% for 8-thread, 8% for 128-thread





A Visual Demo of Object Profiling



Agenda

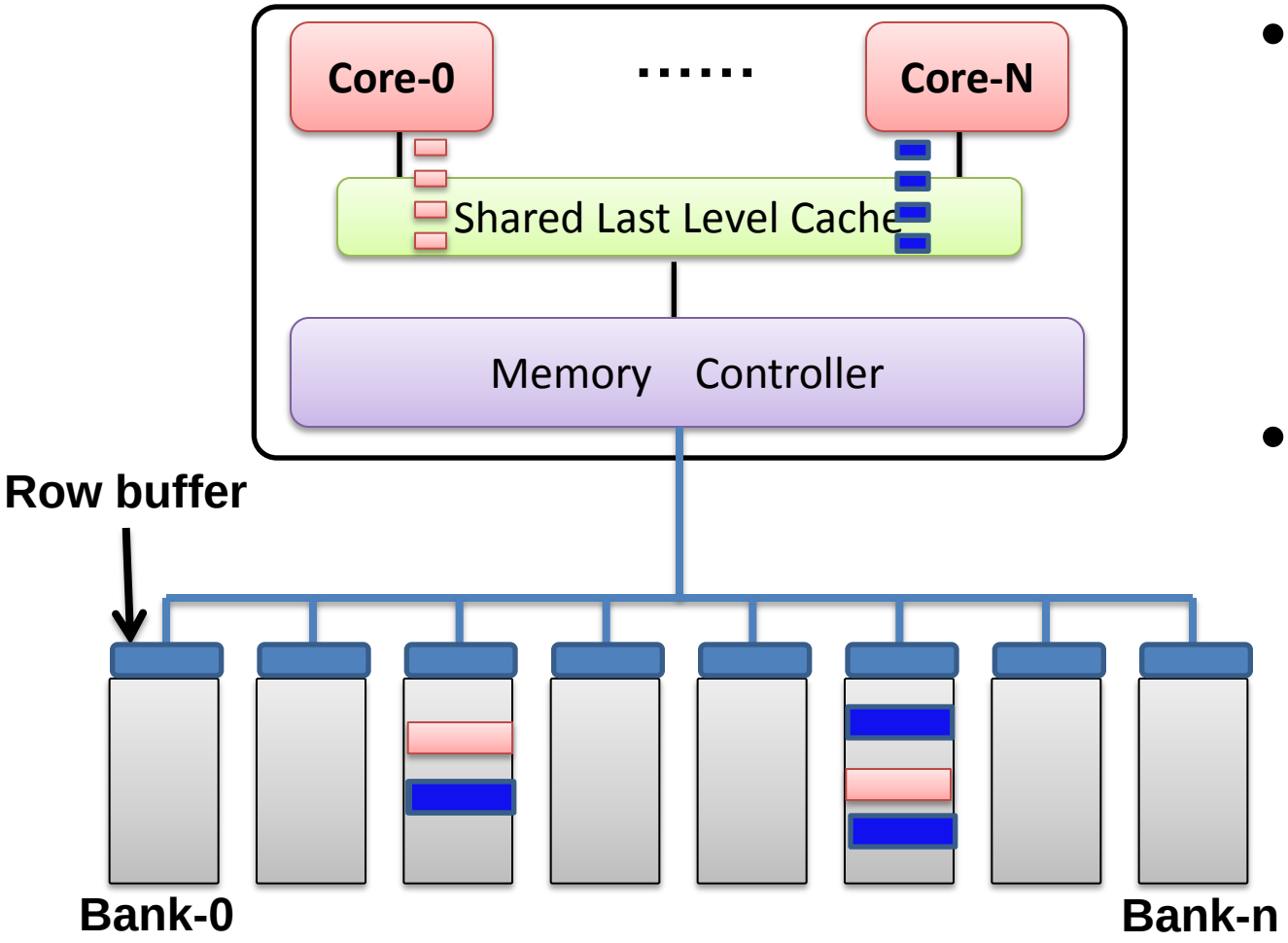
- **Principle/Design of HMTT**
- **Derivatives Tools of HMTT**
  - Object-Relative Memory Profiling
- **Leveraging HMTT to Optimize Memory System**
  - Page-Coloring based Bank-level Partition
- **Conclusion**

Multicore-Posed Challenges

- DRAM system is a shared resource in modern multicore machines.

- The shared DRAM system becomes the major bottleneck of multicore scalability due to two reasons:
  - **Interference**
  - **Unfairness**

DRAM Organization



- **Interference**
  - Row buffer Miss
  - **65% (16cores)**
- **Unfairness**
  - Favor sequential pattern
  - **7.7X vs 1.1X**

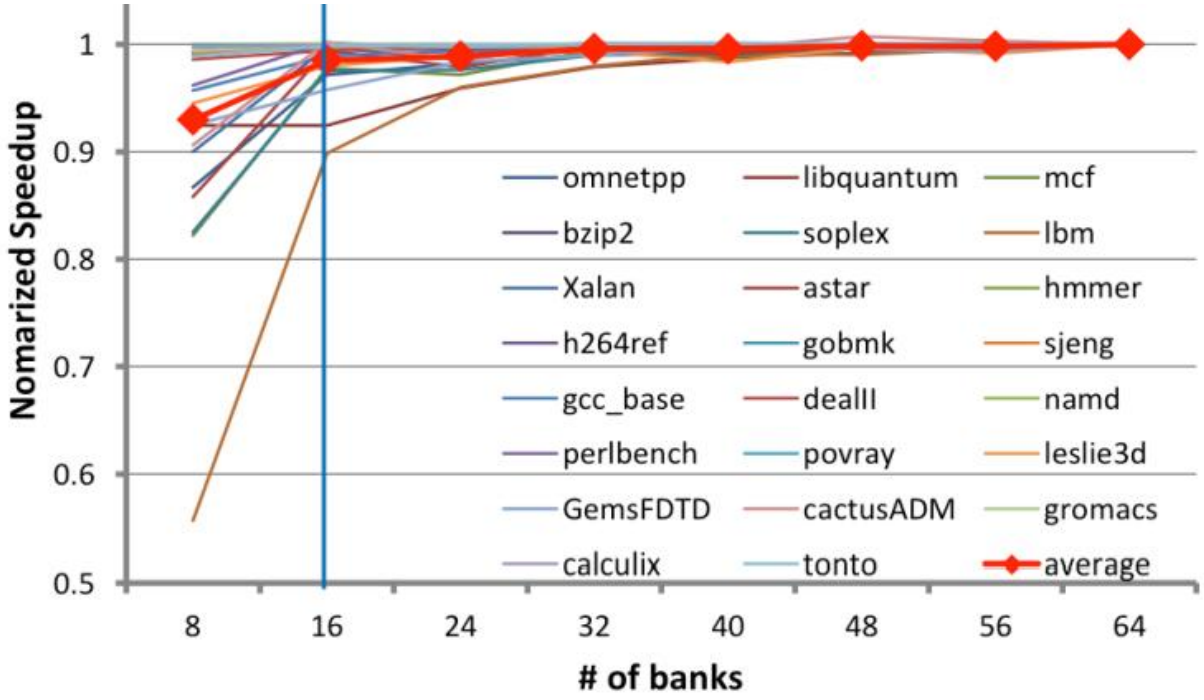
Current Solutions

- Most previous studies focus on memory scheduling algorithms.
- Few researchers realize the phenomenon that **all memory banks are shared by all CPU cores**
  - Interleaved address mapping policy to explore bank-level parallelism
  - Make consecutive memory space across multiple banks

The inter-thread bank-level conflicts can be fully eliminated by exclusively mapping a thread's data to specific banks

The Impact of Bank Amount

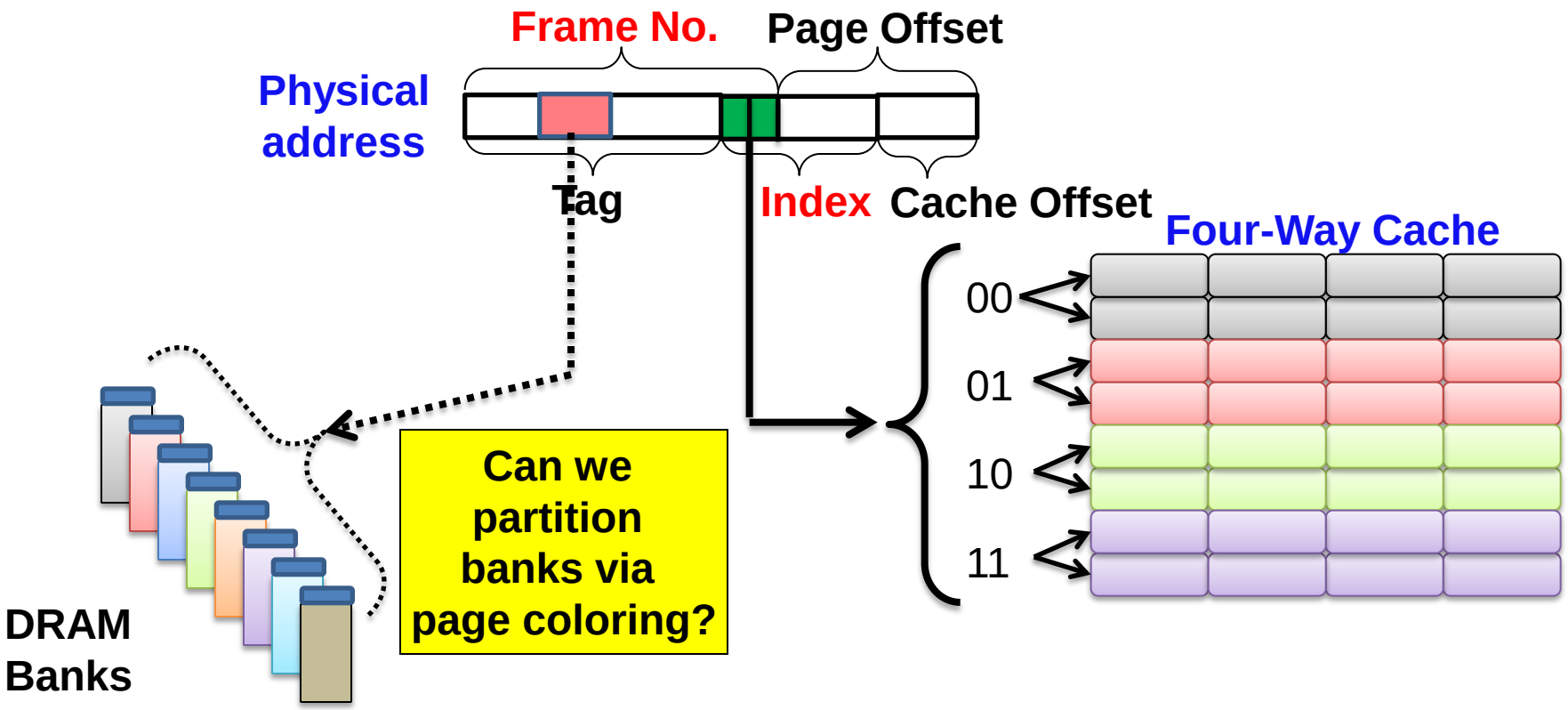
- Partition bank would reduce the available bank amount for one thread.
- Will this influence performance?



The necessary amount of banks one program requires is limited

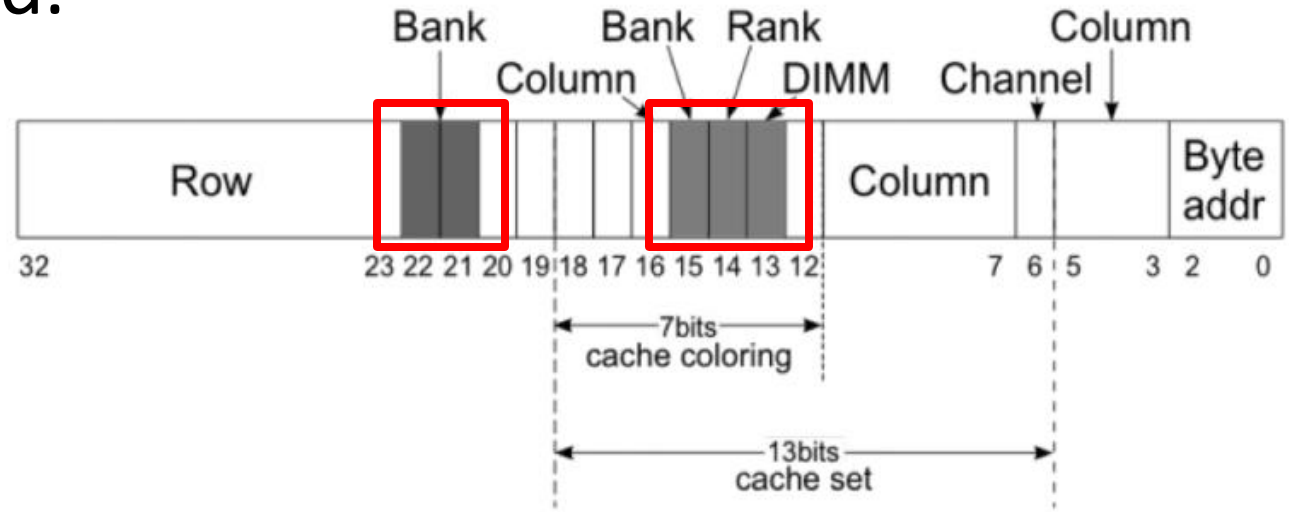
Page-Coloring Partitioning Approach

- Page coloring technique has been proposed to partition cache.



Address Mapping Challenges

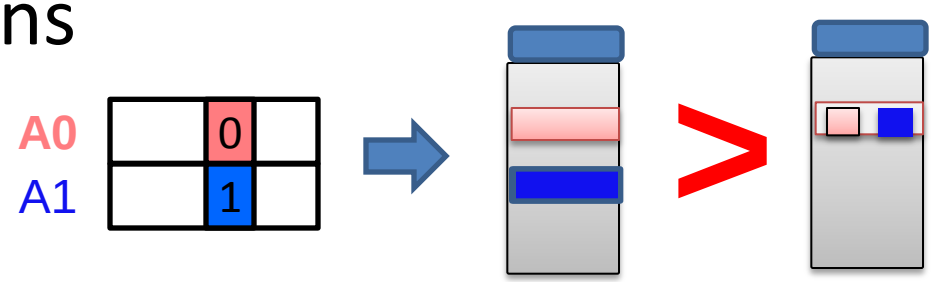
- The idea is simple, but in practice the mapping from physical address to DRAM banks is not fixed.



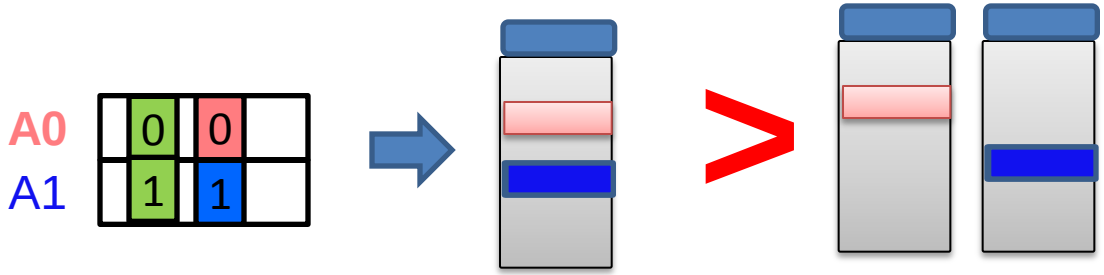
- **Challenge**: How to figure out memory mapping information to extract the bank bits?

Discover Bank Bits (1)

- We use HMTT to measure the latency of different access patterns

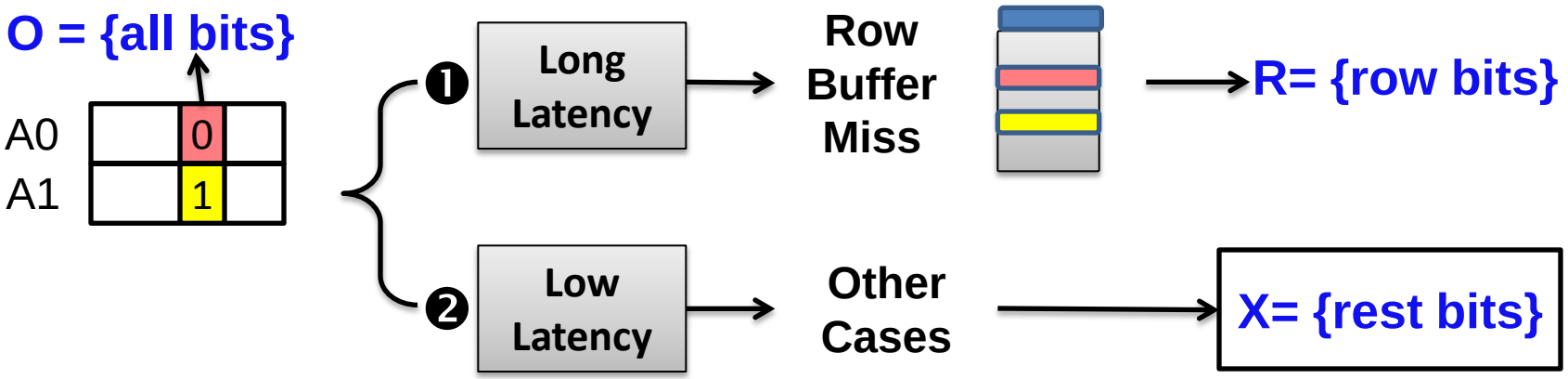


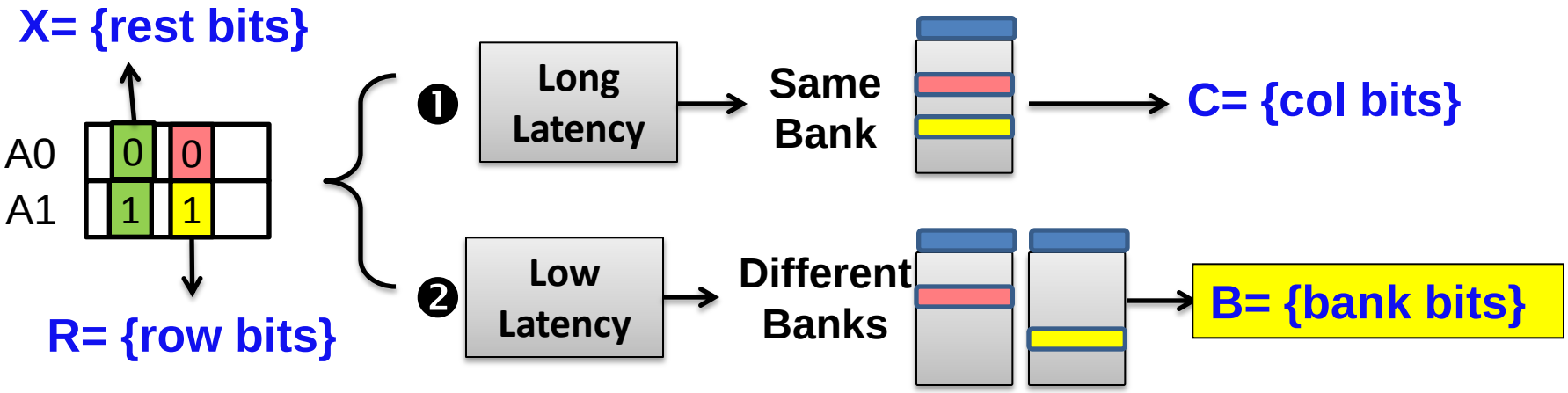
- Two Facts:
  - the latency of row buffer miss is much longer than the latency of row buffer hits
  - The latency of row buffer miss within a bank is still longer than concurrent accesses to two different banks



Discover Bank Bits (2)

Algorithms





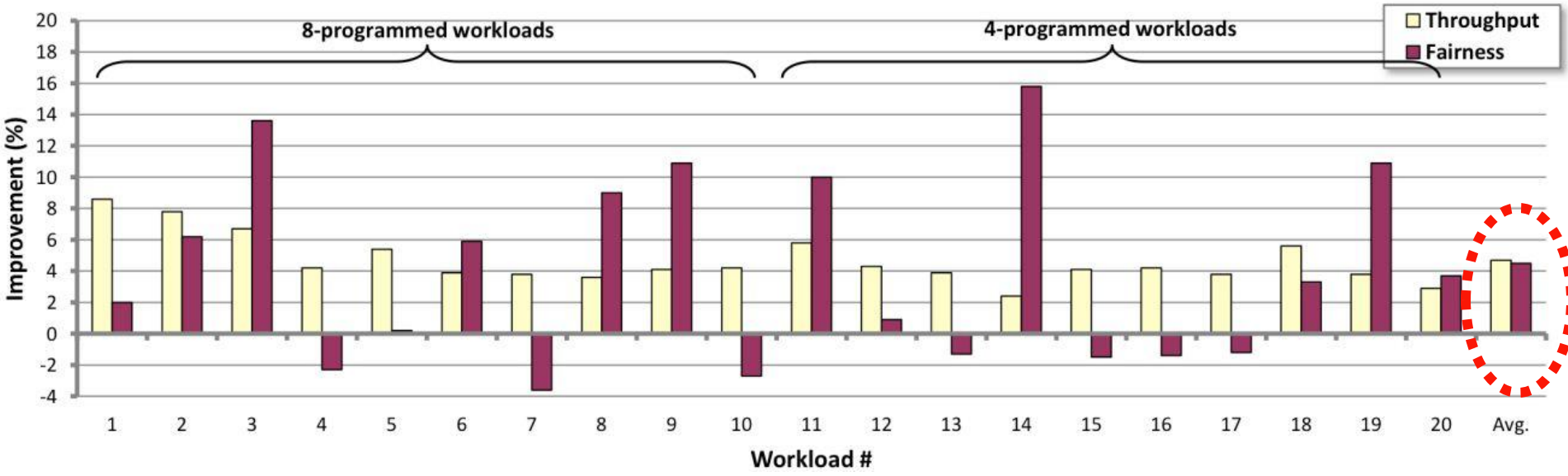
Bank-level Partition Mechanism (BPM)

- **Implementation**: adopt page-coloring base BPM in Linux kernel 2.6.32 by modify its buddy system.
  - group free pages into 32 colors.
  - Adjust the page allocation algorithm.

- **Experiments**
  - 4-core/8-thread Intel Core i7 CPU
  - 8GB DRAM, 64 banks, 32 color
  - SPECCPU 2006 (Multi-Program), PARSEC (Multi-Thread)

Experimental Results

- System throughput : **4.7%↑ (up to 8.6%)**
- Maximum slowdown: **4.5%↓ (up to 15.8%)**
- Memory Power **: 5.2% ↓**



Conclusion

- We design and implement a **hardware/software hybrid memory trace tool (HMTT)**
  - High-Level Event Encoding Mechanism (HLE2M)
- We also design some tools based on HMTT
  - Object-Relative Profiling
- The HMTT tool chains allow us to study memory system on real systems
- We have **provided free memory traces (>1TB)** to many research groups

Thanks

Q&A?