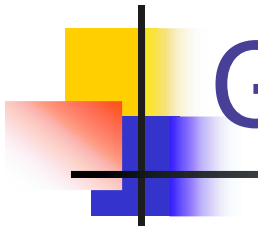
Hardware Supported Memory Access for High Performance Main Memory Databases

Jun Miyazaki [1), 2)]

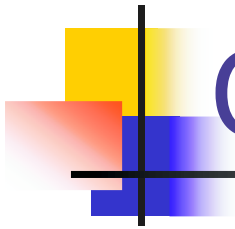
[1)] Graduate School of Information Science
Nara Institute of Science and Technology

[2)] PRESTO, Japan Science and Technology Agency



General question

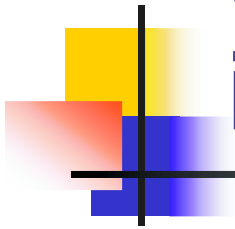
- Do you think it is comfortable to install a large amount of memory on a PC?
  - It just avoids chances of thrashing.
  - Modern computer is not really designed for random access, but for cache oriented memory access.



Objectives

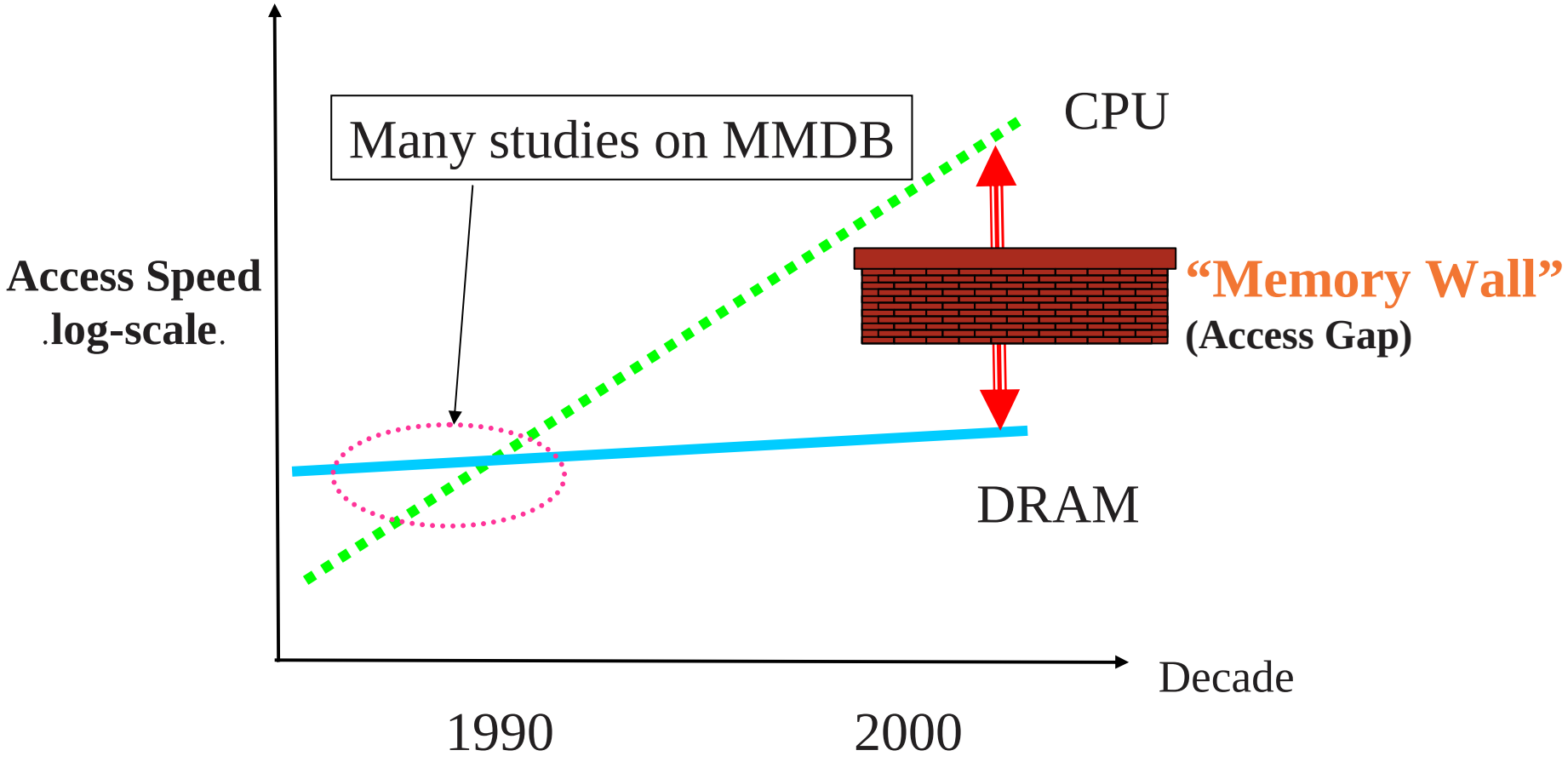
- People expect faster database processing with their inexpensive PC
  - Price of memory drastically drops
    - Giga-bytes of main memory is large enough to store databases in many cases
  - Speed of processors becomes much faster than that of memory
    - But, multi-way memory interleaving is still expensive

Demand for fast data processing in main memory



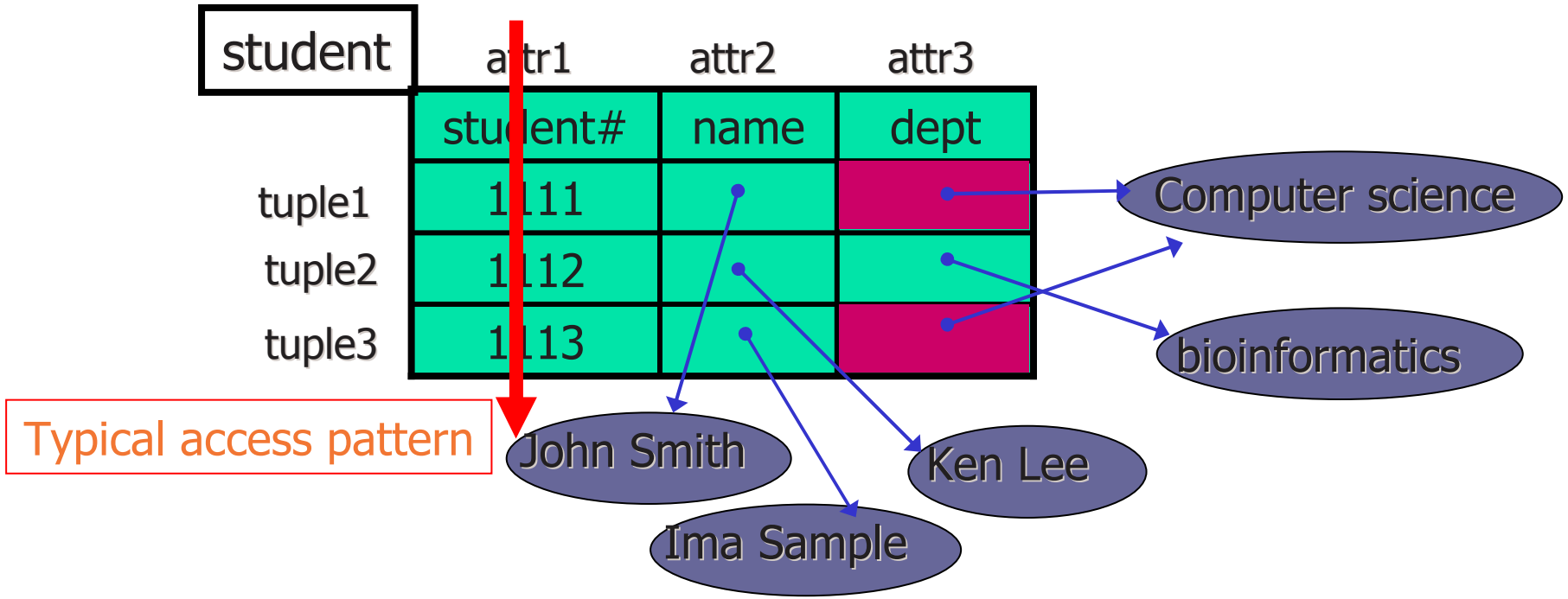
- Main memory database (MMDB)
- Data mining
- Bioinformatics
- Stream processing
- Scientific computation
- Etc, etc, etc…

The Memory Wall issue



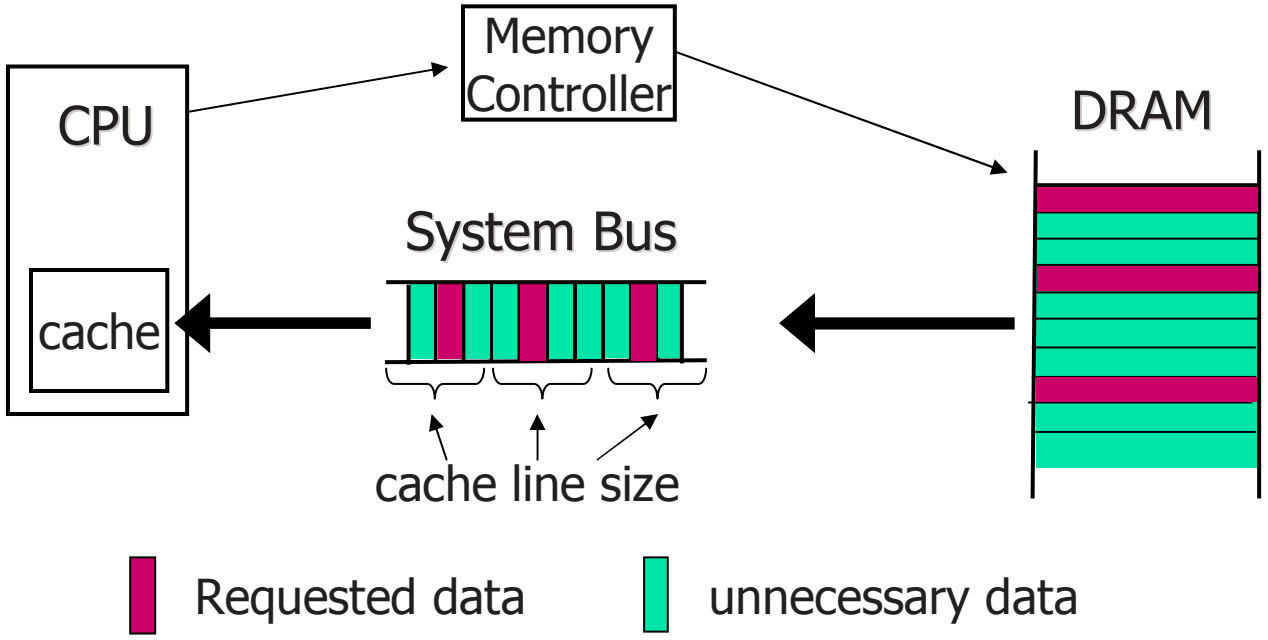
Main Memory Database (MMDB)

- A data structure for MMDB
  - Tuple size can be forced to be regular size.
    - Integer and other small data type are aligned with word size
    - Large objects are stored in other memory space, and then, the corresponding pointers are stored, instead.



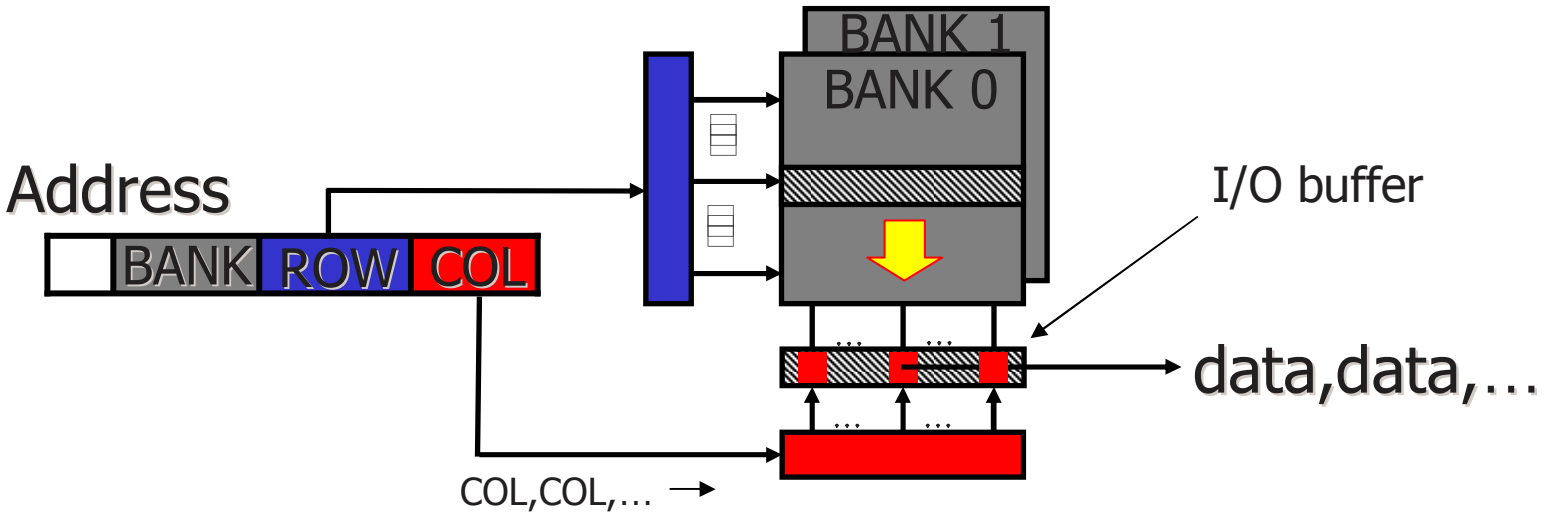
Memory access of modern processors

- Cache line oriented data transfer
  - Unnecessary data in the same line are also transferred through the system bus
    - Does not match the access patterns in MMDB



Page mode of DRAM

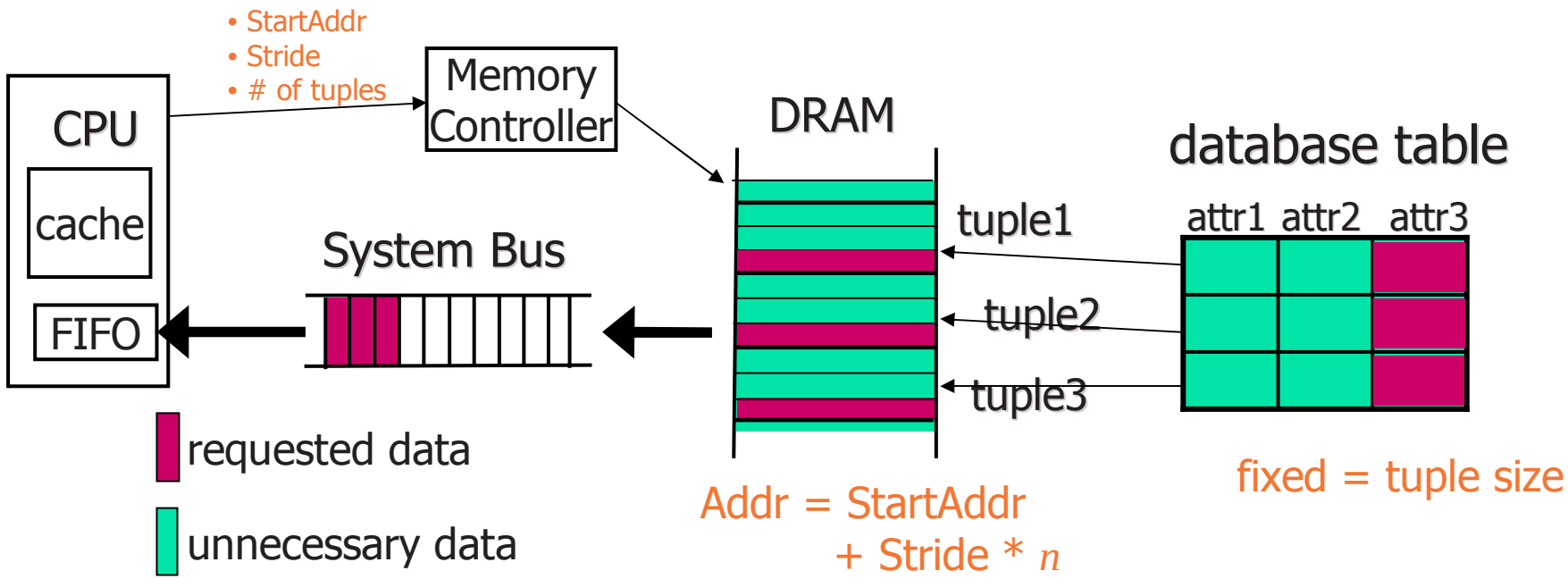
- Fast data read mode in the same row
  - If giving COL addrs sequentially while BANK and ROW addrs are retained, the corresponding data are output in a pipeline fashion.



Pipelined transfer is valid only for the data in the same row.

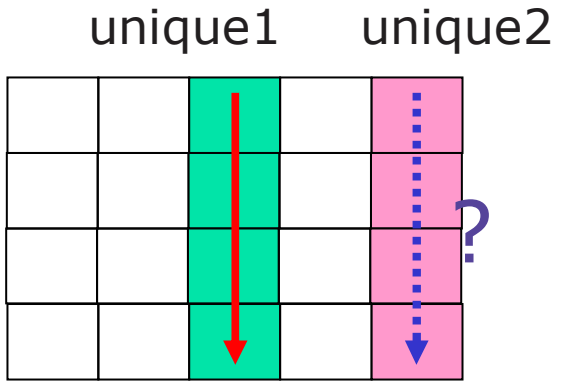
Stride data transfer (SDT)

- Data placed at a fixed stride are contiguously transferred to CPU with page mode of DRAM
  - COL addr can be automatically generated by memory controller using start addr, stride, and # of tuples



Issue of SDT

- operations which scan more than one attributes
  - (ex)
    ```
    select  *
    from    T
    where unique1 < 1000
    and     unique2 < 1000
    ```

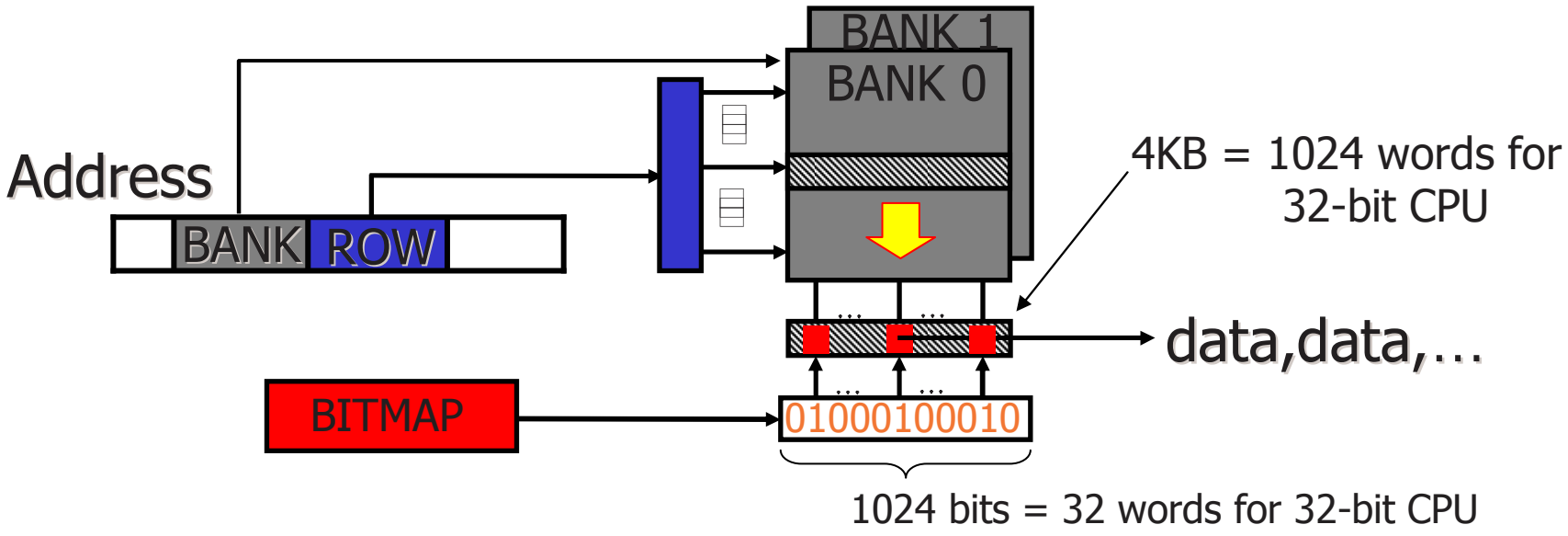


  - Applying SDT multiple times?
    - Setup time for SDT becomes bottleneck

      i.e. applying StartAddr, stride, and # of tuples to MC

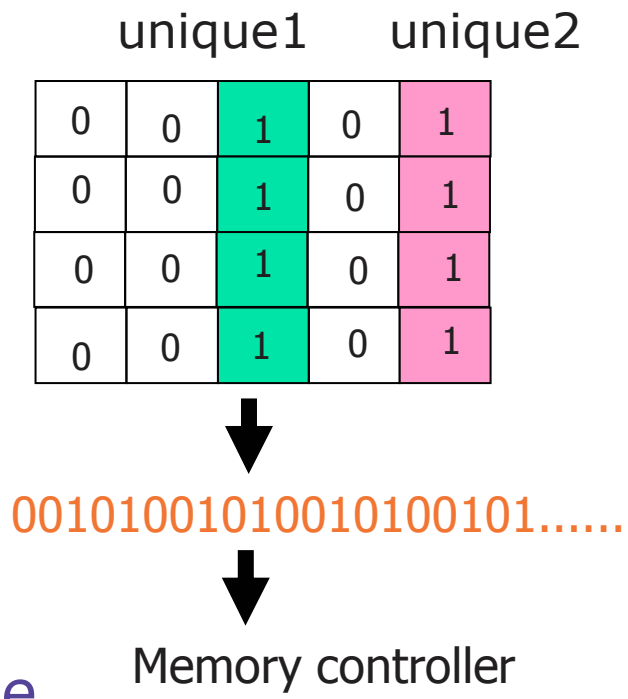
Bitmap-based data transfer (BDT)

- Positions of data requested are specified by bitmap, not addresses
  - Random data (non-fixed stride data) in the same row can be accessed
  - It takes longer time to set bitmap to MC



Accessing memory using BDT

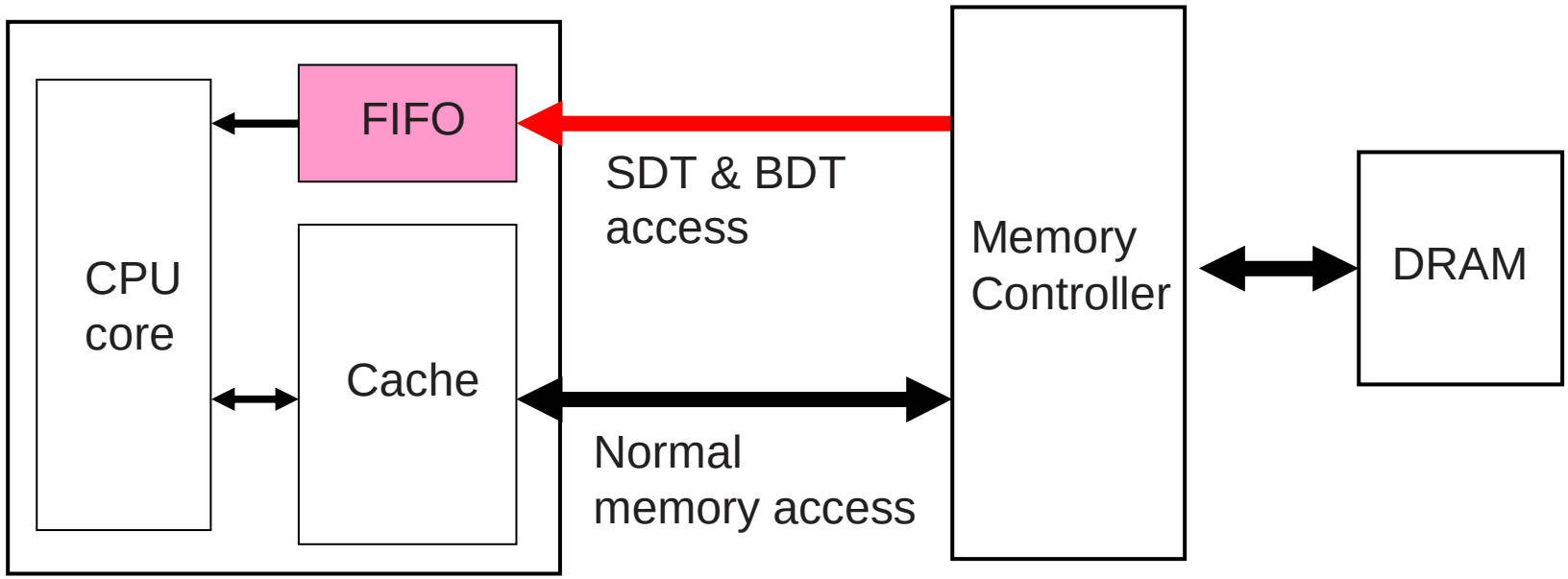
- operations which scan more than one attributes
  - (ex) select *
    from T
    where unique1 < 1000
    and unique2 < 1000

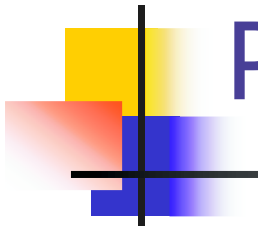


  - Bitmap is nothing but an access pattern to the database table

System overview

- Adding FIFO into CPU
  - SDT & BDT accesses are not compatible with cache
  - Memory access does not need to wait until cache line is filled (computation and memory access work simultaneously)





Performance evaluation

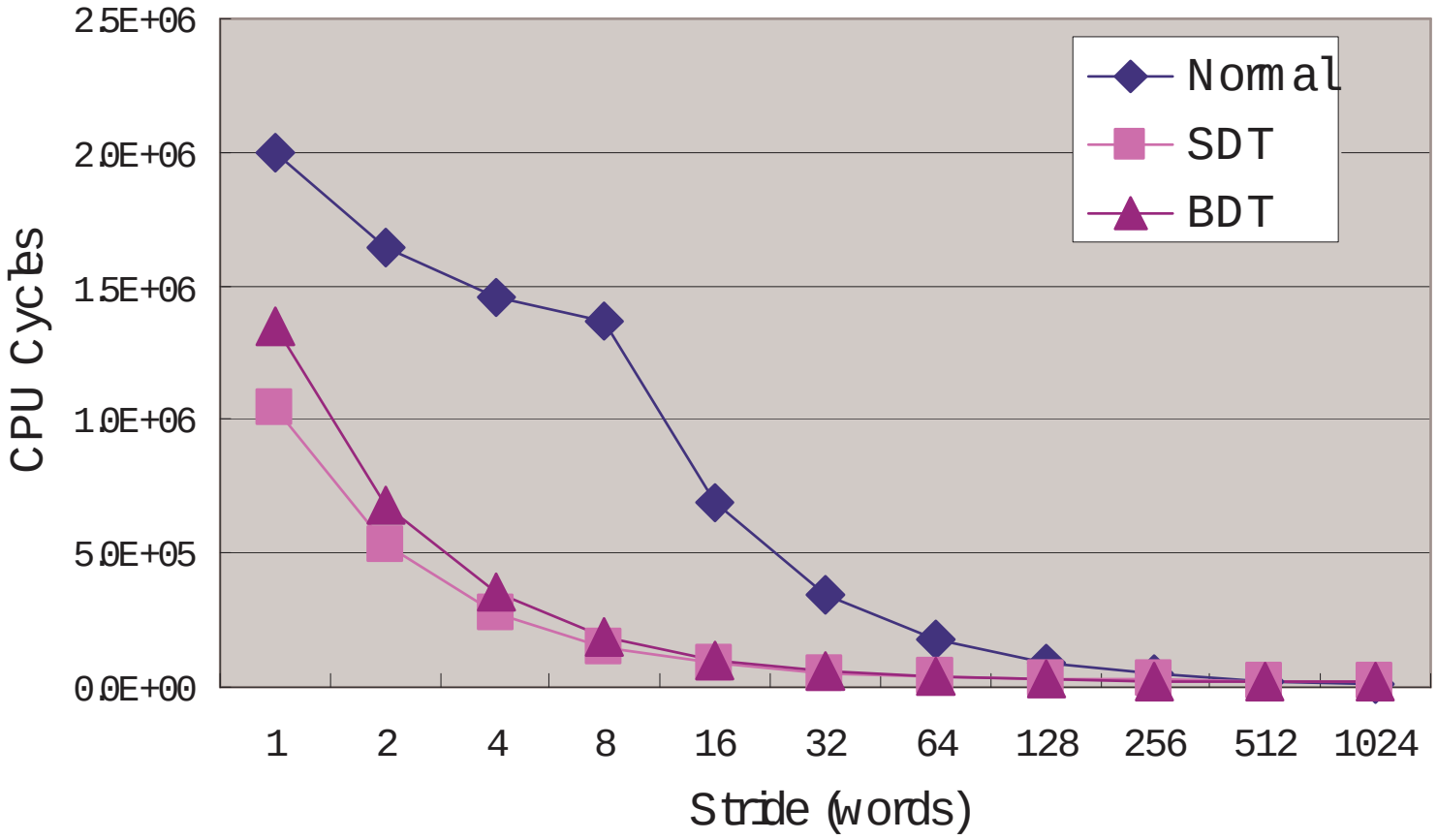
- Specification of simulator
  - 32-bit SPARC compatible CPU
  - 1 instruction per CPU cycle
  - 1 system bus cycle: r CPU cycles
  - Row width of DRAM: 4KB (=1K words)
  - Instruction and data caches
    - L1: 8KB, 2-way set associative
    - L2: none
  - FIFO: 4KB (the same performance of cache)

Performance evaluation

- CPU cycles related to memory access
  - Cache hit: 1
  - Cache miss: 10r
  - Writeback one cache line to DRAM: 14r
  - Set one word data to MC: 4r
  - Latency to start and restart SDT & BDT: 12r
  - Read one word from FIFO: 1 (under FIFO hit)
    2~r (under FIFO miss)

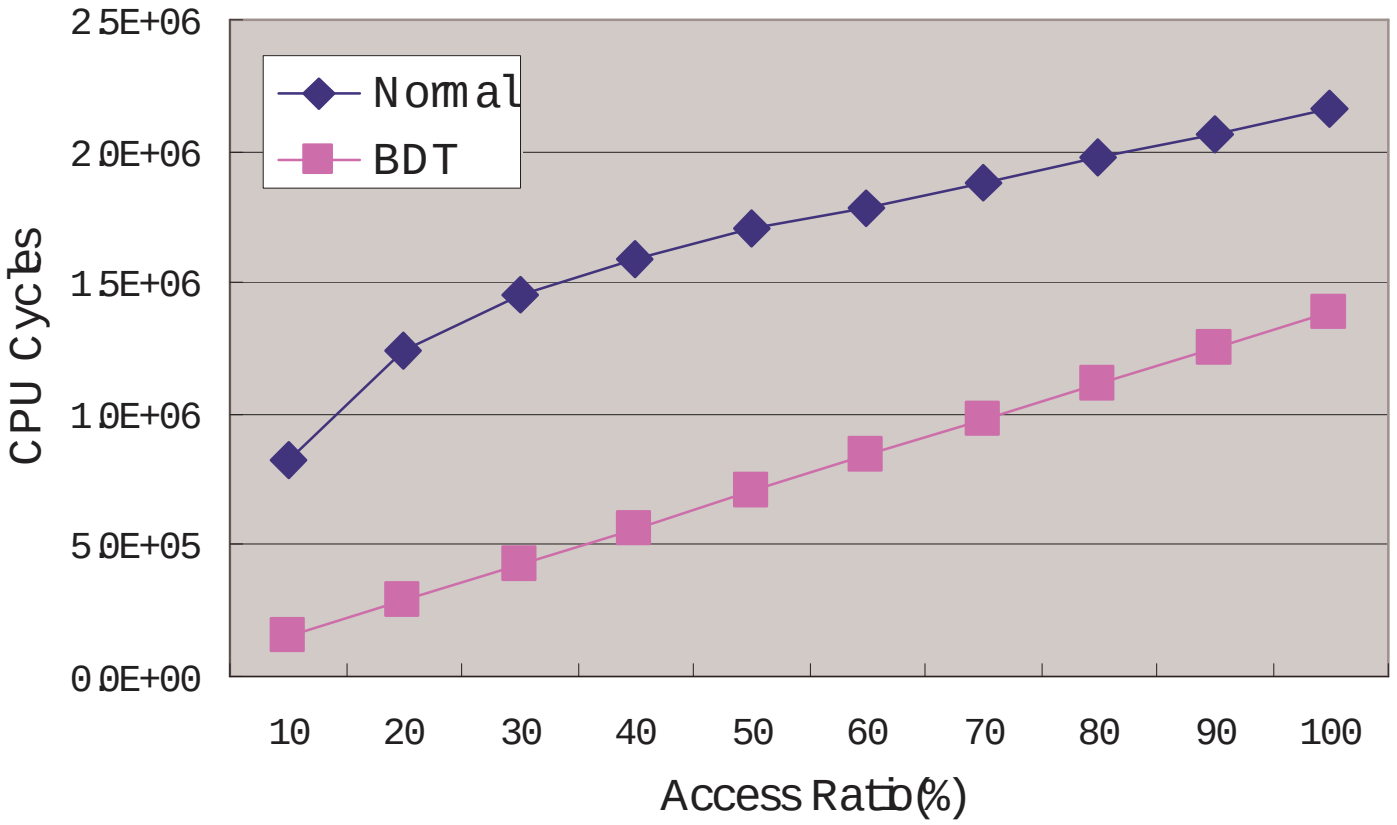
- r = 10 for the following experiments

Fixed stride data access



Random data access

- Addresses are randomly chosen in given access ratio



Queries for benchmark

(Q1) SELECT *
FROM tenk1
WHERE unique1 > 10000
AND unique2 > 10000

(Q2) SELECT *
FROM tenk1
WHERE unique1 > 10000
AND unique2 > 10000
AND unique3 > 10000

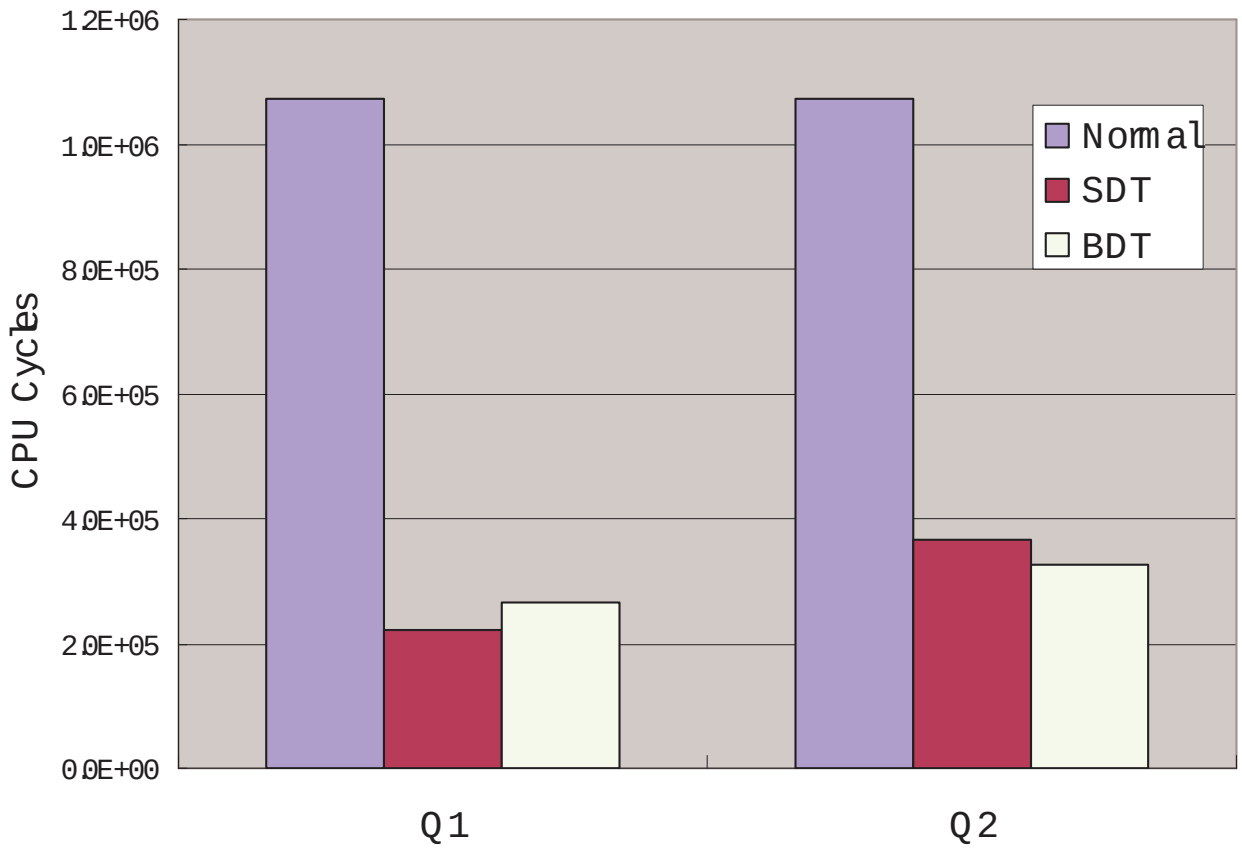
(Q3) SELECT *
FROM tenk1
WHERE unique2 > 301
AND unique2 < 402

(Q4) SELECT *
FROM tenk1 t1, tenk1 t2
WHERE t1.unique2 =
t2.unique2
AND t2.unique2 < 1000

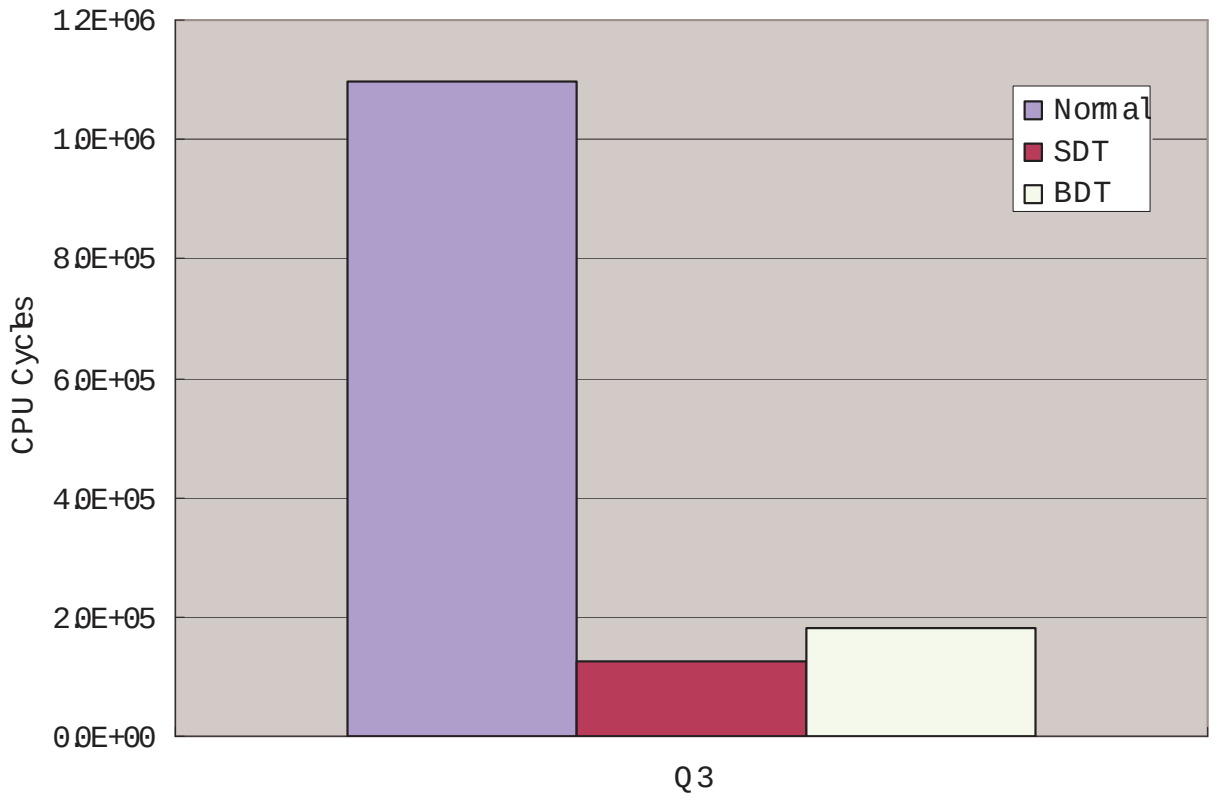
Comparing SDT and BDT

realistic queries

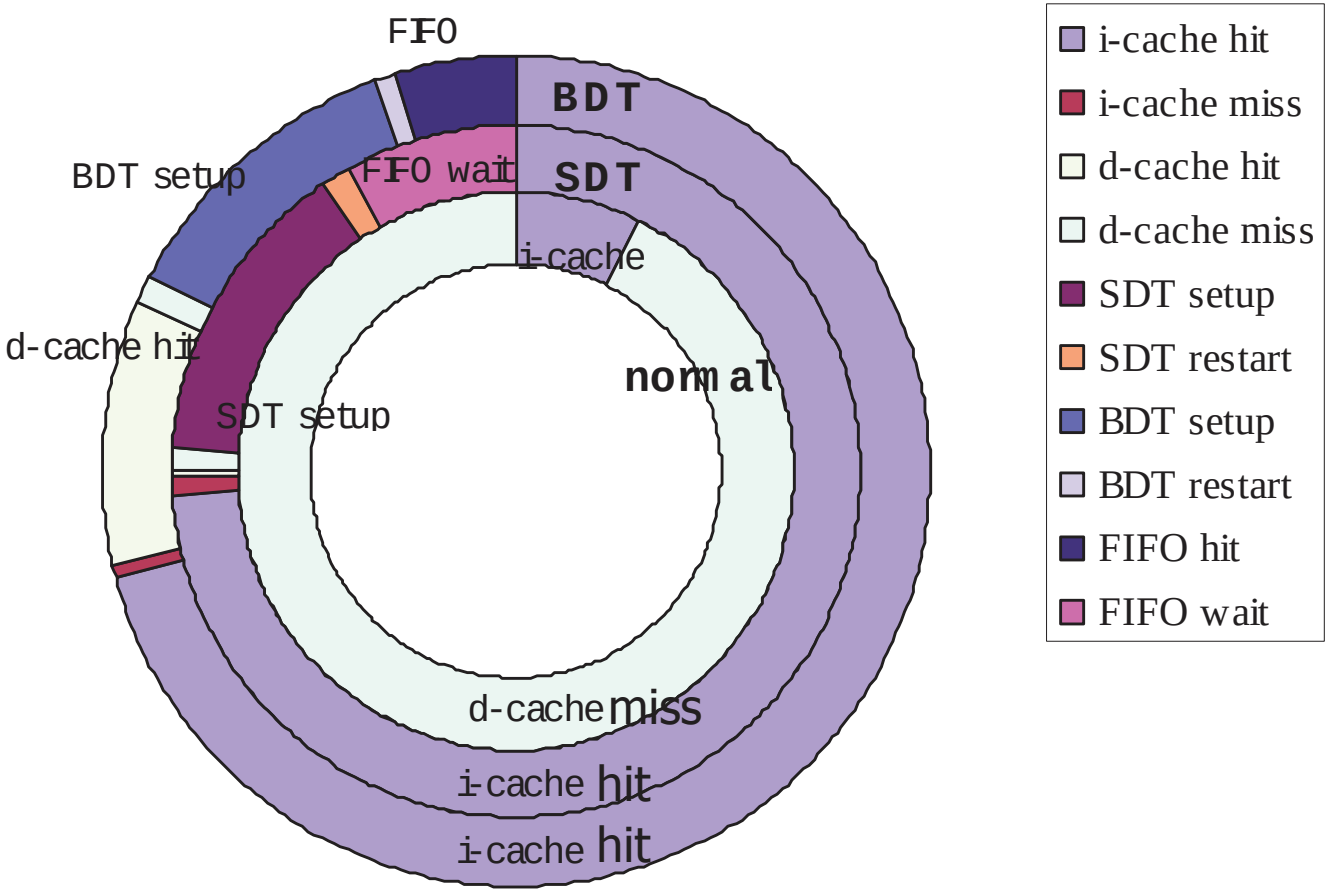
Results of Q1 and Q2



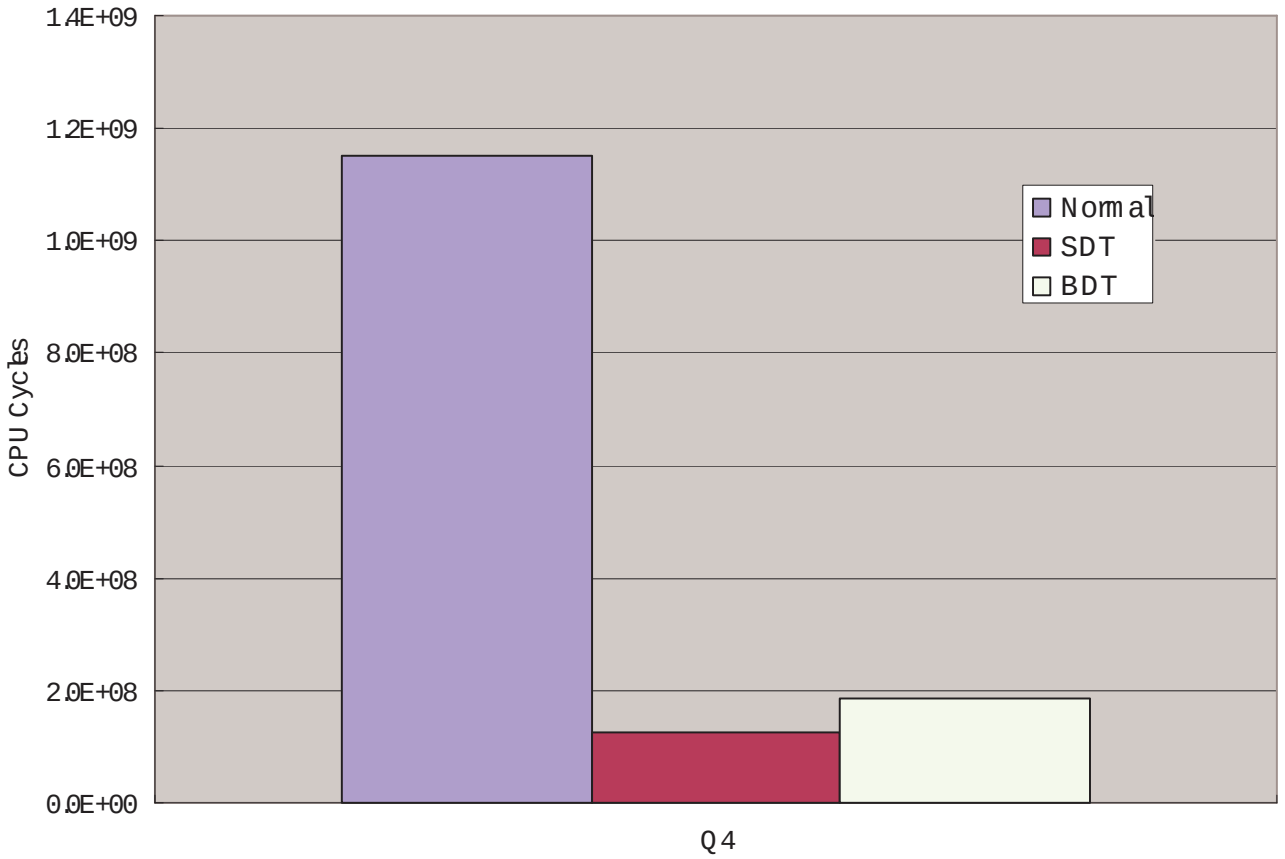
Results of Q3



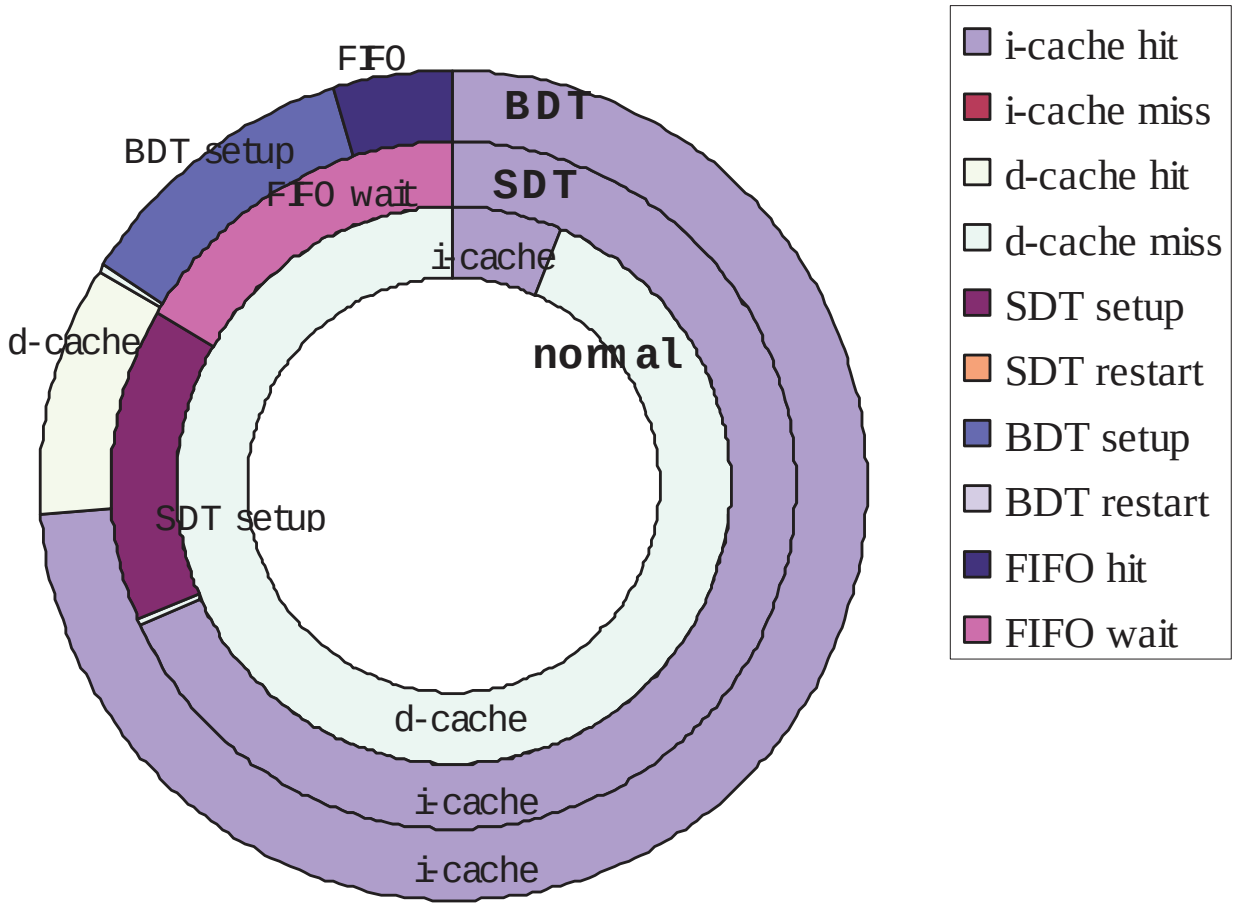
Breakdown of Q3



Results of Q4

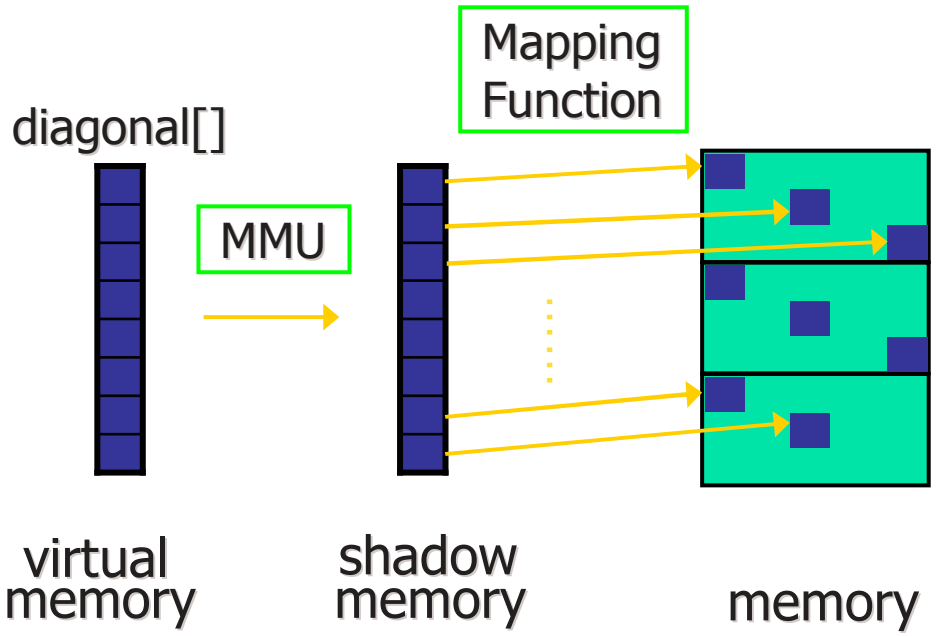


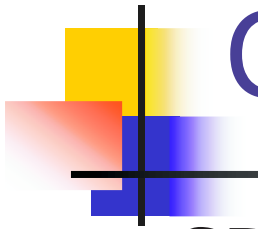
Breakdown of Q4



Related Work

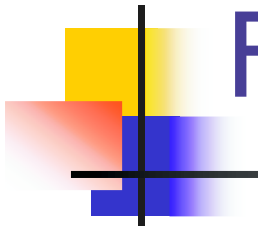
- Impulse [Zhang et al. 2001]
  - Supports *stride mapping* (similar to SDT)
  - Does not support bitmap-based access which enables truly random access





Conclusion

- SDT and BDT are proposed for memory access in MMDB
  - SDT is suitable only for fixed stride data
  - BDT is suitable for complex random access like scanning multiple attributes
  - Queries using these methods outperform normal memory access by one order of magnitude

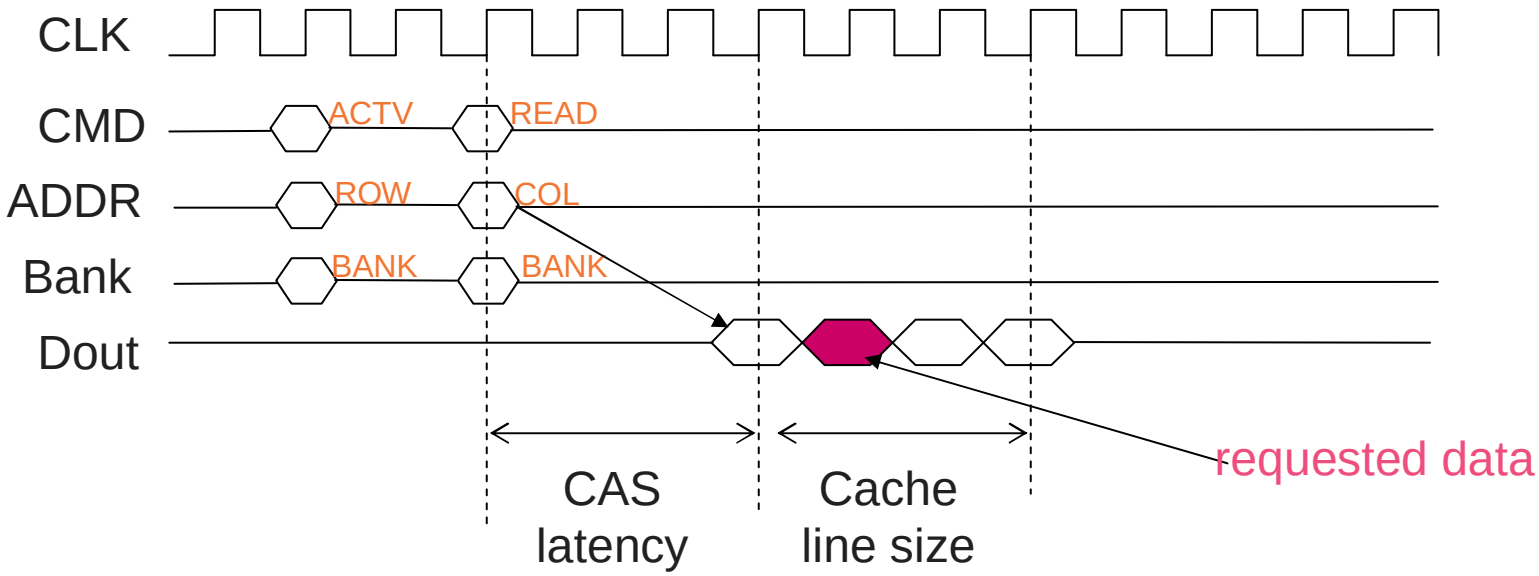


Future work

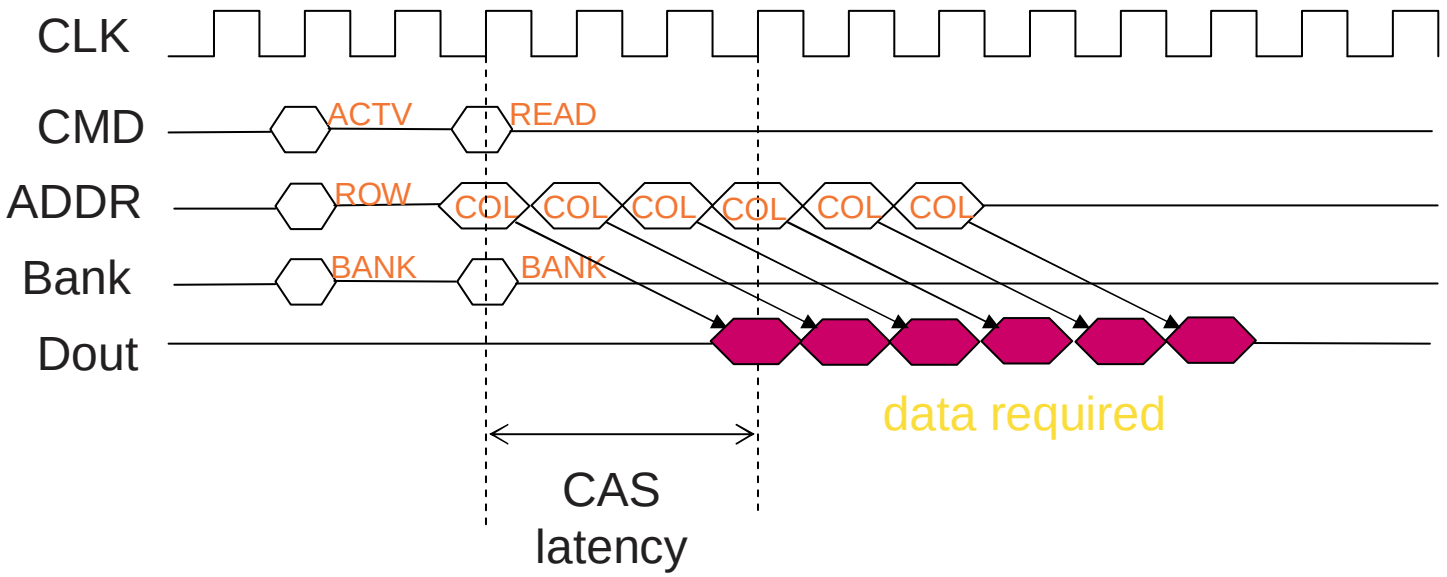
- Optimization by selecting the best memory access method among normal, SDT, and BDT
- Index using SDT and BDT
- Evaluation by more complex queries such as TPC-H

Memory access of modern processors

- Cache line oriented data transfer
  - Unnecessary data are also accessed

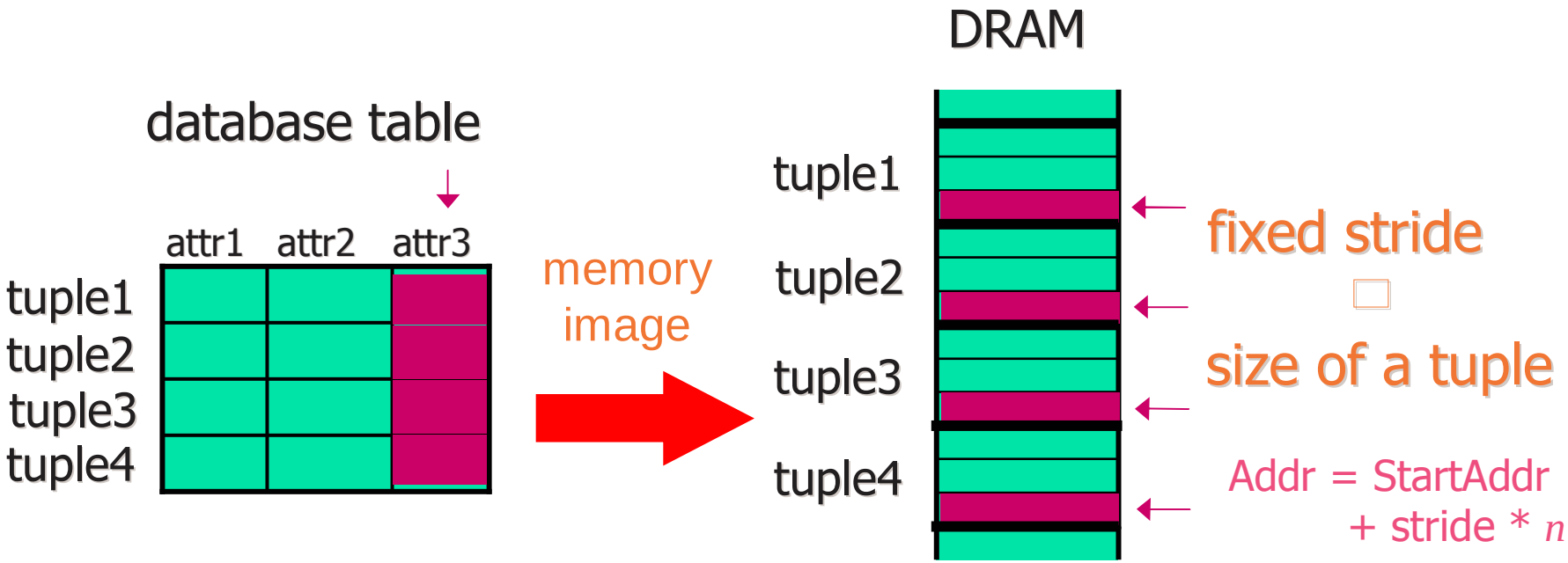


Stride data transfer / SDT

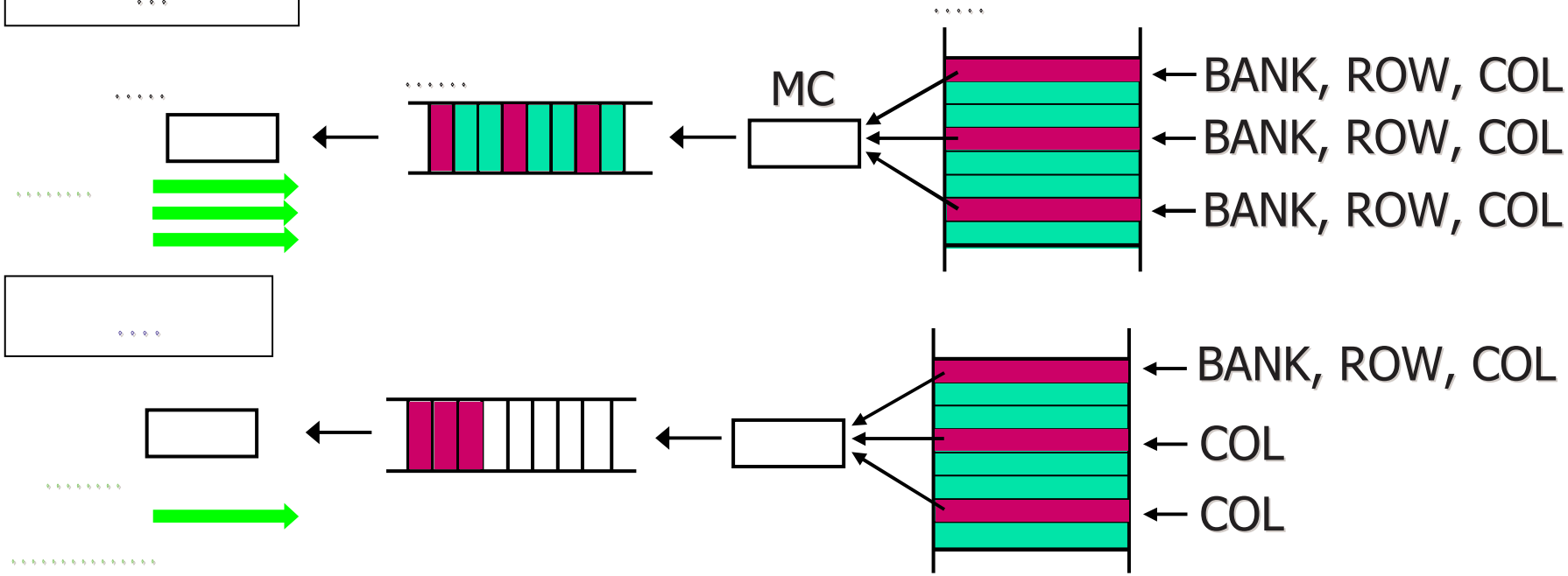


Stride data transfer (SDT)

- Data placed at a fixed stride are contiguously transferred to CPU using page mode of DRAM
  - COL addr is automatically generated by memory controller using start addr, stride, and # of tuples

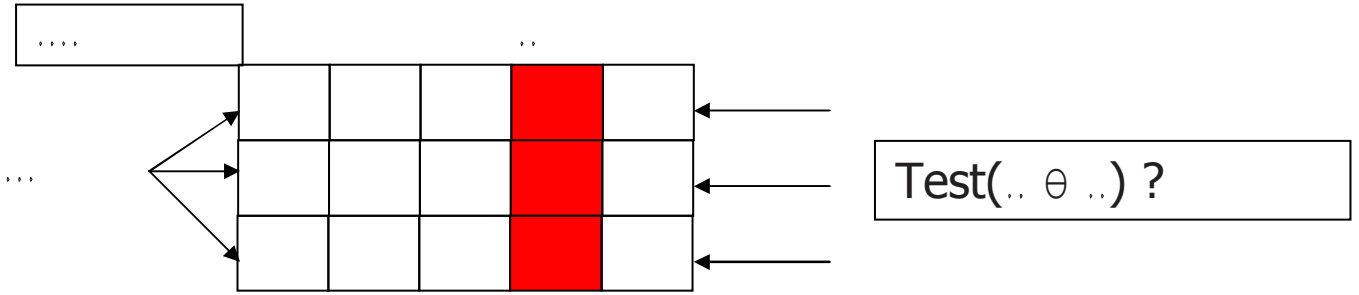


MC

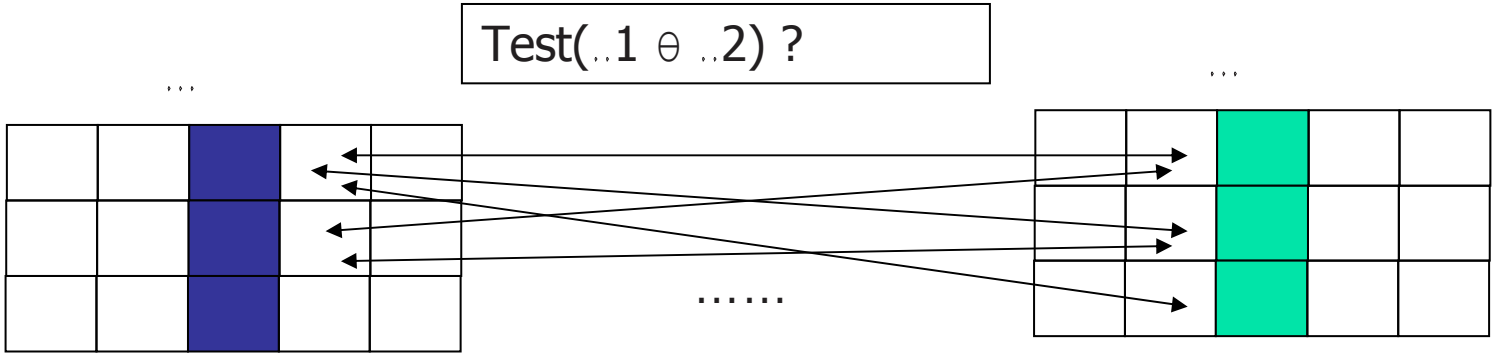


- MC

- selection (



- join ( )



Bitmap-based Data Transfer (BDT)