A Low-Overhead High-Performance Unified Buffer Management Scheme that Exploits Sequential and Looping References

Jong Min Kim Jongmoo Choi Jesung Kim
Sam H.Noh Sang Lyul Min Yookun Cho
Chong Sang Kim

School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea

Department of Computer Engineering
Hong-Ik University
Seoul 121-791, Korea

Outline

- Introduction
- Related work
- The Unified Buffer Management Scheme
- Simulations
- Implementation
- Conclusions

Introduction(1/2)

- Efficient management of the buffer cache is important.
- The LRU scheme is still widely used due to its simplicity.
- A main drawback of the LRU scheme, is that it cannot exploit regularities in block accesses such as sequential and looping references.

Introduction(2/2)

- It devises a new buffer management scheme called the Unified Buffer Management(UBM) scheme.
- The UBM scheme exploits regularities in reference patterns such as sequential and looping references.

Related Work

- Page/block replacement schemes :
  - It's based on frequency and/or recency factors.

    ex : LRU,LFU,LRFU.
  - It's based on user-level hints.
  - It's making use of regularities of references such as sequential references and looping references.

    ex : 2Q,SEQ,EELRU.

Related Work

Access order : 6 , 4 , 5 , ...

	A1in			A1out		
Initial:	9	4	0	3	7	1
access 6 :	4	0	6	3	7	1
access 4 :	0	6	1	4	3	7
access 5 :	6	1	5	4	3	7

MRU

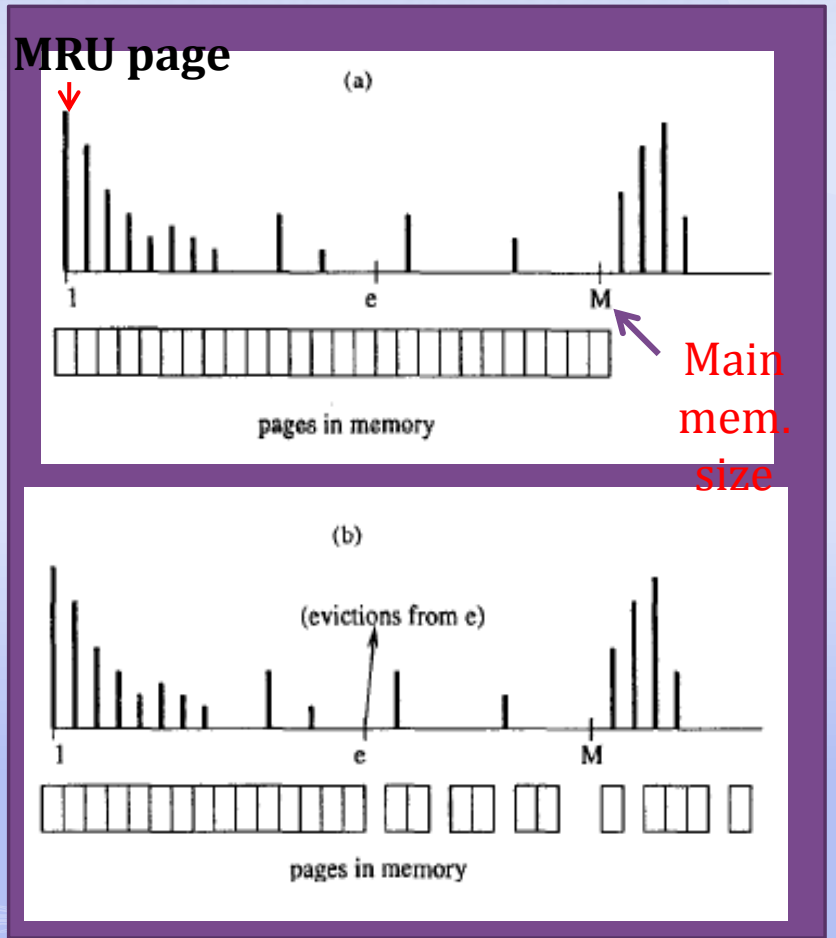
- SEQ :

  6

  It detects long sequences of page faults and applies the MRU scheme to those pages.

If
time 0 , access 6 pages (page: a~f)
time 2 , access 1 page (page : z)
time 3 , access 6 pages(page: g~l)
time 4 , access 2pages (page : x,y)
time 5 , access 10 pages(page: n~w)
Then
Evicted block order : (n~w),(g~l),(a~f),(z),(x,y)

- 2Q
  - A1in queue (FIFO)
  - A1out queue(LRU)
- EELRU
  (early eviction LRU)



The UBM Management Scheme

- It is composed of following three main modules :
  - Detection
  - Replacement
  - Allocation

Part 1 : Detection

- **Sequential references** that are consecutive block references occurring only once.
- **Looping references** that are sequential references occurring repeatedly with a regular interval.
- **Other references** that are detected neither as sequential nor as looping references.

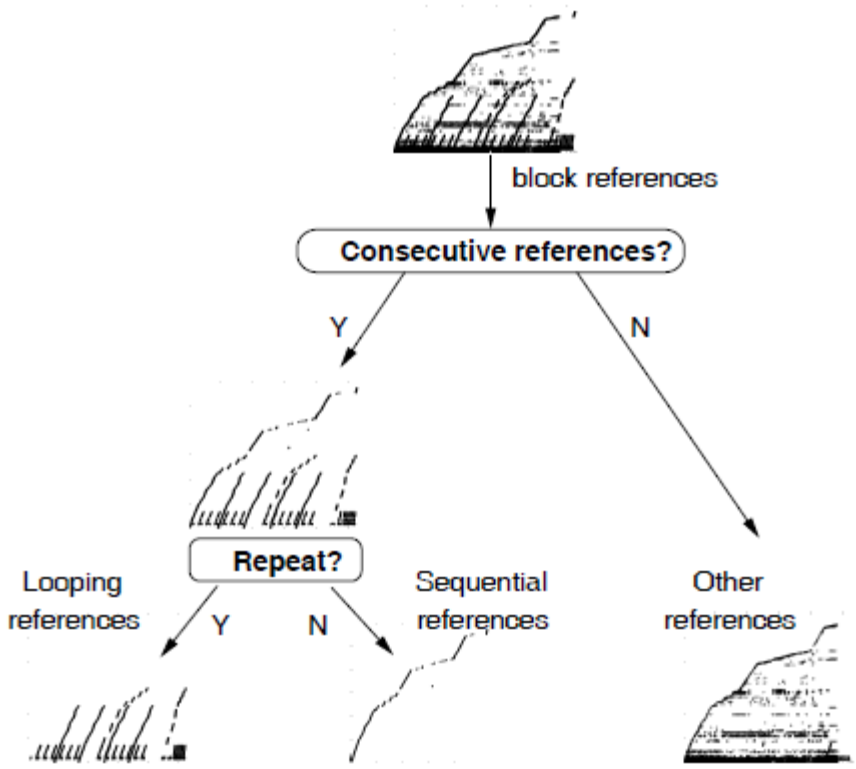
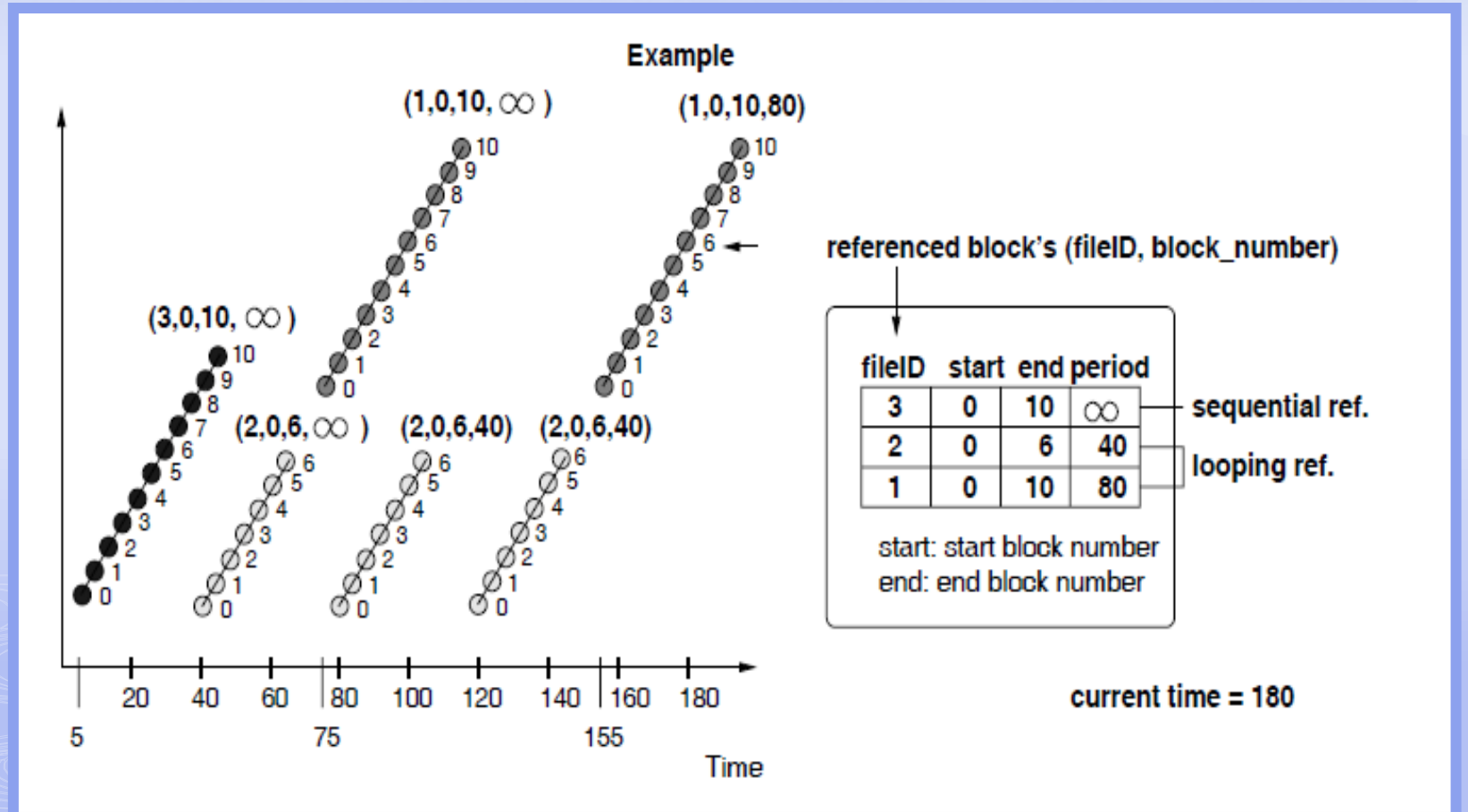


Figure 2: Classification process of the UBM scheme.

Part 1 : Detection

- For example :



Part 2 : Block Replacement Schemes

- The buffer cache is divided into three partitions to accommodate the three different types of references.
  - **Sequential reference** :

    MRU replacement policy is used.
  - **Loop reference** :
    It's based on their periods.
  - **Other reference** :
    It's based on LRU replacement scheme.

Part 3 : Buffer Allocation Based on MG

- Marginal gain
  - Which has frequently been used in resource allocation strategies in various computer systems areas.
  - It's defined as $MG(n) \approx Hit(n) - Hit(n-1)$
- The expected number of buffer hits per unit time of sequential references when using n buffers is $Hit_{seq}(n) = 0$, $\therefore MG_{seq}(n) = 0$.

Part 3 : Buffer Allocation Based on MG

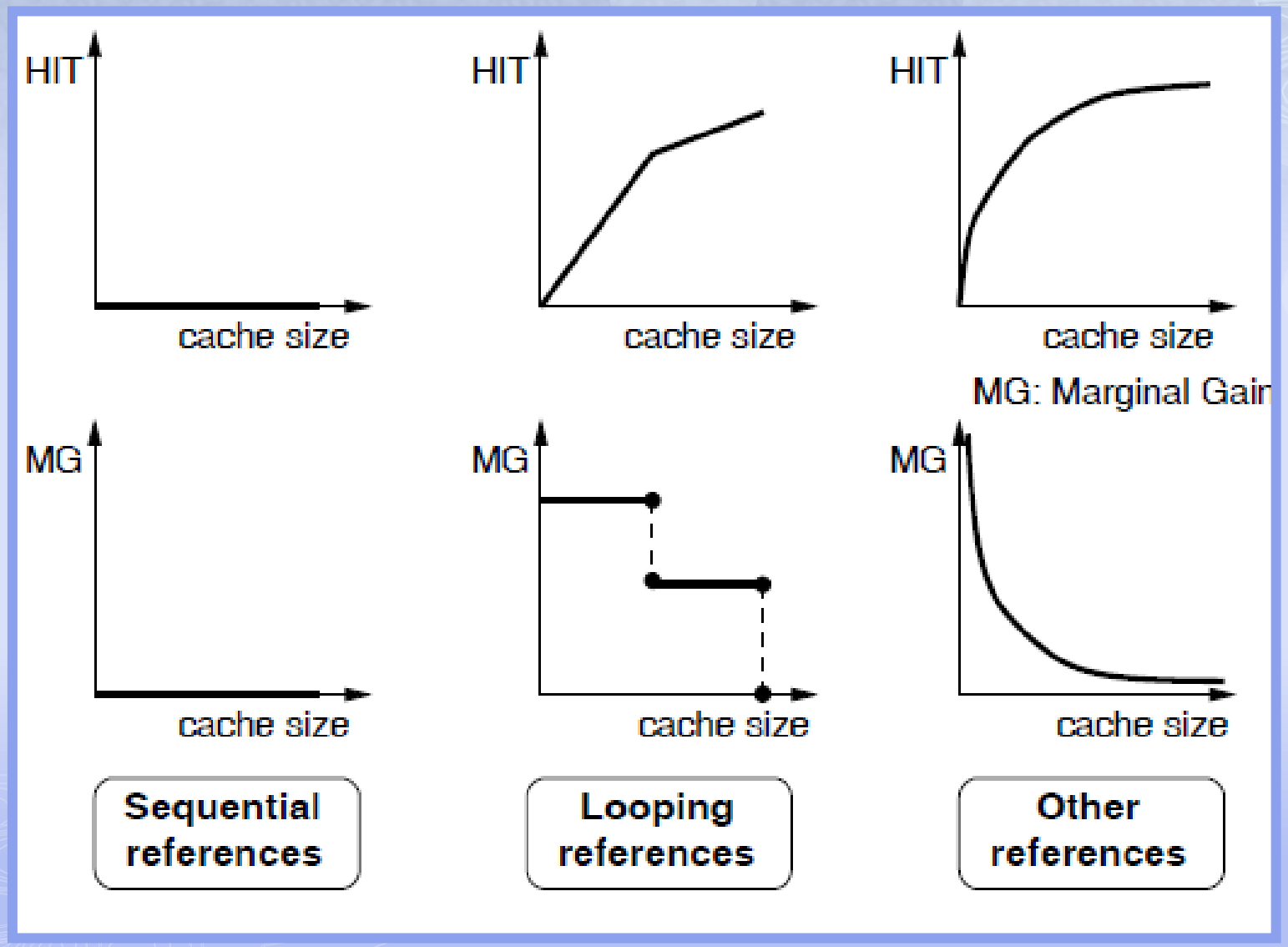
- For a looping reference :

  $loop_i$, with loop length l_i and loop period p_i

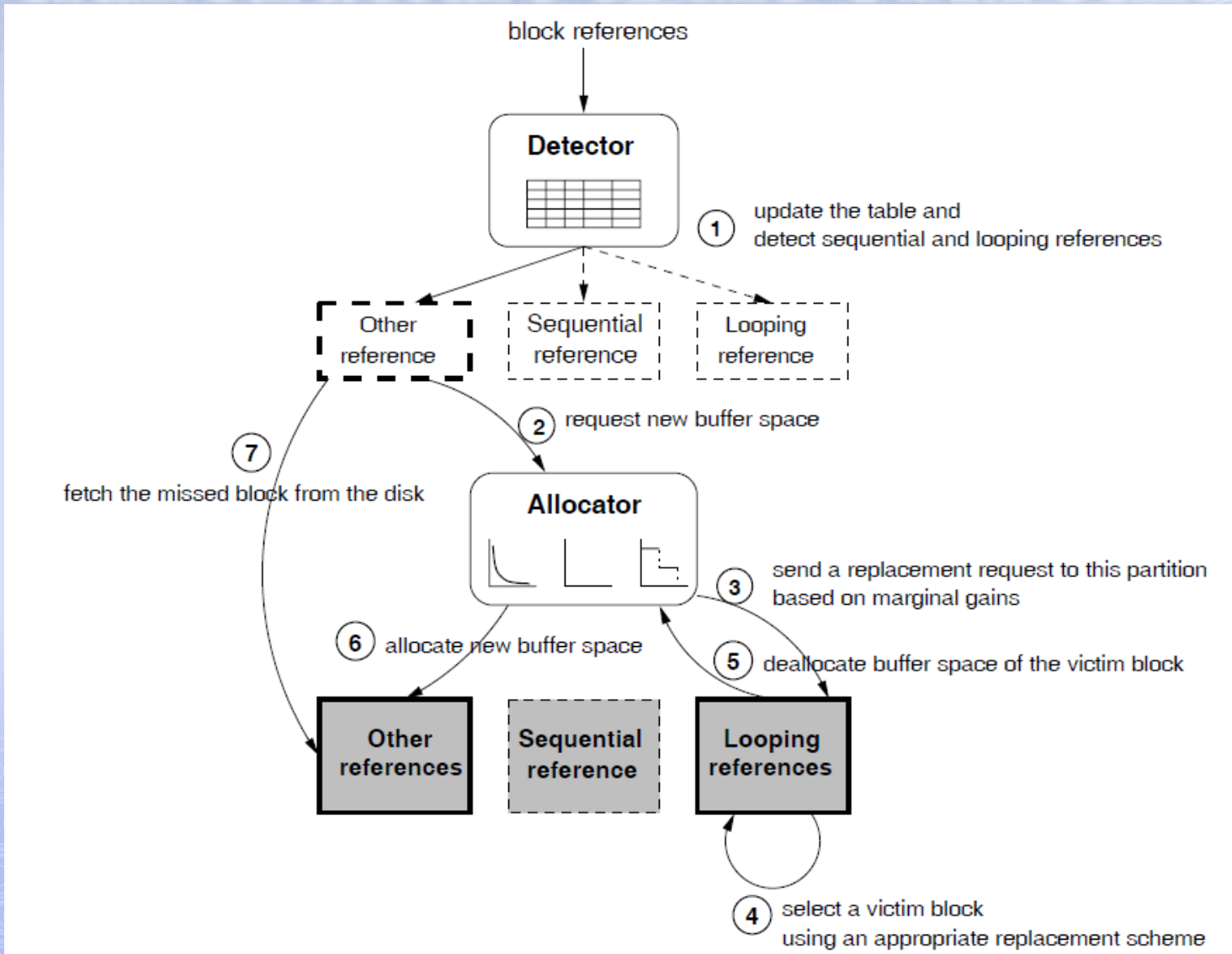
  - $Hit_{loop_i}(n)$ = $min[l_i, n]/p_i$
  - If $n < l_i$, $MG_{loop_i}(n) = n/p_i - (n-1)/p_i = 1/p_i$
  - If $n > l_i$, $MG_{loop_i}(n) = l_i/p_i - l_i/p_i$ = 0

Part 3 : Buffer Allocation Based on MG

- For other references :
  - Belady's lifetime function
    $hit_{other}(n) = h_1 + h_2 + h_3 + \ldots + h_n \approx 1 - c * n^{-k}$
  - $Hit_{other}(n) = hit_{other}(n) \times [\, n_{other} / n_{total} \,]$
  - $MG_{other}(n) = Hit_{other}(n) - Hit_{other}(n-1)$.


HIT
cache size
HIT
cache size
HIT
cache size
MG: Marginal Gain
MG
cache size
MG
cache size
MG
cache size
Sequential references
Looping references
Other references

Overall structure of the UBM scheme



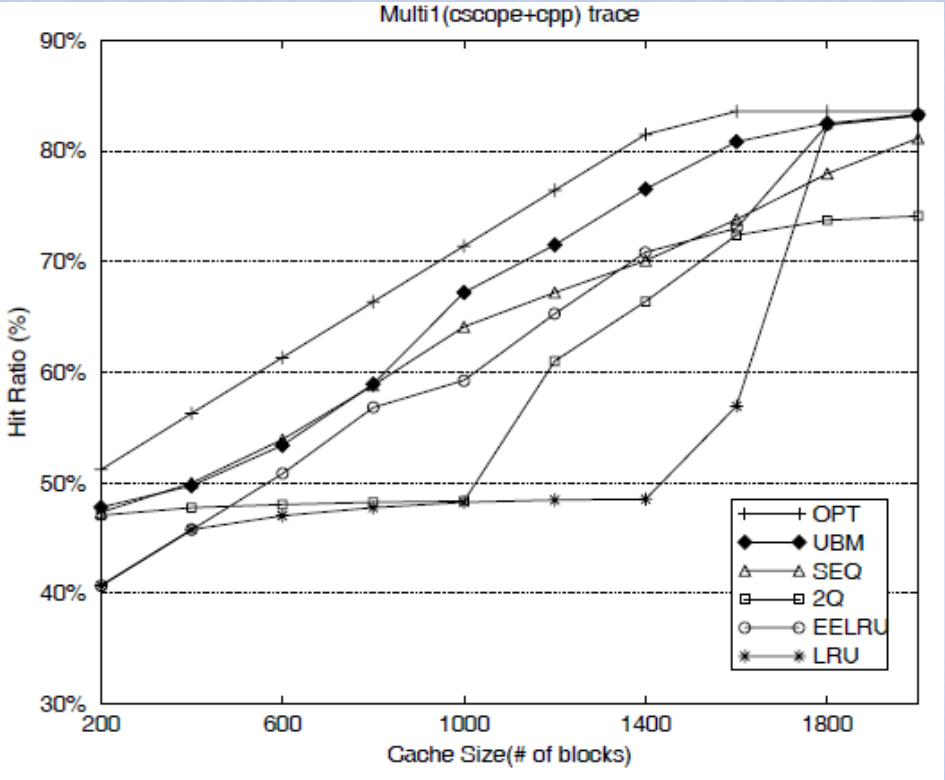
Experimental settings

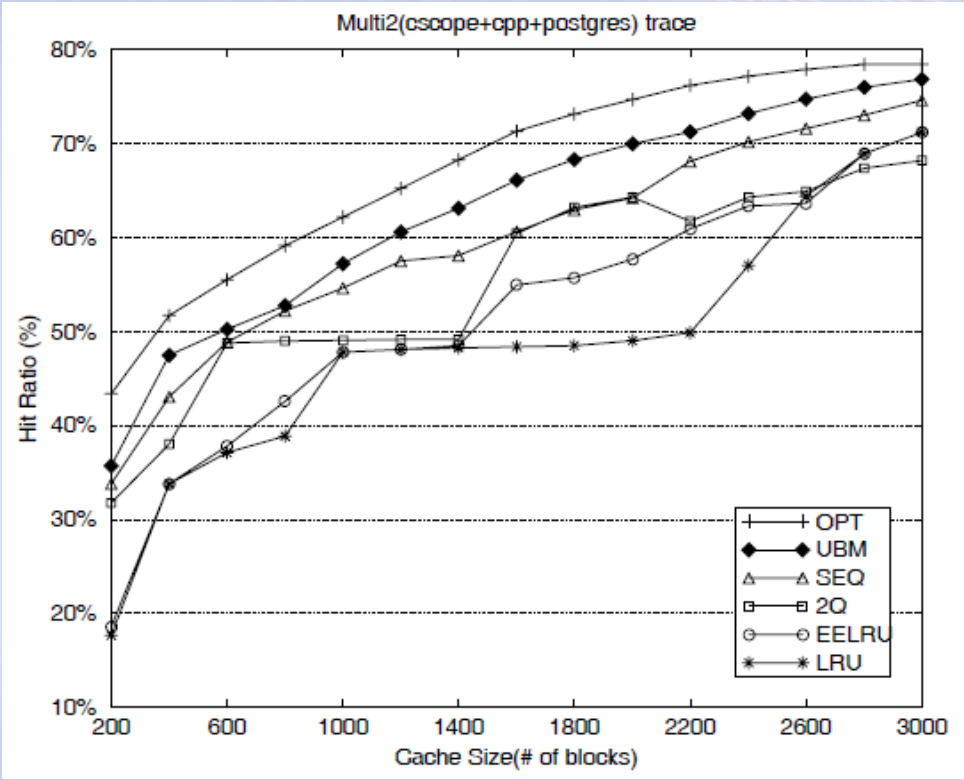
- Characteristics of the traces used.

Trace	Applications executed concurrently	# of references	# of unique blocks
Multi1	cscope, cpp	15858	2606
Multi2	cscope, cpp, postgres	26311	5684
Multi3	cpp, gnuplot, glimpse, postgres	30241	7453

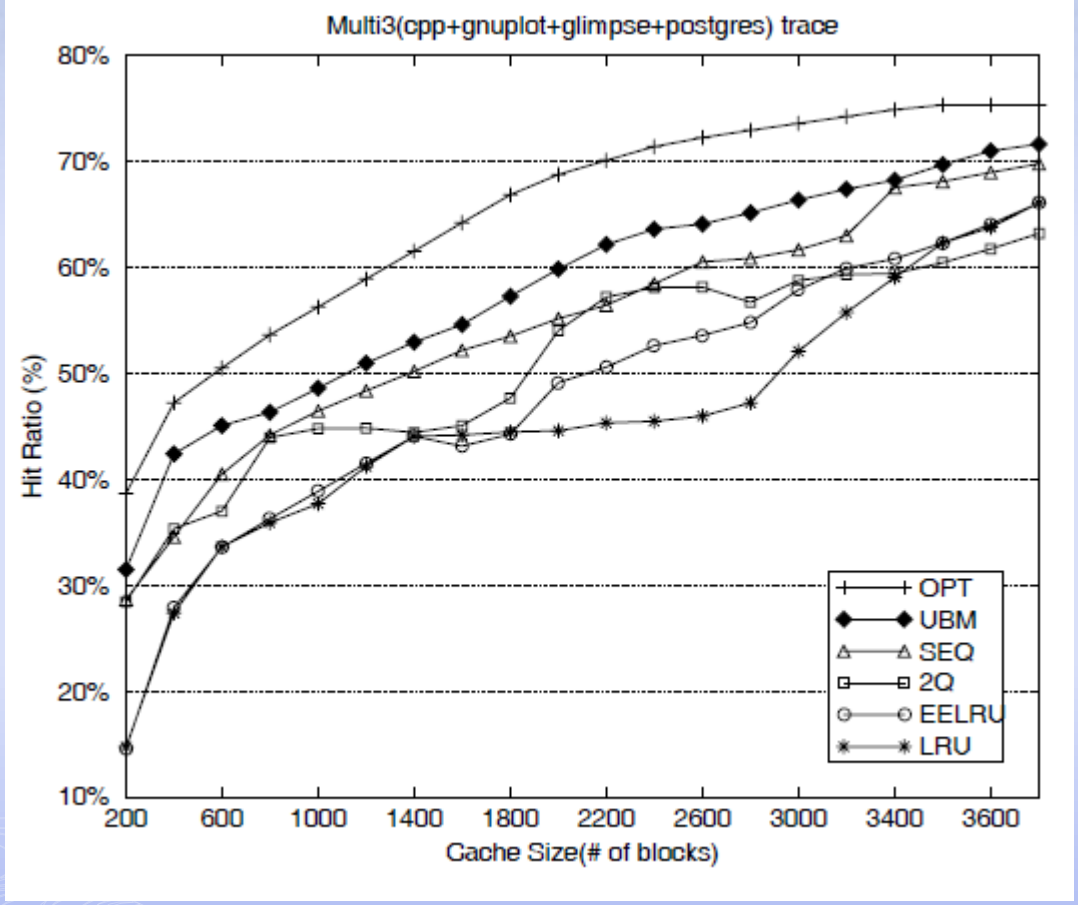
- We built simulators for the LRU, 2Q, SEQ, EELRU, and OPT schemes as well as the UBM scheme.

Simulations





Simulations

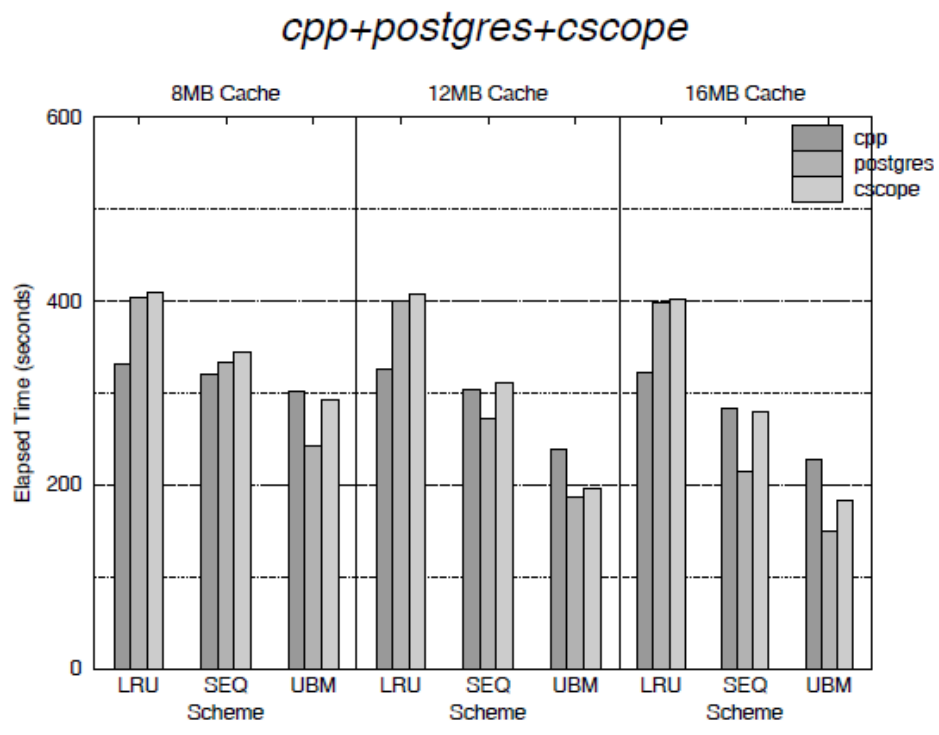


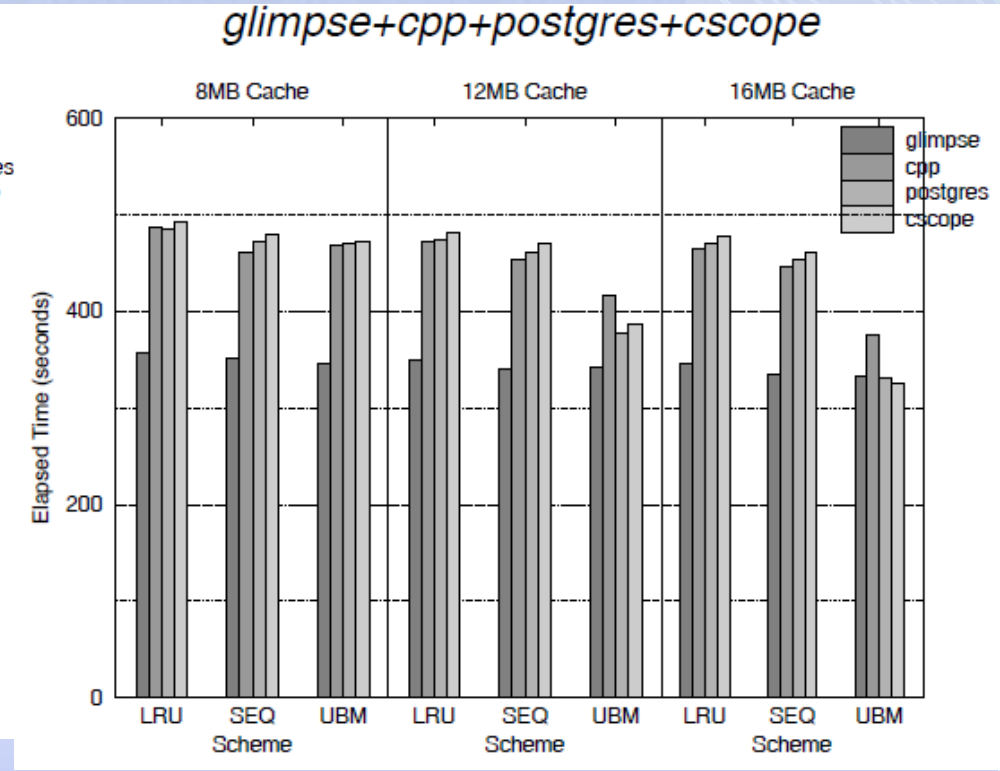
Implementation

- Settings
  - In the FreeBSD operation system
  - On a 133MHz Intel Pentium PC with 128MB RAM and a 1.6GB Quantum Fireball hard disk.
  - Cache size : 8MB,12MB and 16MB

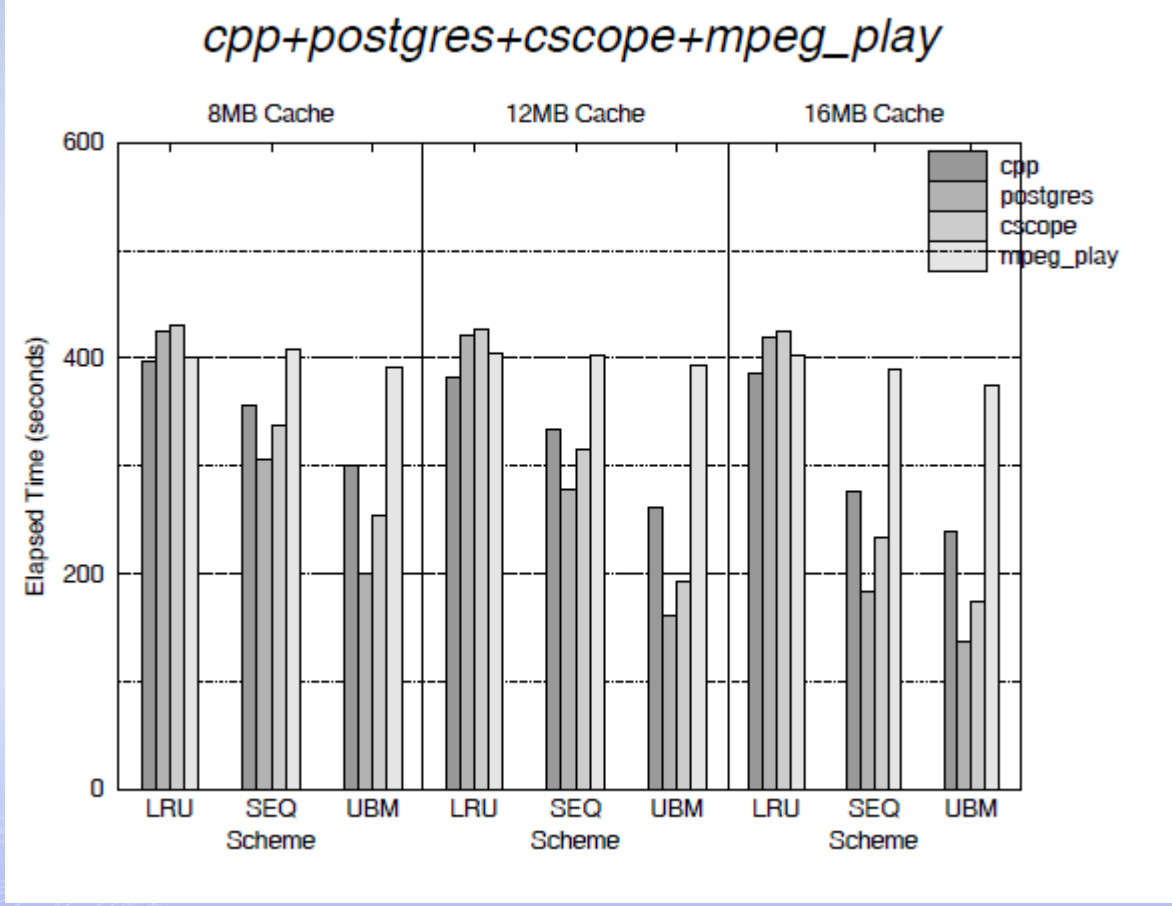
    with block size set to 8KB

Implementation





Implementation



Conclusions

- The UBM scheme shows substantial performance improvements increasing the buffer hit ratio and the elapsed time by compared to the LRU scheme.

- We are attempting to apply to other references the Least Recently/Frequently Used (LRFU) scheme based on both recency and frequency factors rather than the LRU scheme.