

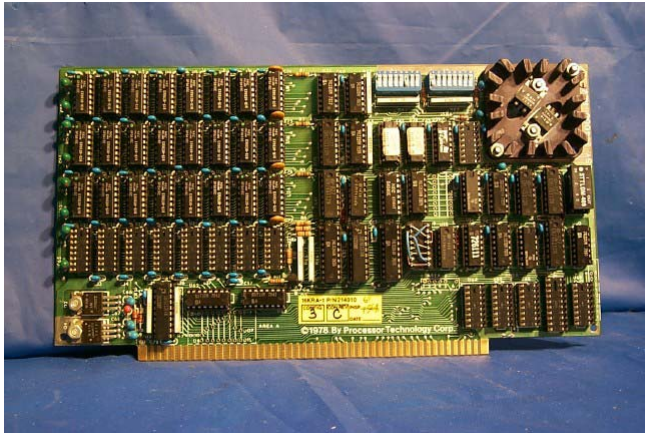


FAST 2004, San Francisco, CA

CAR: CLOCK with ADAPTIVE REPLACEMENT

Sorav Bansal, Stanford University
Dharmendra S. Modha, IBM Research

The Caching Problem



expensive, but fast







cheap, but slow

How to manage the cache?

Assume demand paging: Which page to replace?

How to maximize the hit ratio?

	L R U
Recency	
Constant Time	

	L R U
Recency	
Constant Time	
Scan Resistance	
“Frequency”	

	L R U	L F U
Recency	😊	
Constant Time	😊	
Scan Resistance		😊
“Frequency”		😊

	L R U	L F U	A R C
Recency	😊		😊
Constant Time	😊		😊
Scan Resistance		😊	😊
“Frequency”		😊	😊

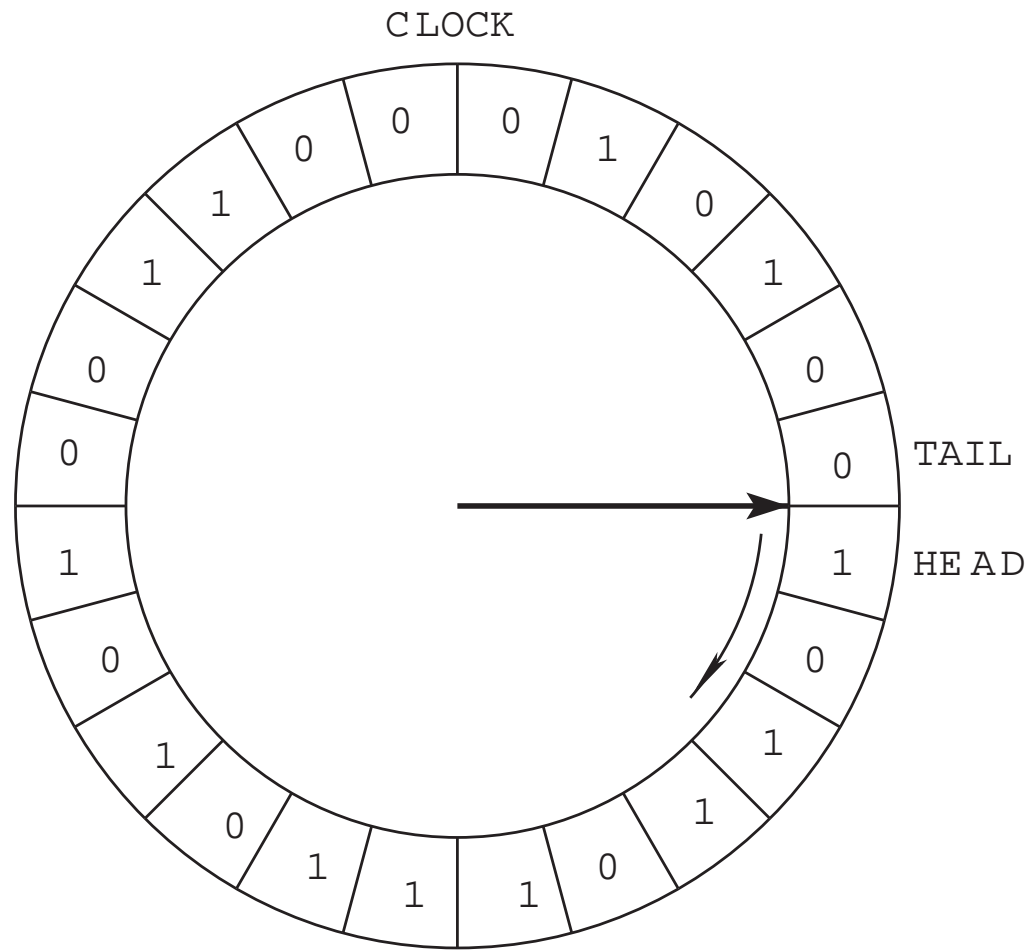
	L R U	L F U	A R C
Recency	😊		😊
Constant Time	😊		😊
Scan Resistance		😊	😊
“Frequency”		😊	😊
Lock Contention/ MRU Overhead			

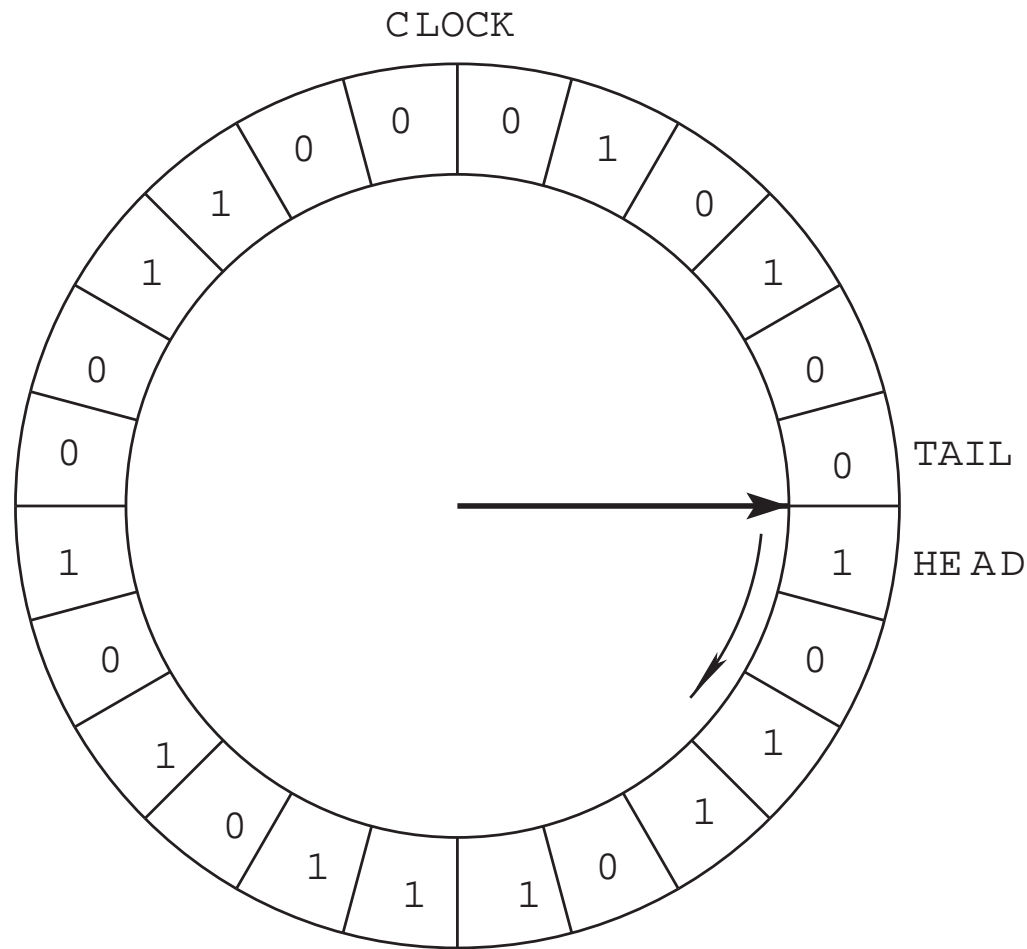


“In Multics a paging algorithm has been developed that has the implementation ease and low overhead of the FIFO and is an approximation to LRU.”

Fernando J. Corbato, 1990 Turing Award Winner

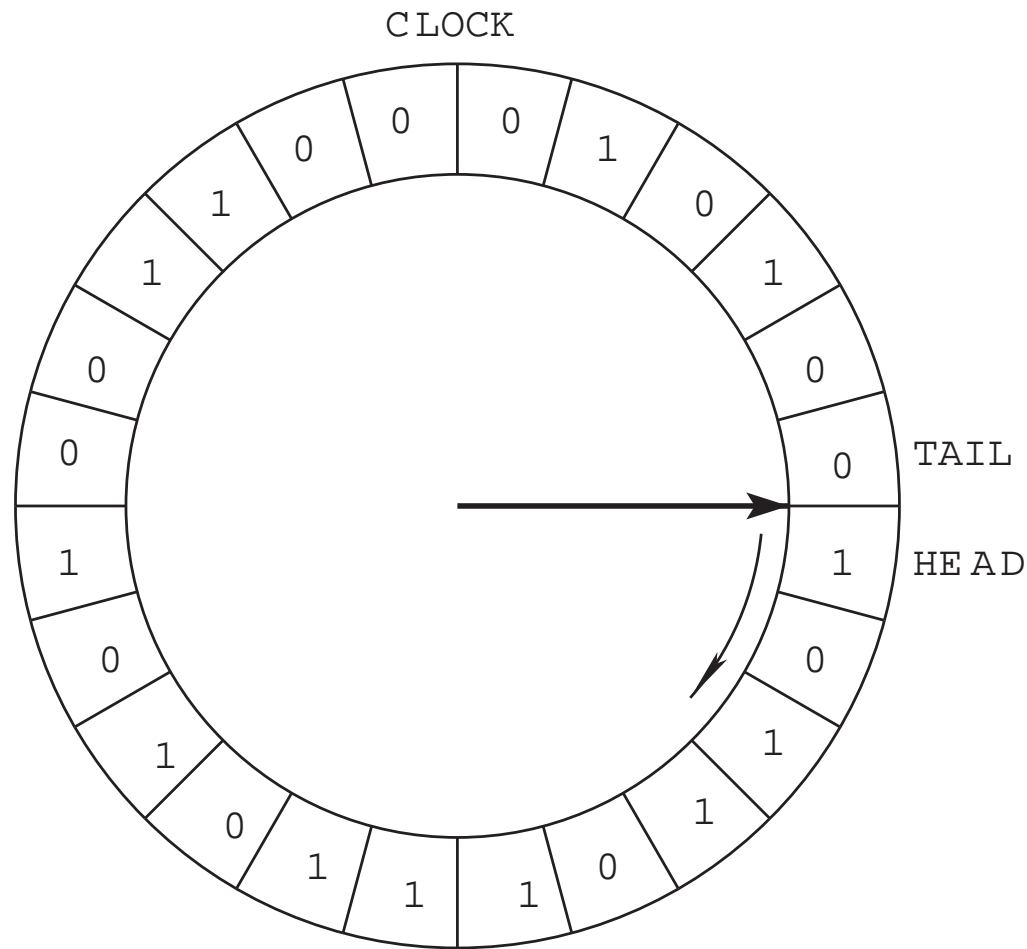
	L R U	L F U	A R C	C L O C K
Recency	😊		😊	😊
Constant Time	😊		😊	😊
Scan Resistance		😊	😊	
“Frequency”		😊	😊	
Lock Contention/ MRU Overhead				😊





HIT: Set the reference bit to “1”

MISS: Insert at the TAIL, initialize the reference bit to “0”



REPLACEMENT POLICY: Evict the first “0” page

Reset “1” pages to “0”—“second chance”—“delayed MRU”

CLOCK Applications and Exposition

- **Multics**
- **UNIX/AIX/LINUX/BSD**
- **VAX/VMS**
- **DB2**
- **Oracle? Windows?
Solaris?**
- **Major OS Textbooks**
 - Tanebaum & Woodhull
 - Silberschatz & Galvin

Prior Work on LRU versus CLOCK

▪ LRU

- (LFU)
- FBR
- LRU-2
- 2Q
- LRFU
- LIRS
- MQ
- ARC

▪ CLOCK

- GCLOCK
- Two-handed CLOCK

Prior Work on LRU versus CLOCK

▪ LRU

- (LFU)
- FBR
- LRU-2
- 2Q
- LRFU
- LIRS
- MQ
- ARC

▪ CLOCK

- GCLOCK
- Two-handed CLOCK



1968

Prior Work on LRU versus CLOCK

▪ LRU

- (LFU)
- FBR
- LRU-2
- 2Q
- LRFU
- LIRS
- MQ
- ARC

▪ CLOCK

- GCLOCK
- Two-handed CLOCK



1968

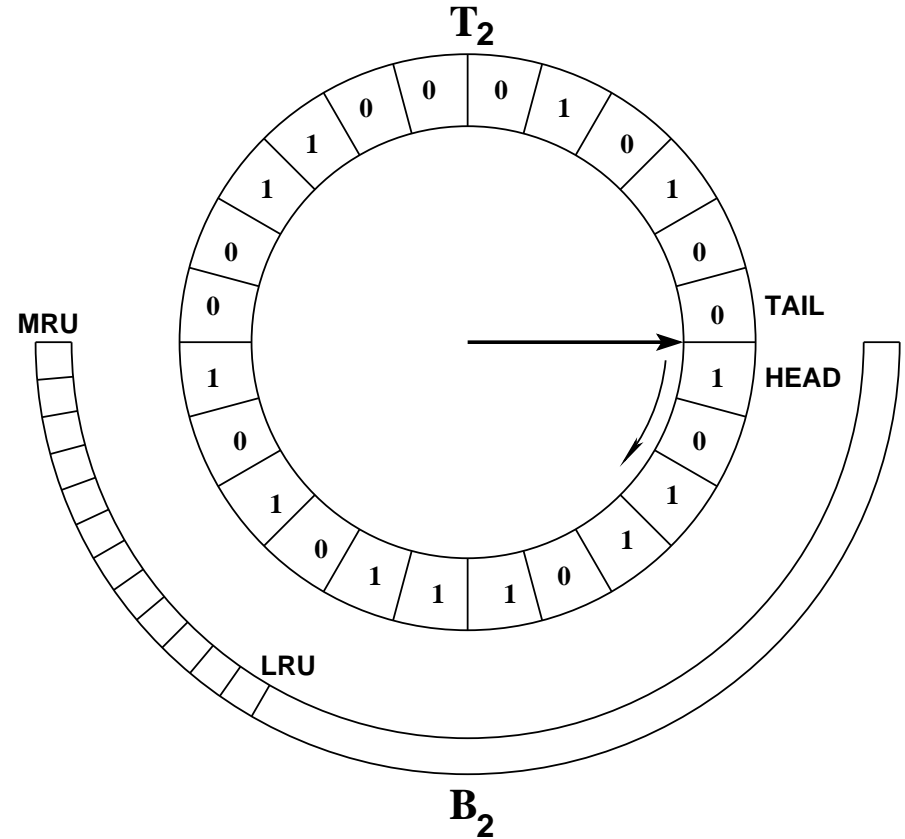
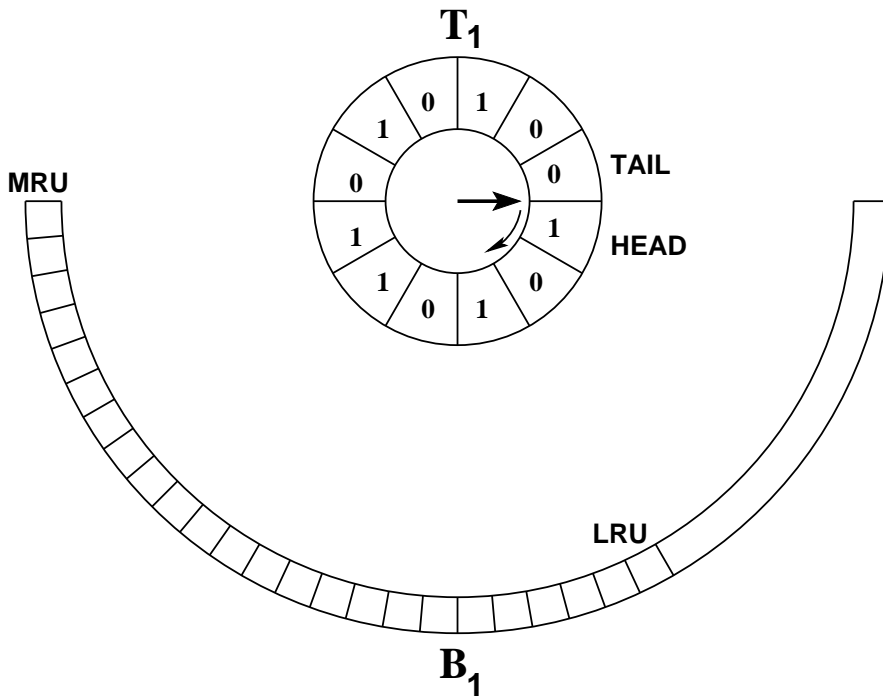


1998

	L R U	L F U	A R C	C L O C K	C A R
Recency	😊		😊	😊	😊
Constant Time	😊		😊	😊	😊
Scan Resistance		😊	😊		😊
“Frequency”		😊	😊		😊
Lock Contention/ MRU Overhead				😊	😊

"Recency"

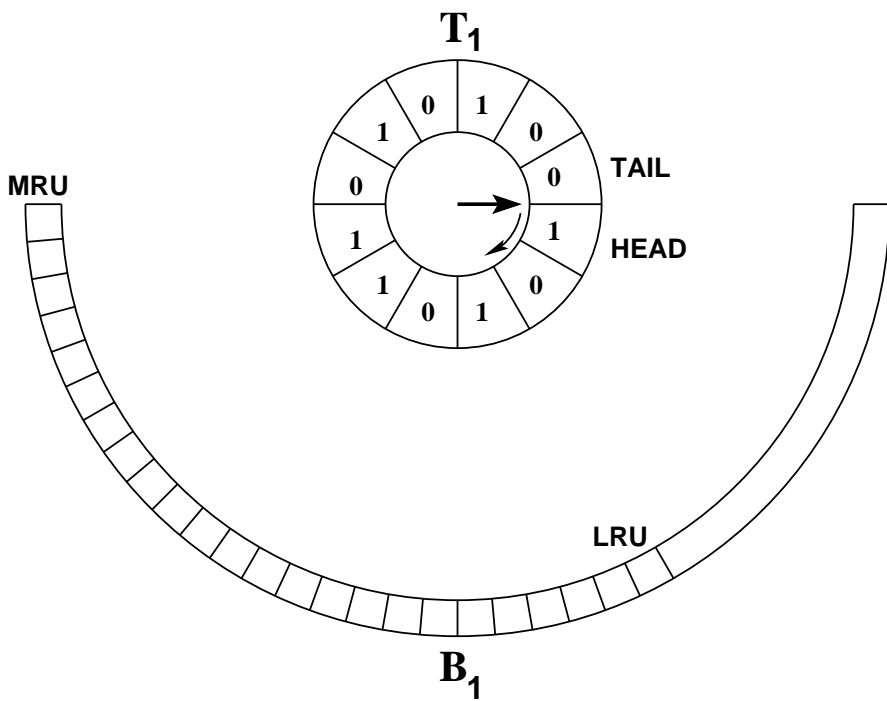
"Frequency"



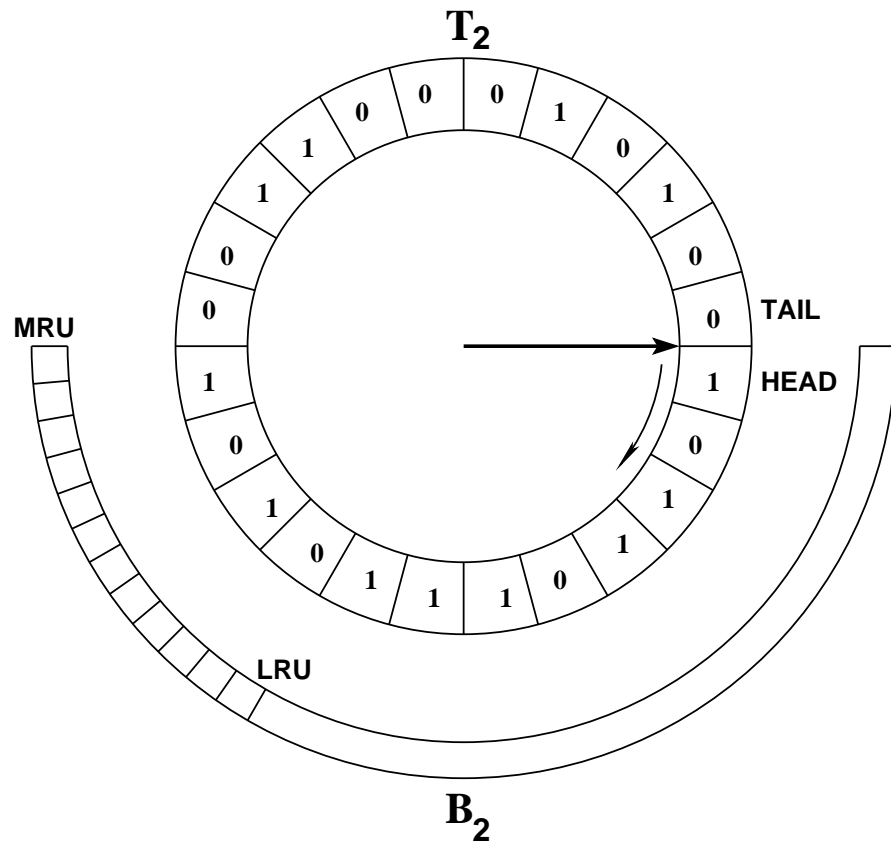
CLOCKs T1 and T2 contain cache pages

LRU lists B1 and B2 contain recently evicted history pages

"Recency"



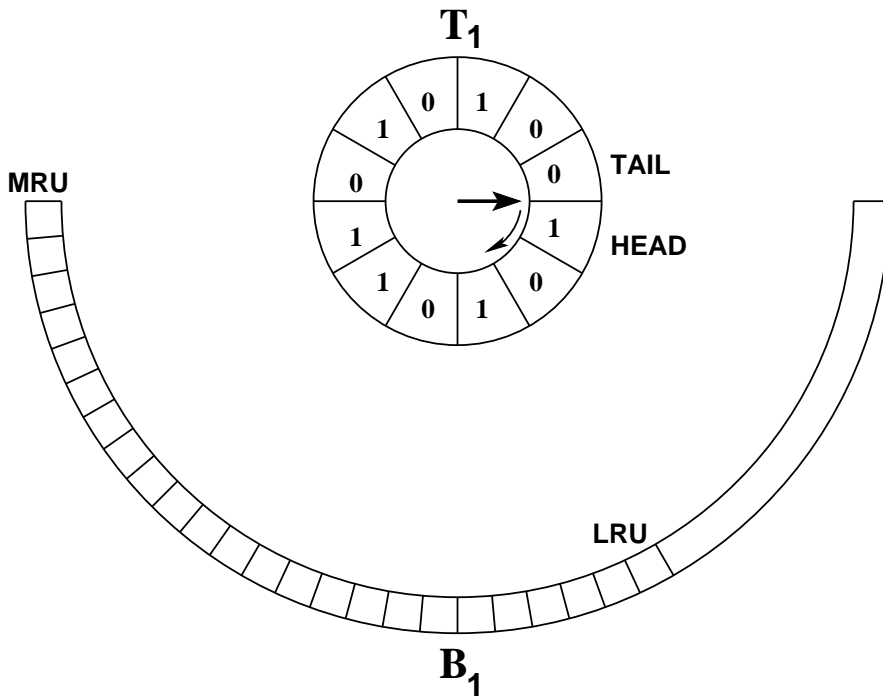
"Frequency"



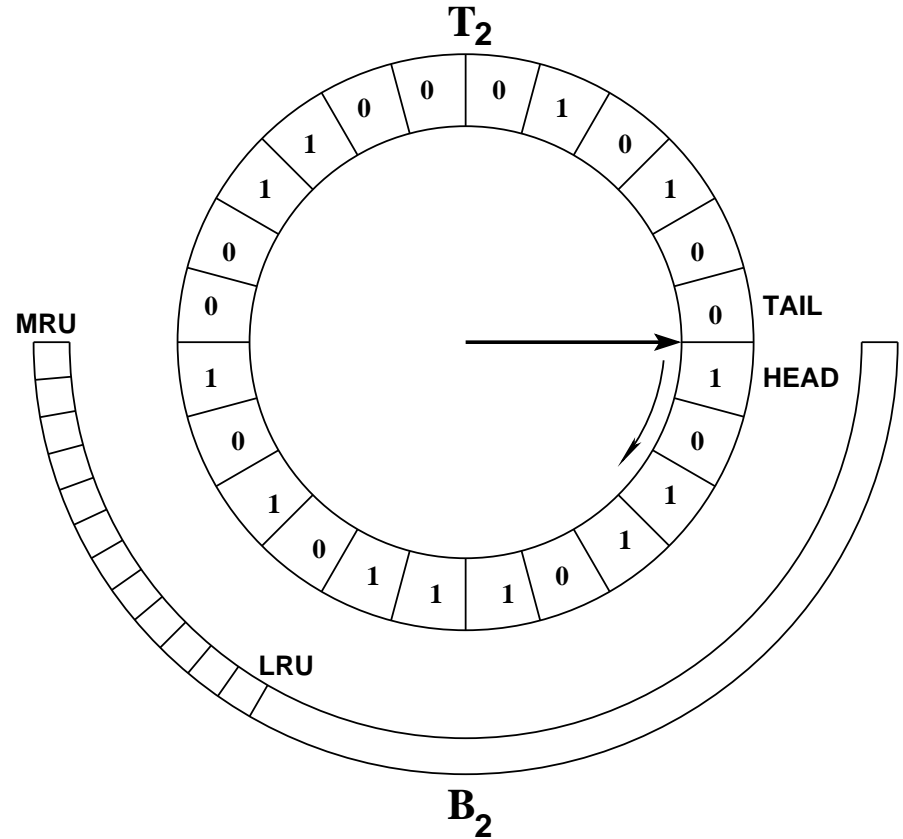
Size of T1 roughly equals B2

Size of T2 roughly equals B1

"Recency"

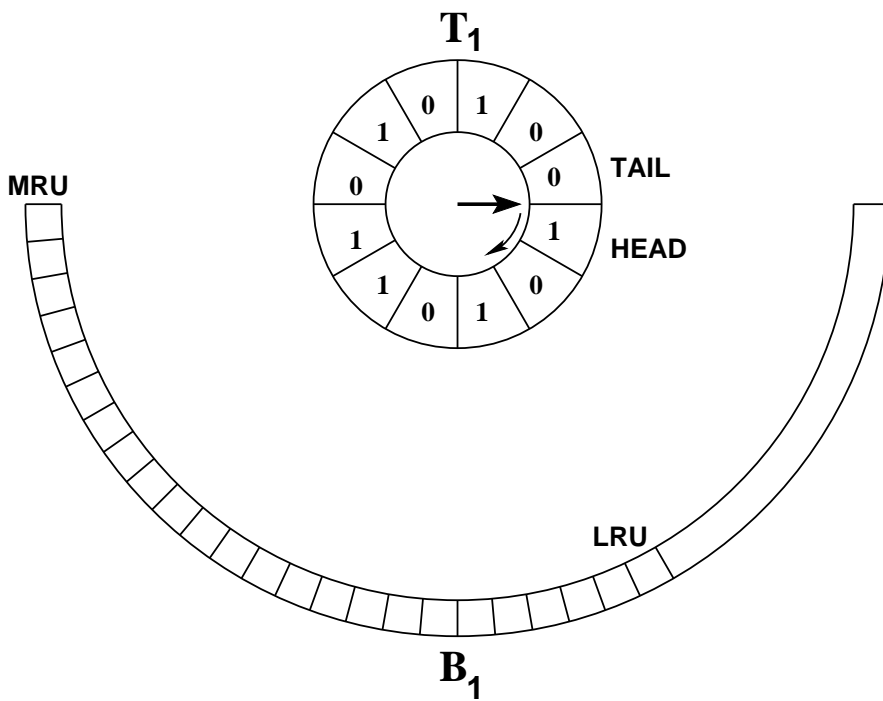


"Frequency"

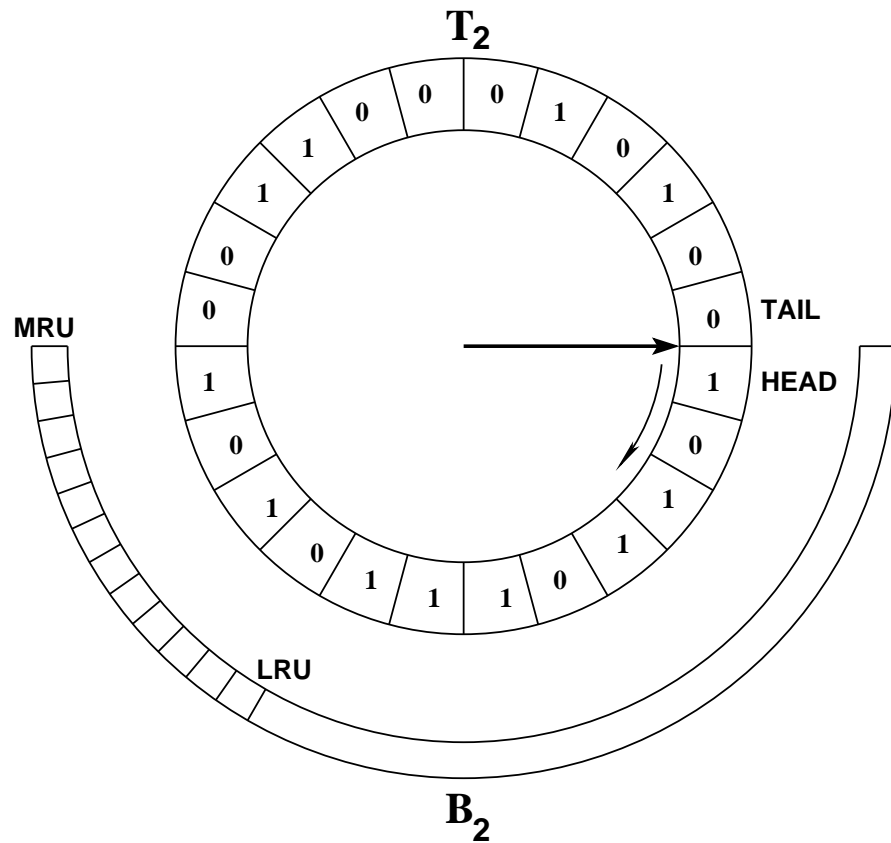


T1(0) & B1 pages seen exactly once recently
T1(1), T2, & B2 pages seen at least twice recently

"Recency"

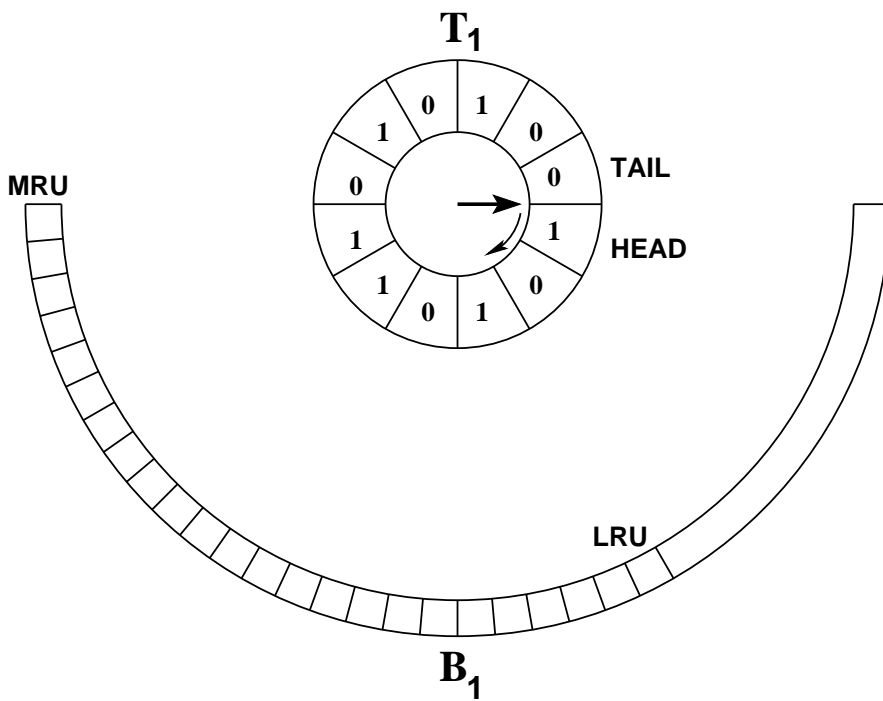


"Frequency"

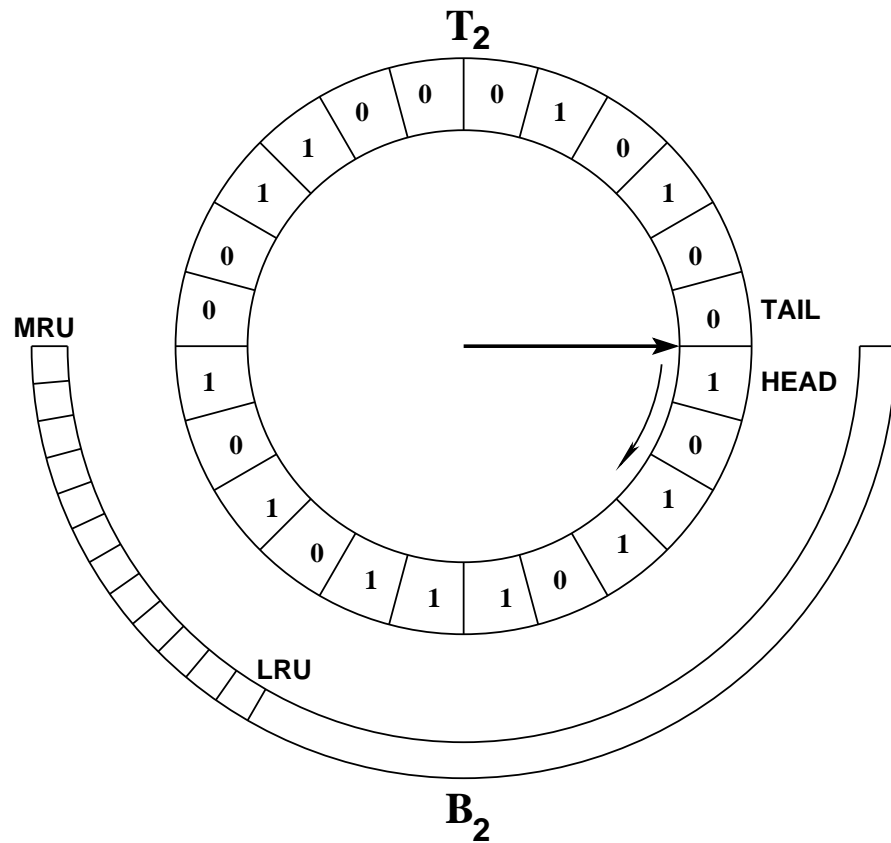


Maintain a target size for CLOCK T1

"Recency"

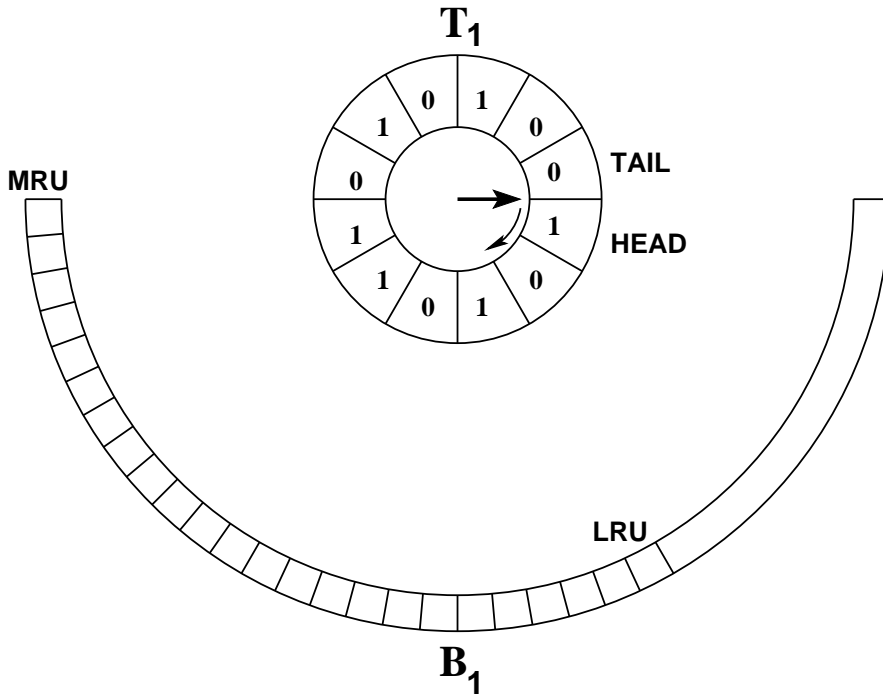


"Frequency"

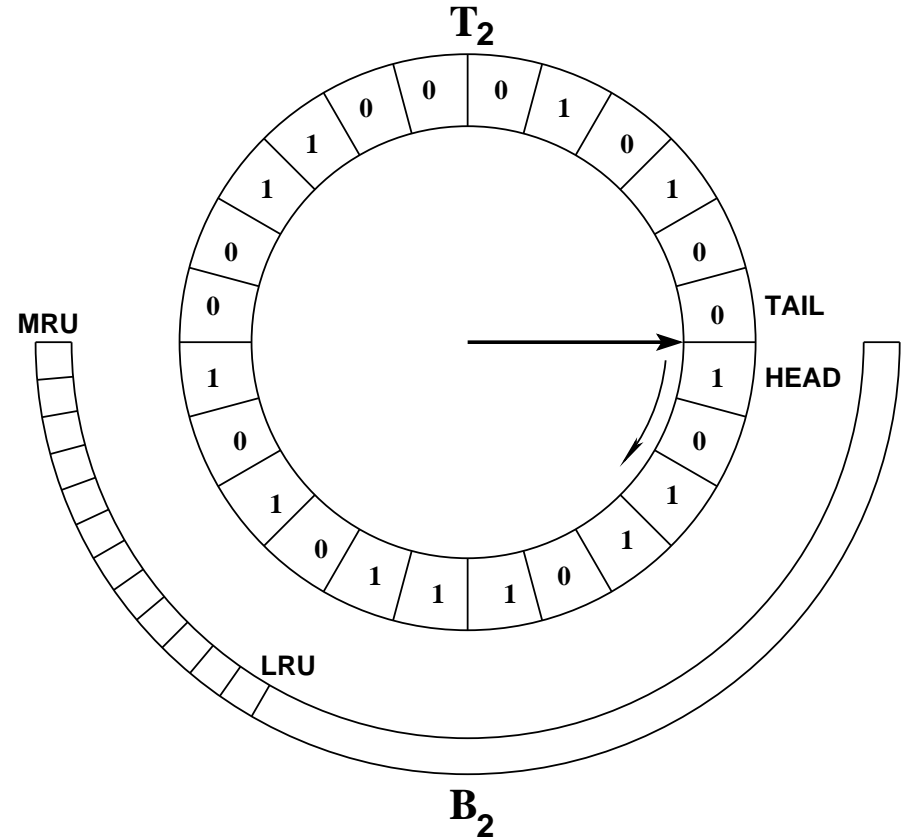


HIT in T1 or T2: Set reference bit to "1"

"Recency"

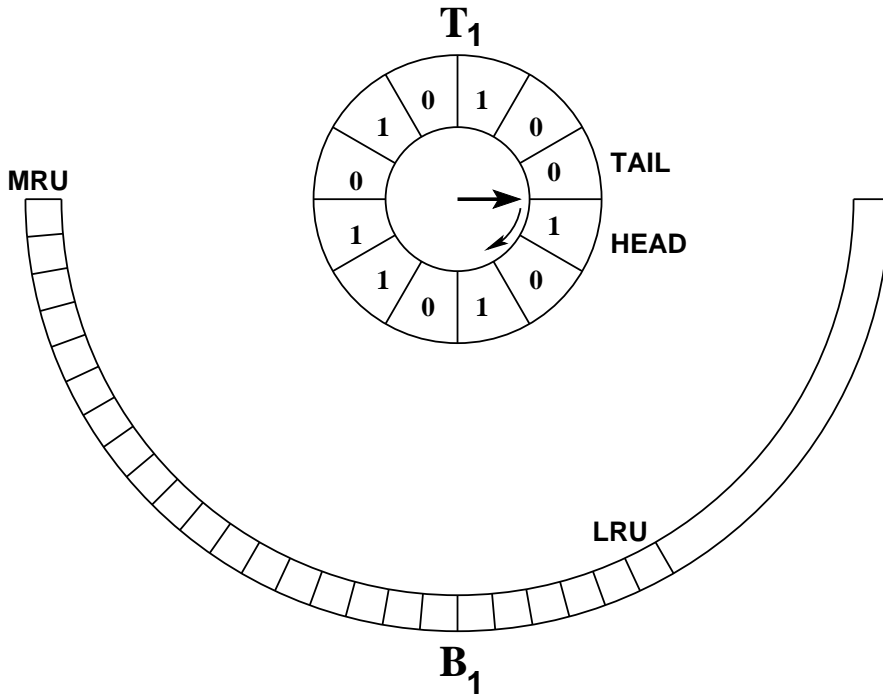


"Frequency"

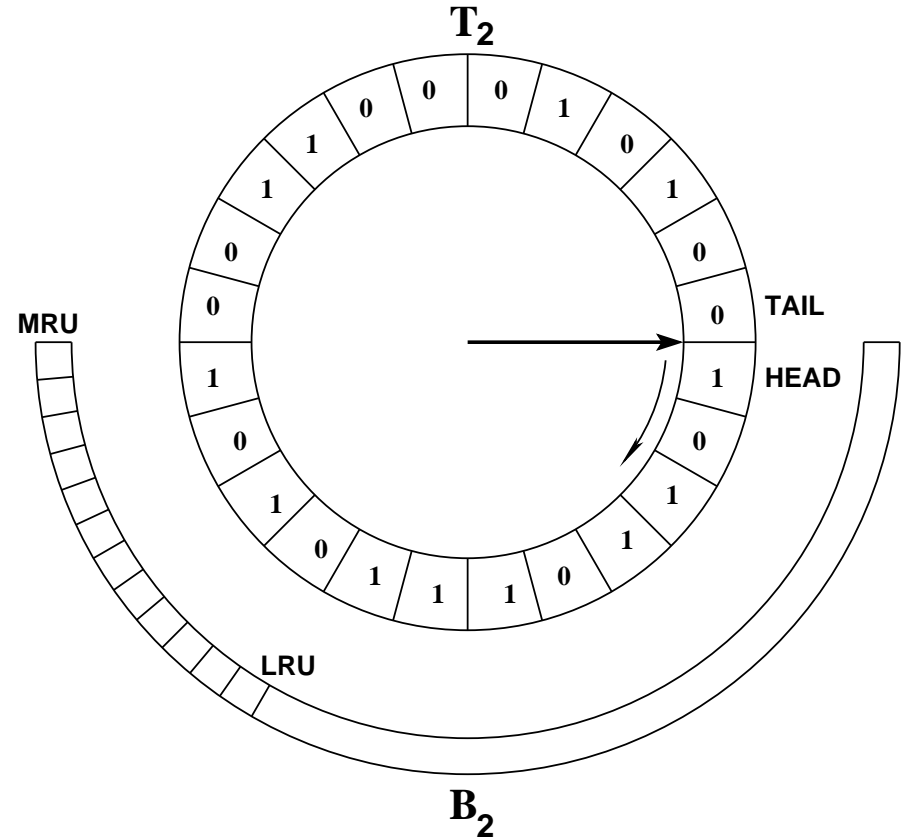


MISS in B1: Set reference bit to "0", move to TAIL of T2, and increase target size of T1

"Recency"

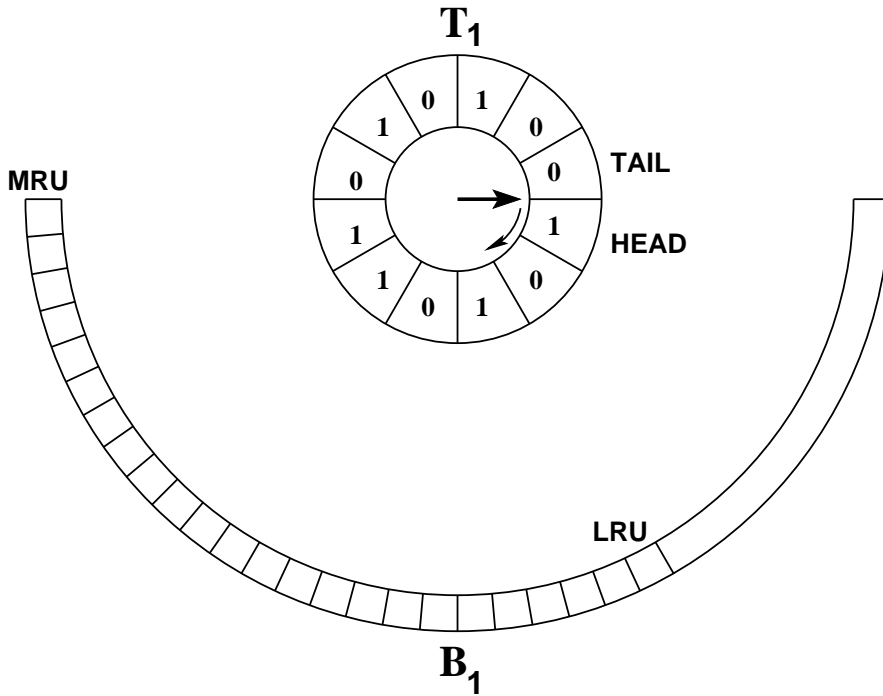


"Frequency"

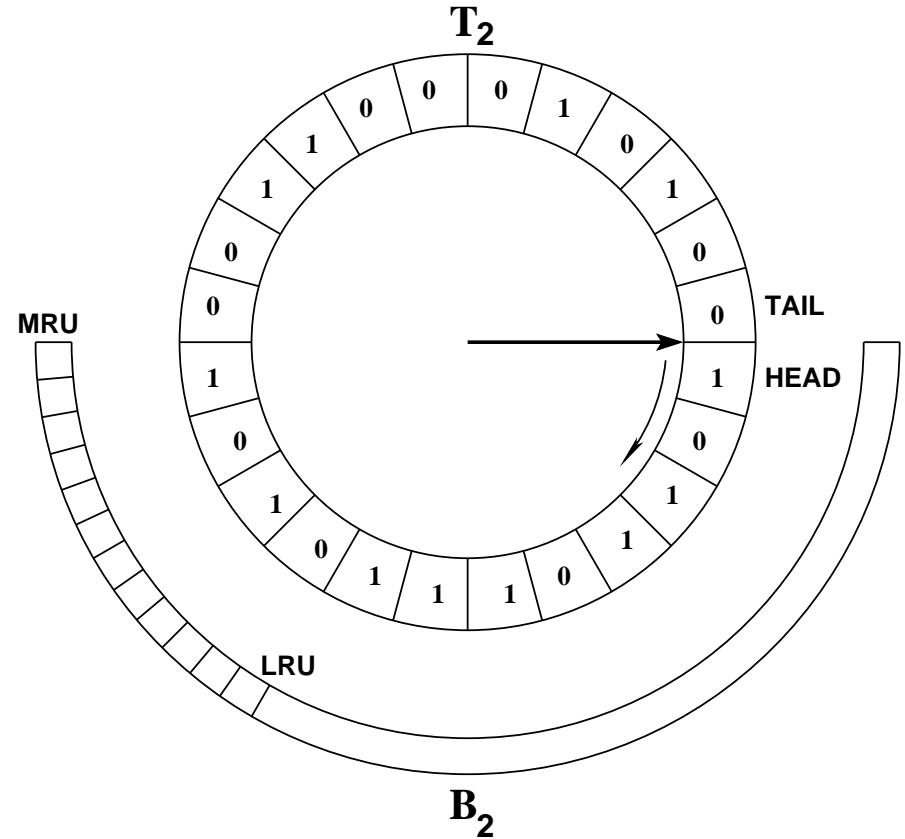


MISS in B2: Set reference bit to "0", move to TAIL of T2, and decrease target size of T1

"Recency"

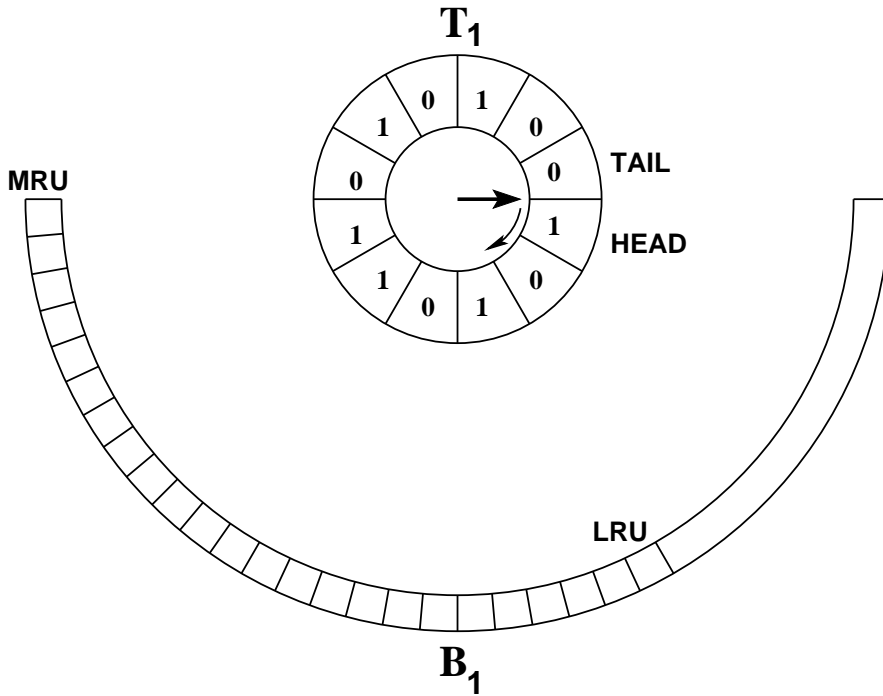


"Frequency"

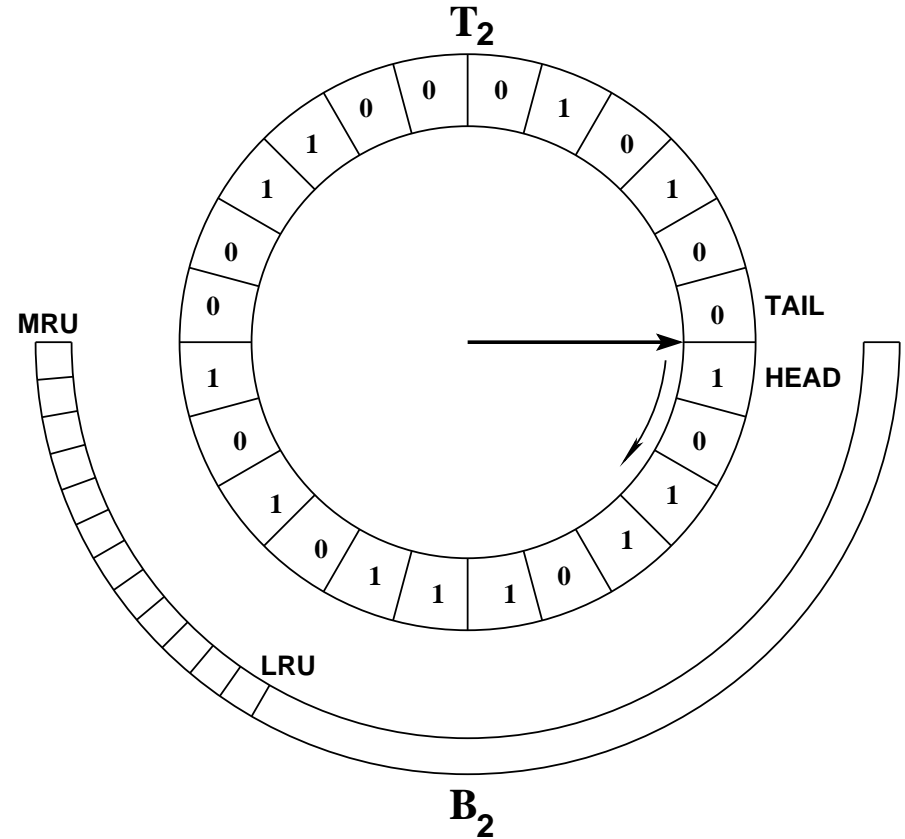


TOTAL MISS: Set reference bit to "0", move to TAIL of T1

"Recency"



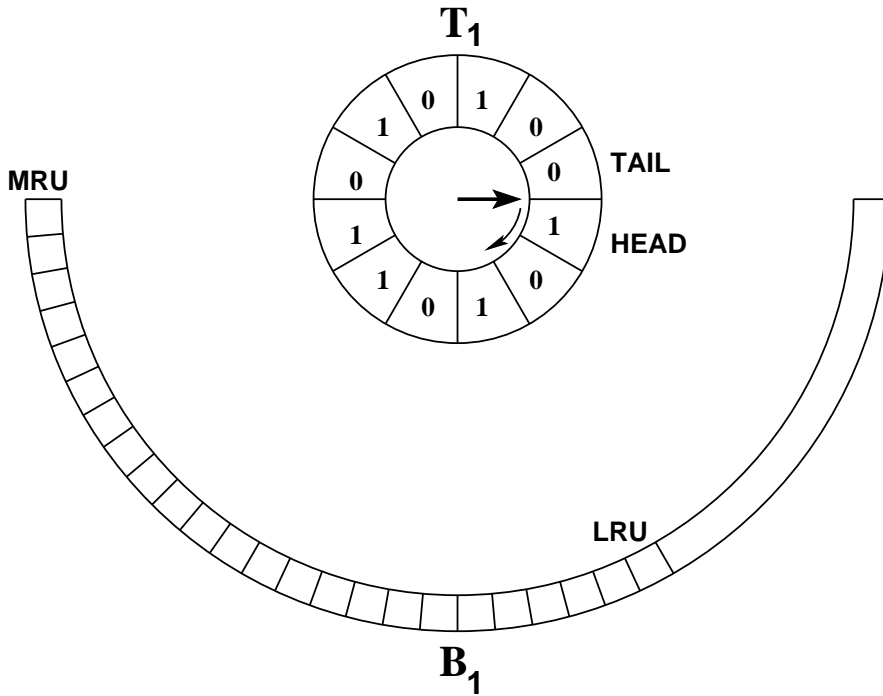
"Frequency"



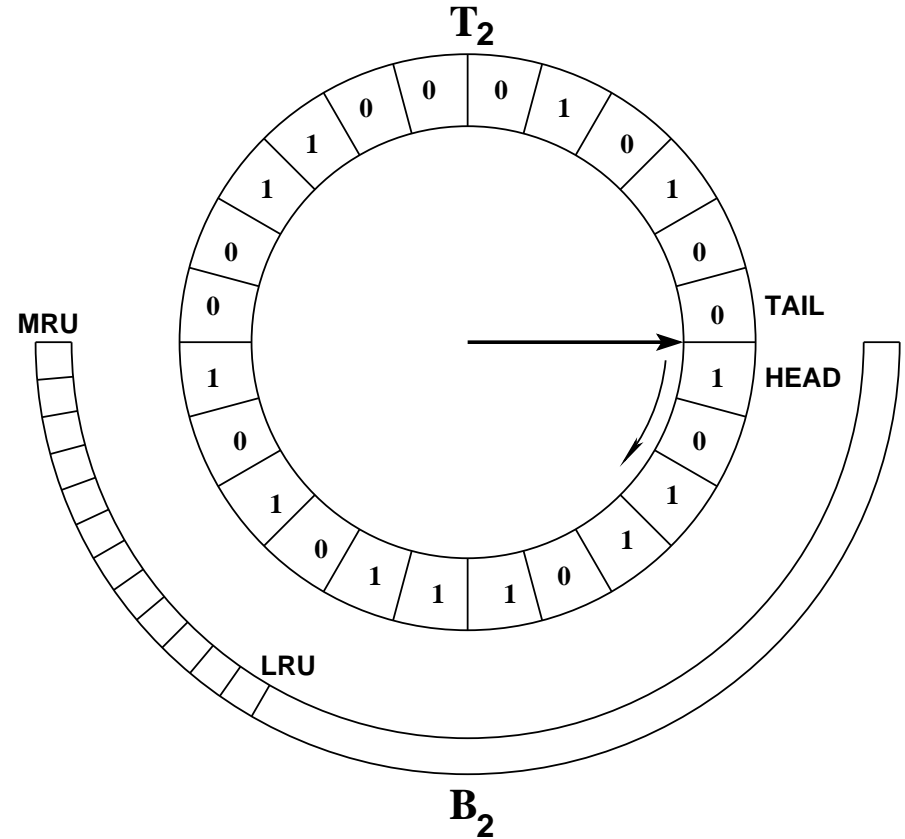
CACHE REPLACEMENT POLICY:

Replace from T1 if larger than target; else from T2

"Recency"

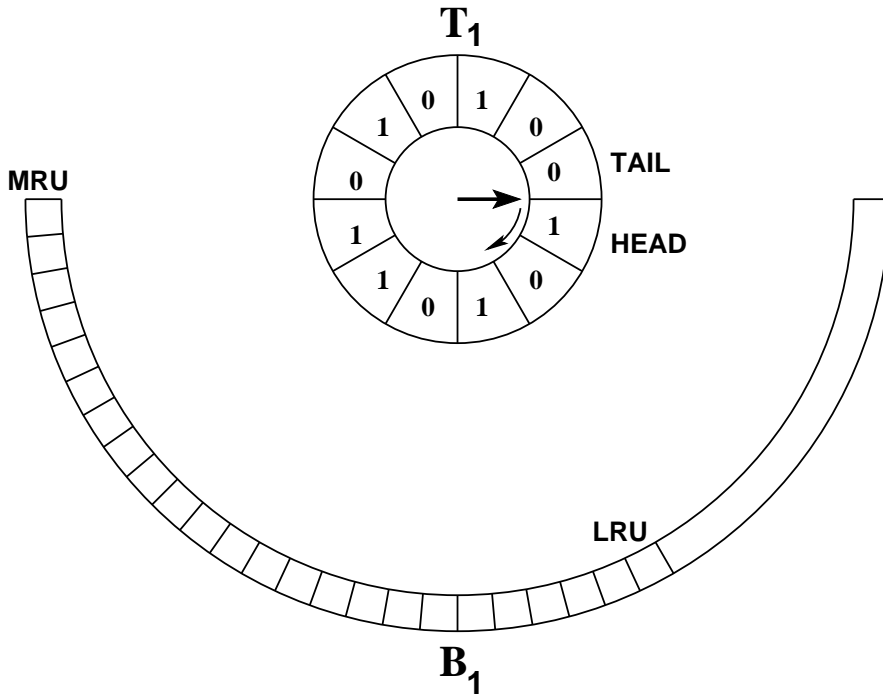


"Frequency"

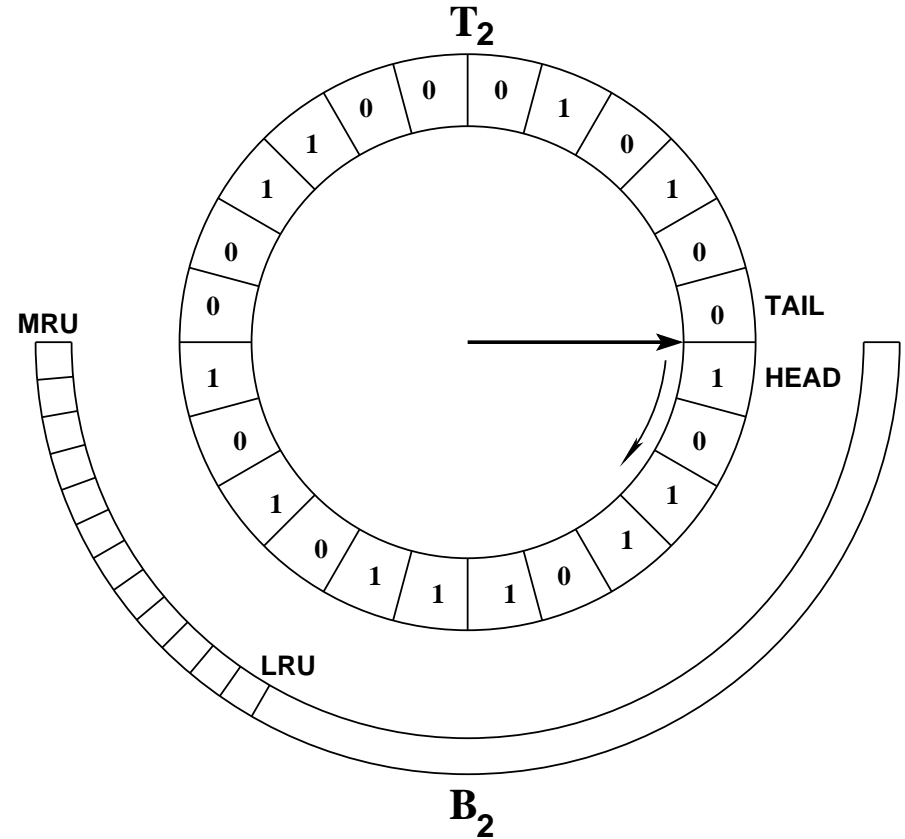


During replacement in T1, if "1" page is found, make "0" and move to T2 TAIL, move evicted page to B1

"Recency"

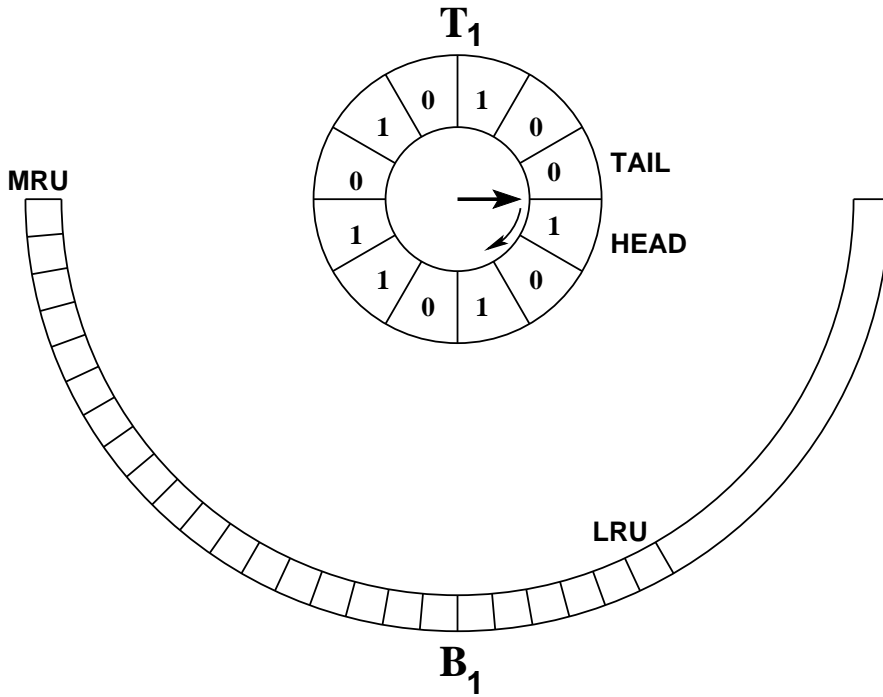


"Frequency"

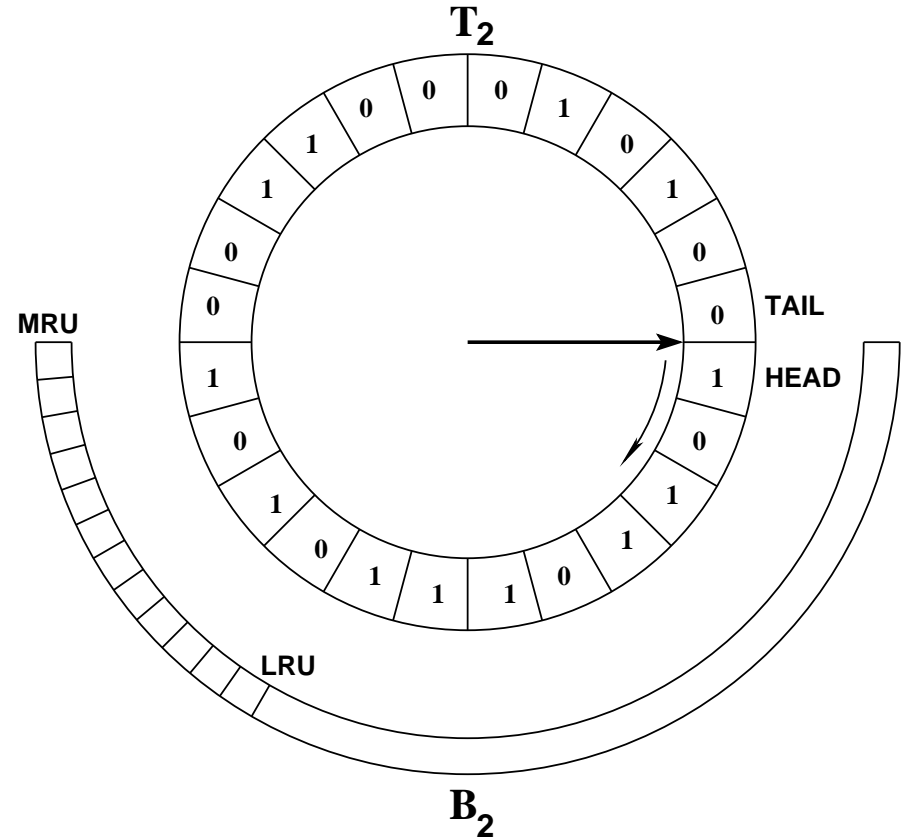


During replacement in T2, if "1" page is found, make "0", move evicted page to B2

"Recency"

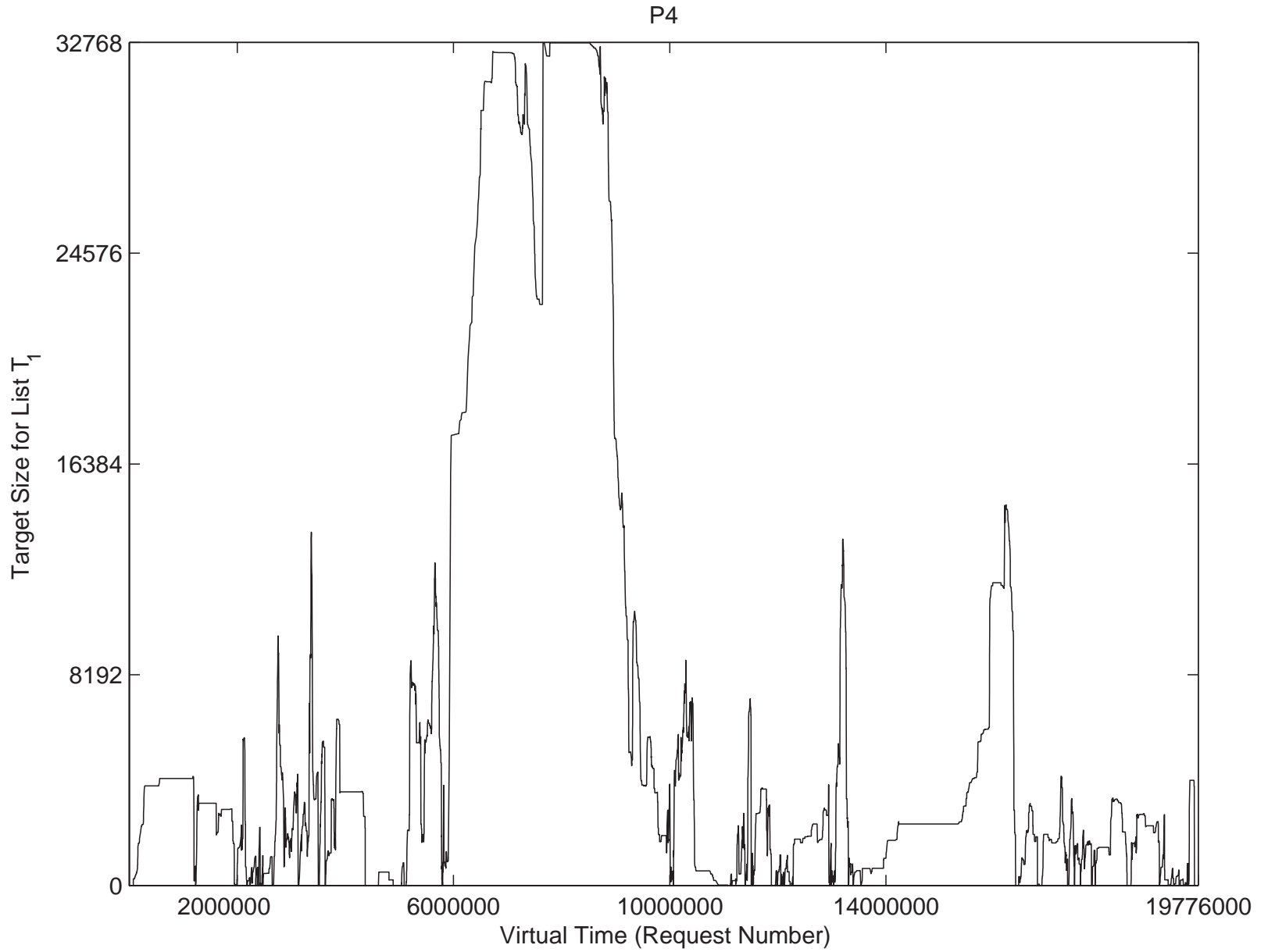


"Frequency"



DIRECTORY REPLACEMENT POLICY:

Replace from B1 if $T1+B1 = c$; else from B2



CART = CAR + Temporal Filtering

- **CAR/ARC: two hits to a page is a criterion for promotion from T1 to T2**
- **CART: promotion from T1 to T2 happens only if two hits are “far”**

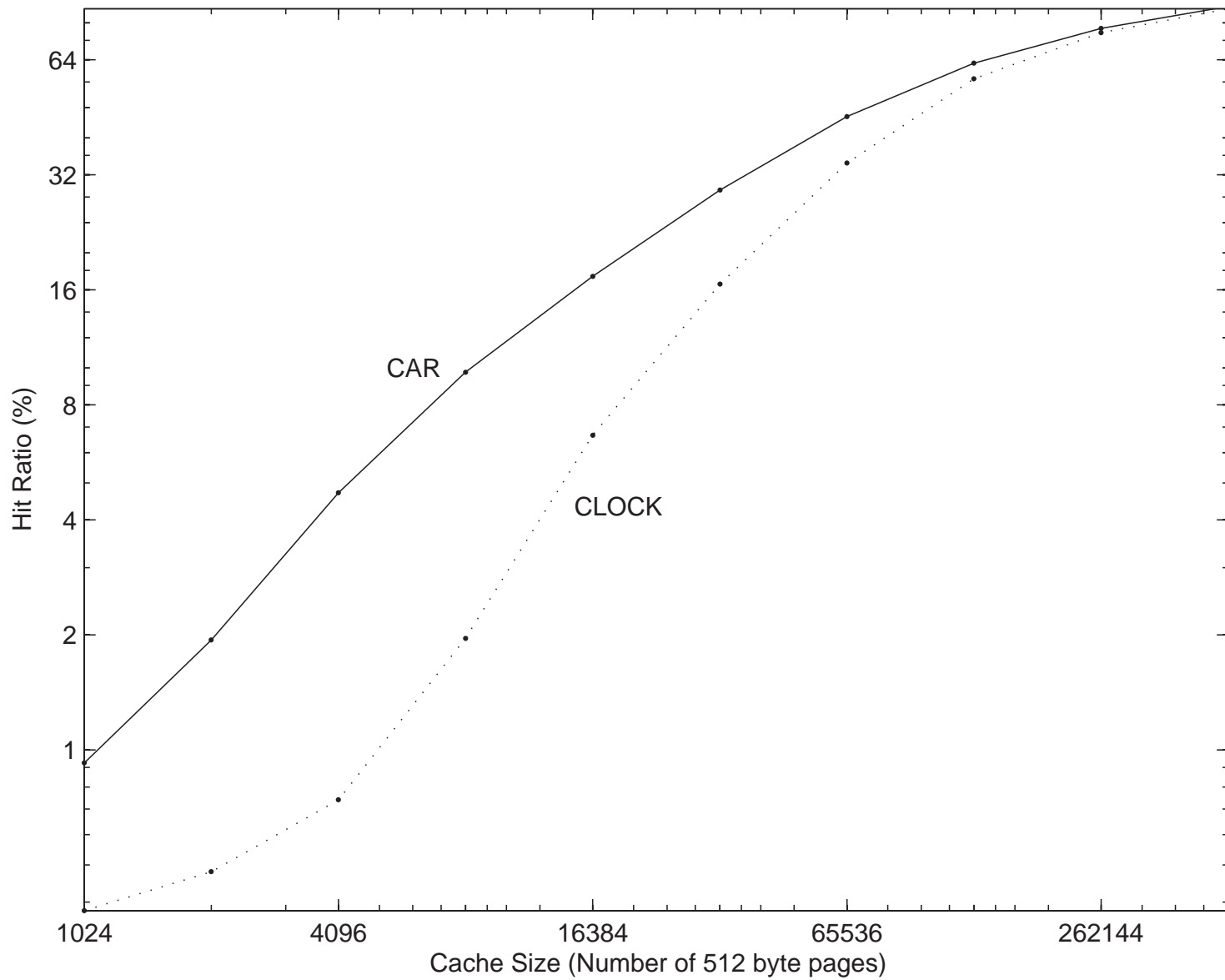
SPC-1 like Workload

Cache Size (4K pages)	LRU	CLOCK	ARC	CAR	CART
65536	0.37	0.37	0.82	0.84	0.90
131072	0.78	0.77	1.62	1.66	1.78
262144	1.63	1.63	3.23	3.29	3.56
524288	3.66	3.64	7.56	7.62	8.52
1048576	9.19	9.31	20.00	20.00	21.90

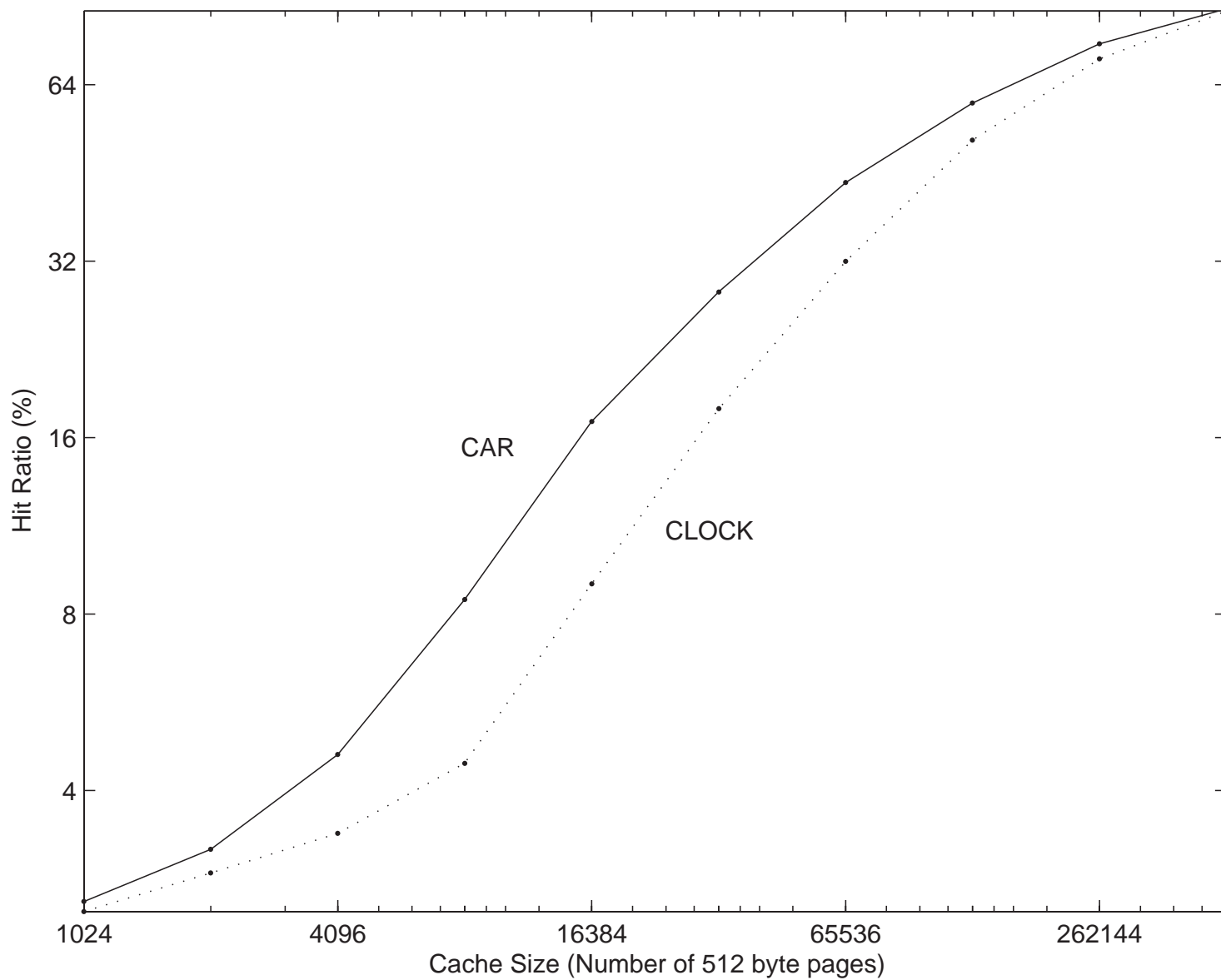
Merge(S) Workload

Cache Size (4k Pages)	LRU	CLOCK	ARC	CAR	CART
16384	0.20	0.20	1.04	1.03	1.10
32768	0.40	0.40	2.08	2.07	2.20
65536	0.79	0.79	4.07	4.05	4.27
131072	1.59	1.58	7.78	7.76	8.20
262144	3.23	3.27	14.30	14.25	15.07
524288	8.06	8.66	24.34	24.47	26.12
1048576	27.62	29.04	40.44	41.00	41.83
1572864	50.86	52.24	57.19	57.92	57.64
2097152	68.68	69.50	71.41	71.71	71.77
4194304	87.30	87.26	87.26	87.26	87.26

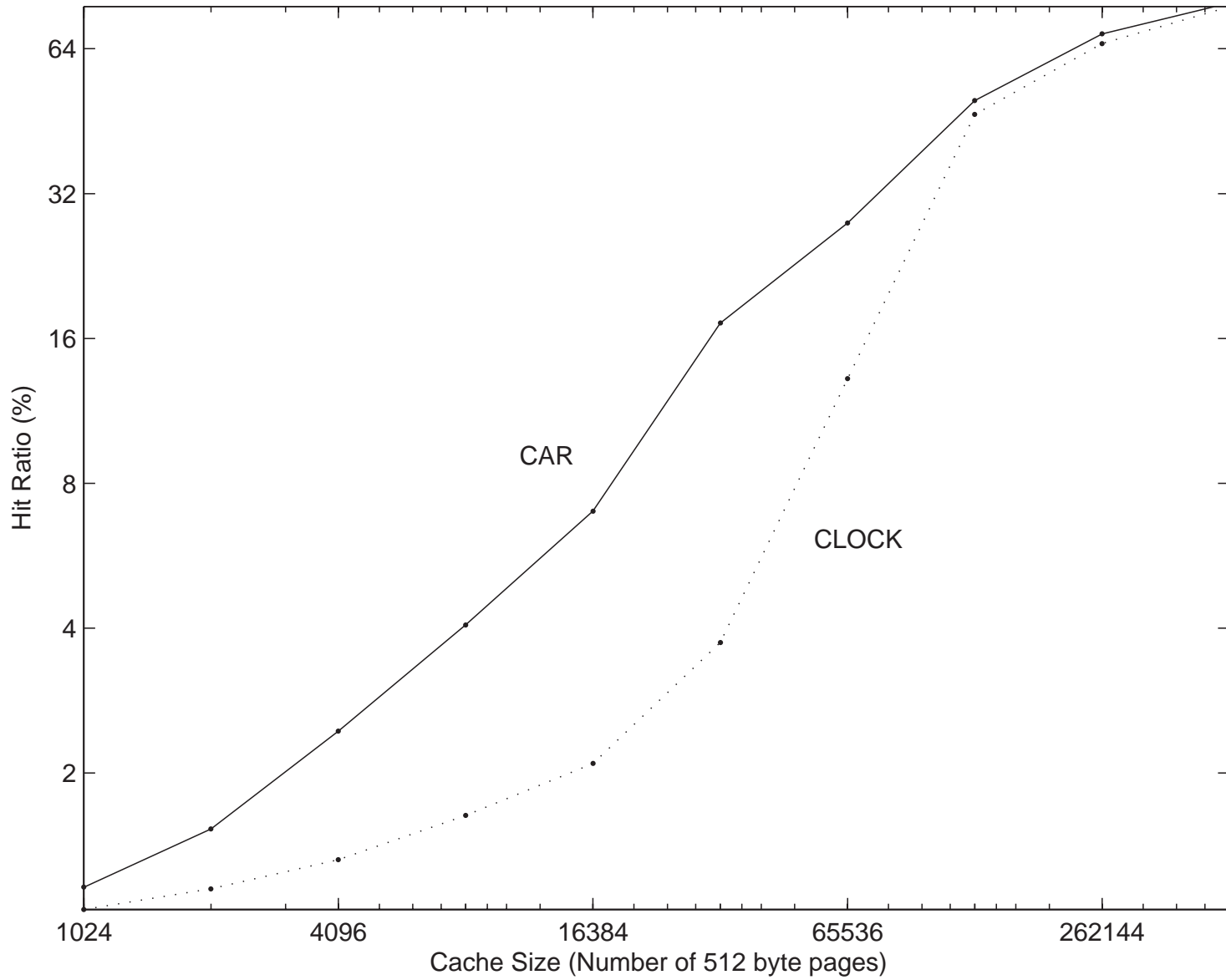
P1



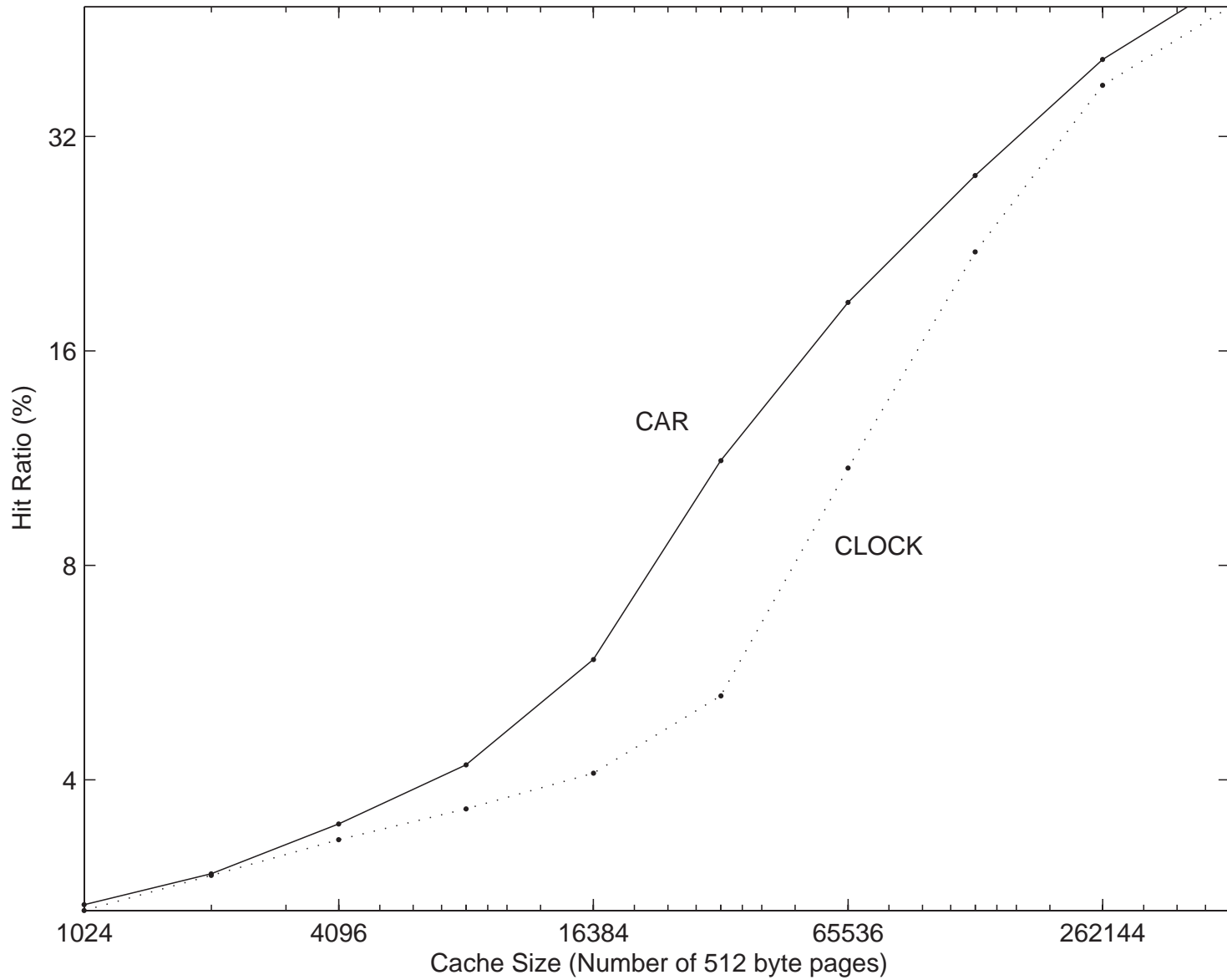
P2

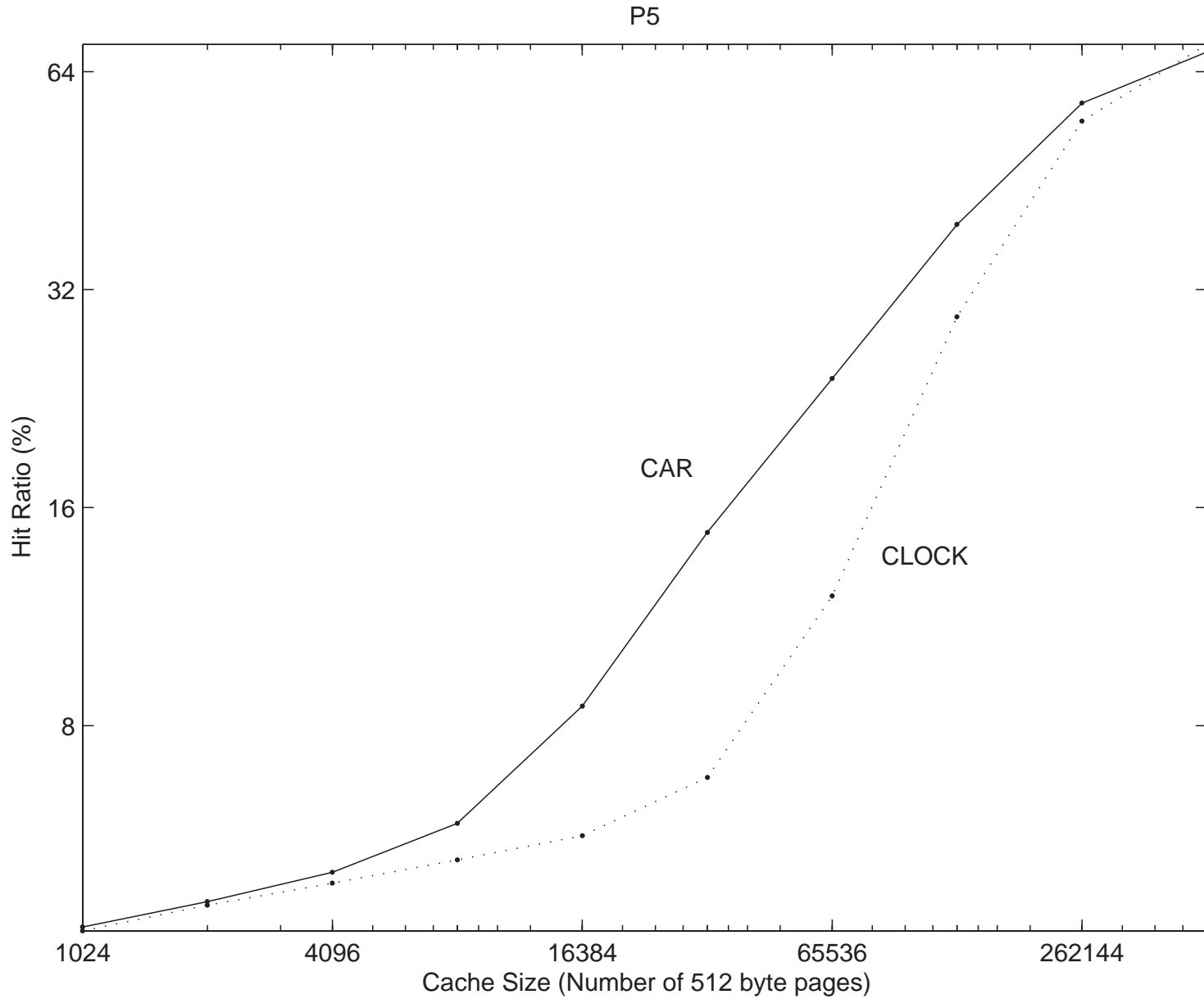


P3

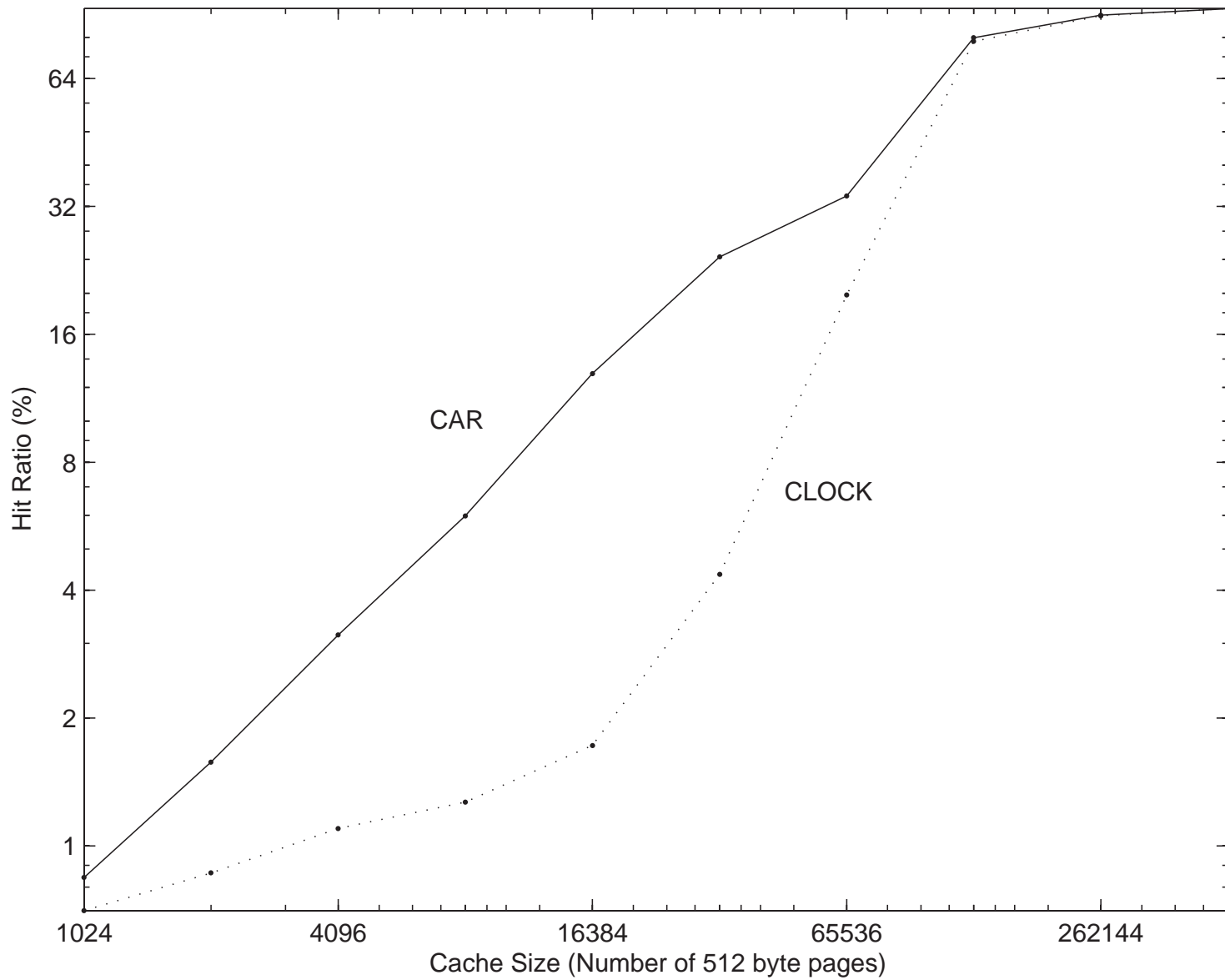


P4

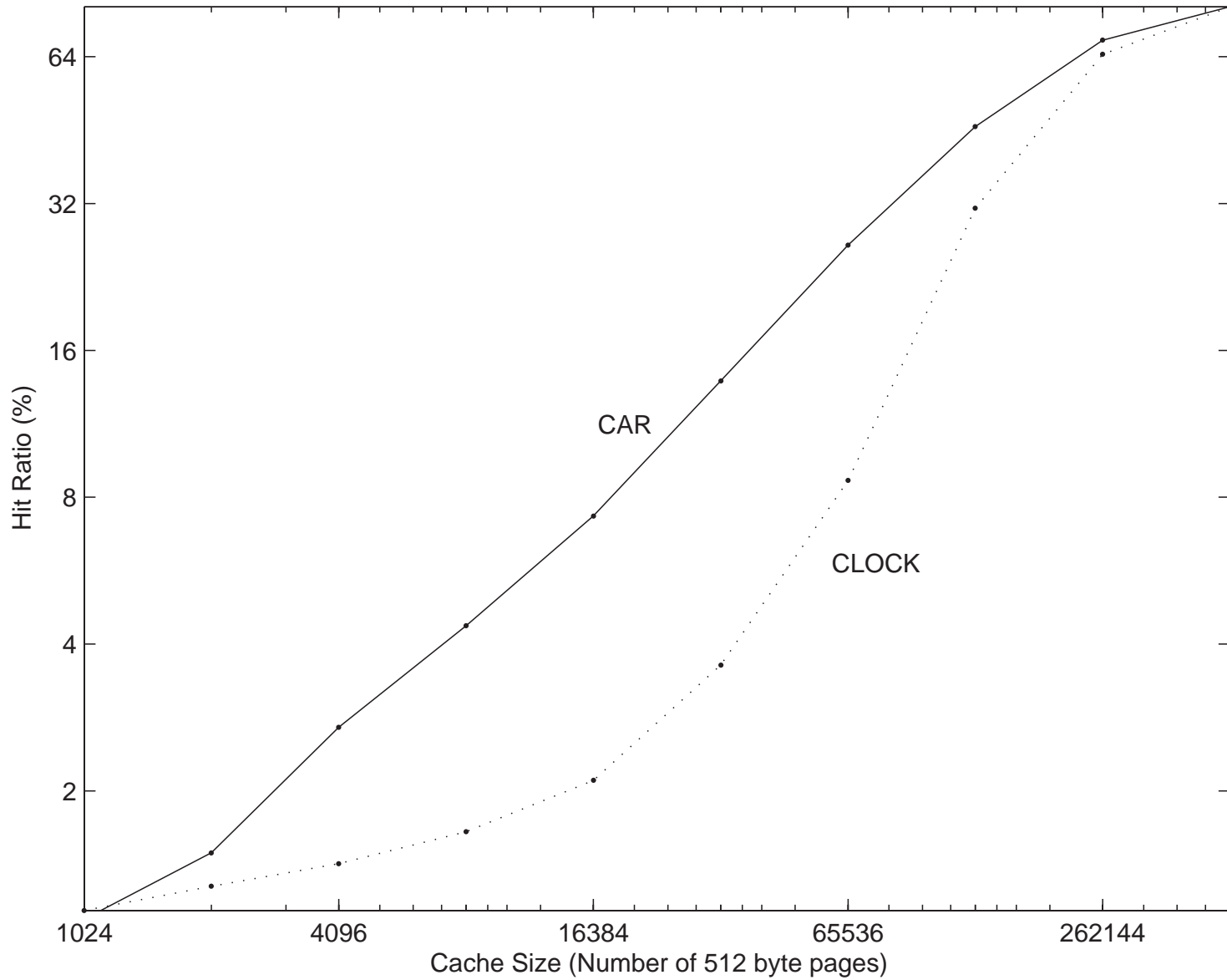


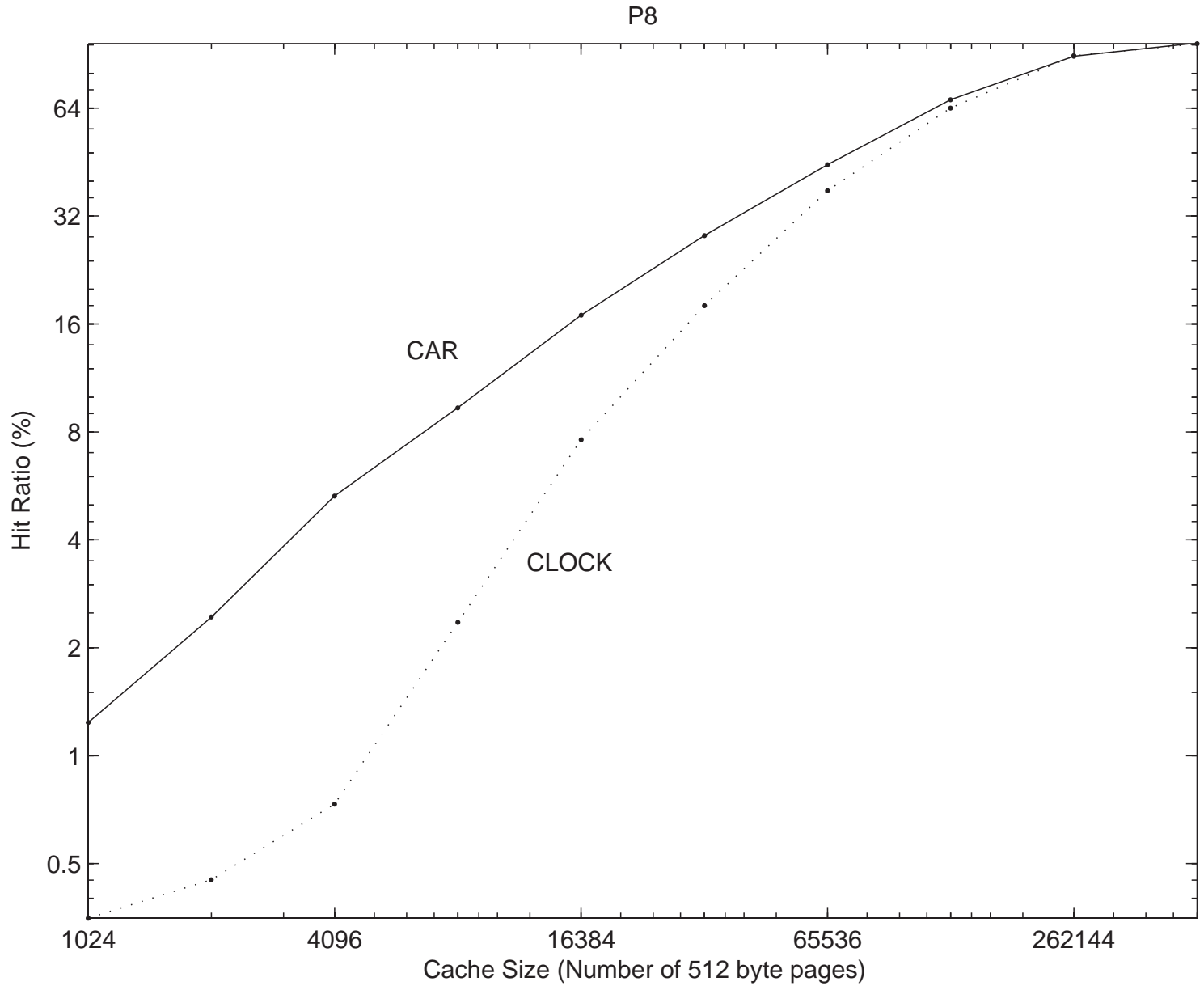


P6

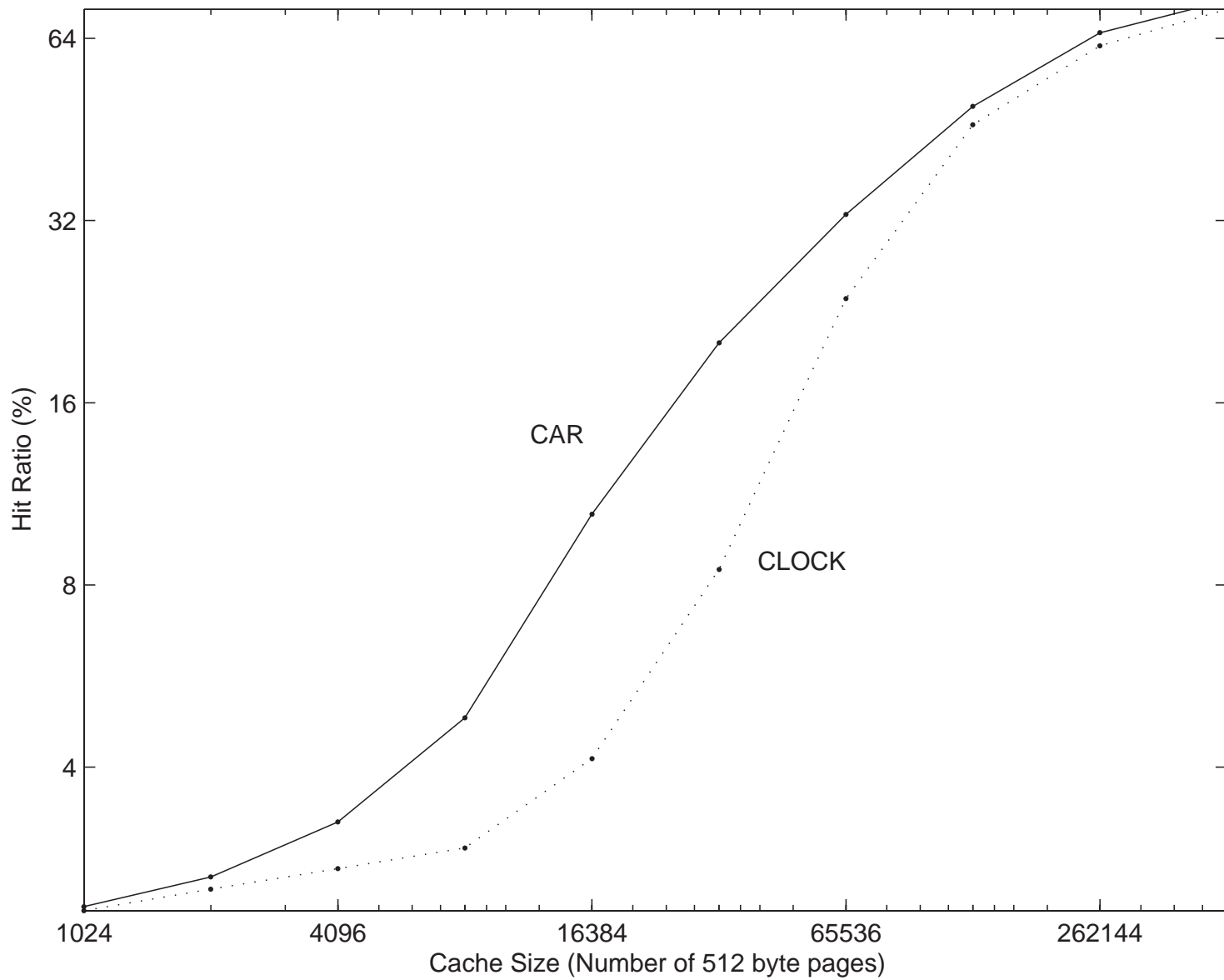


P7

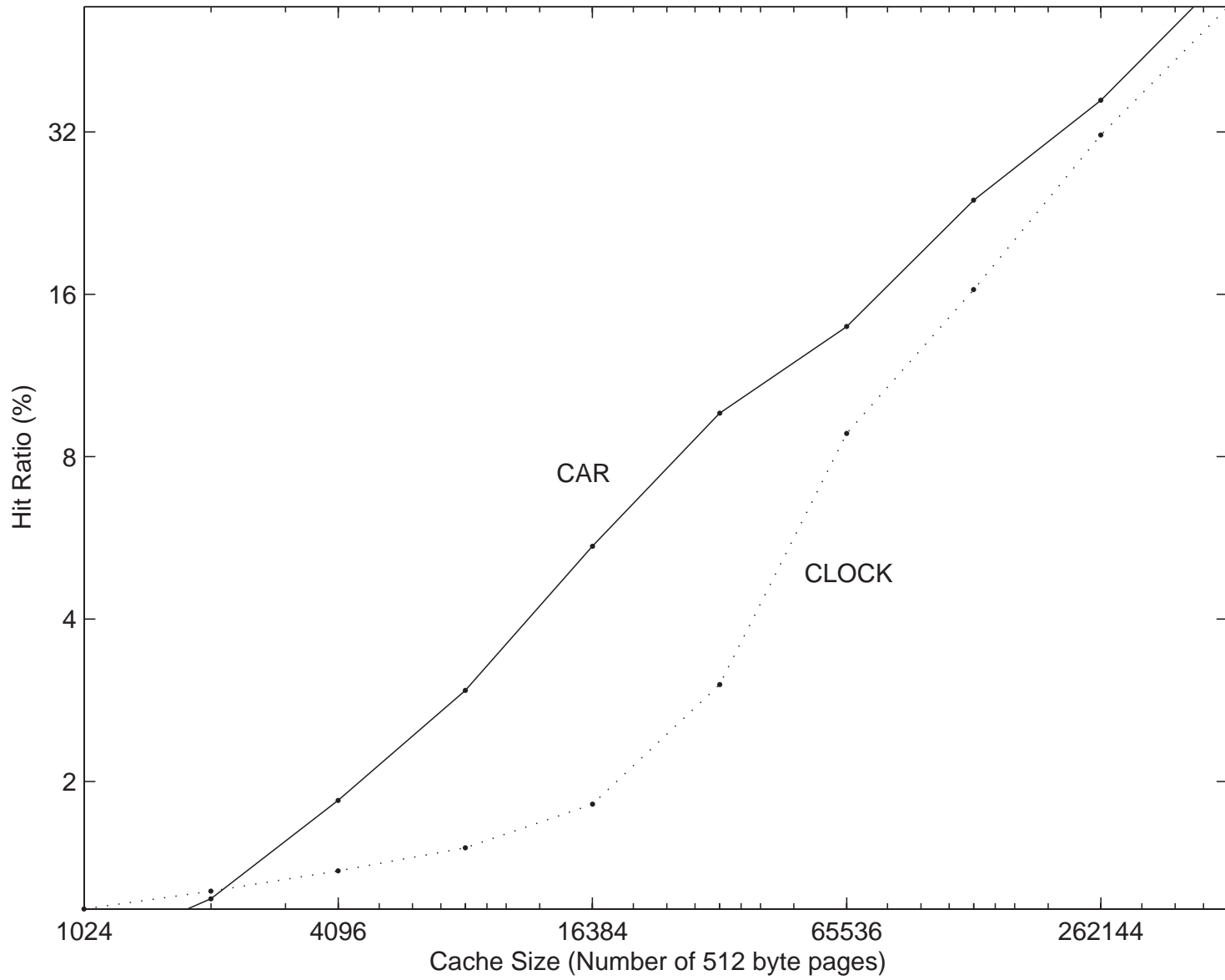




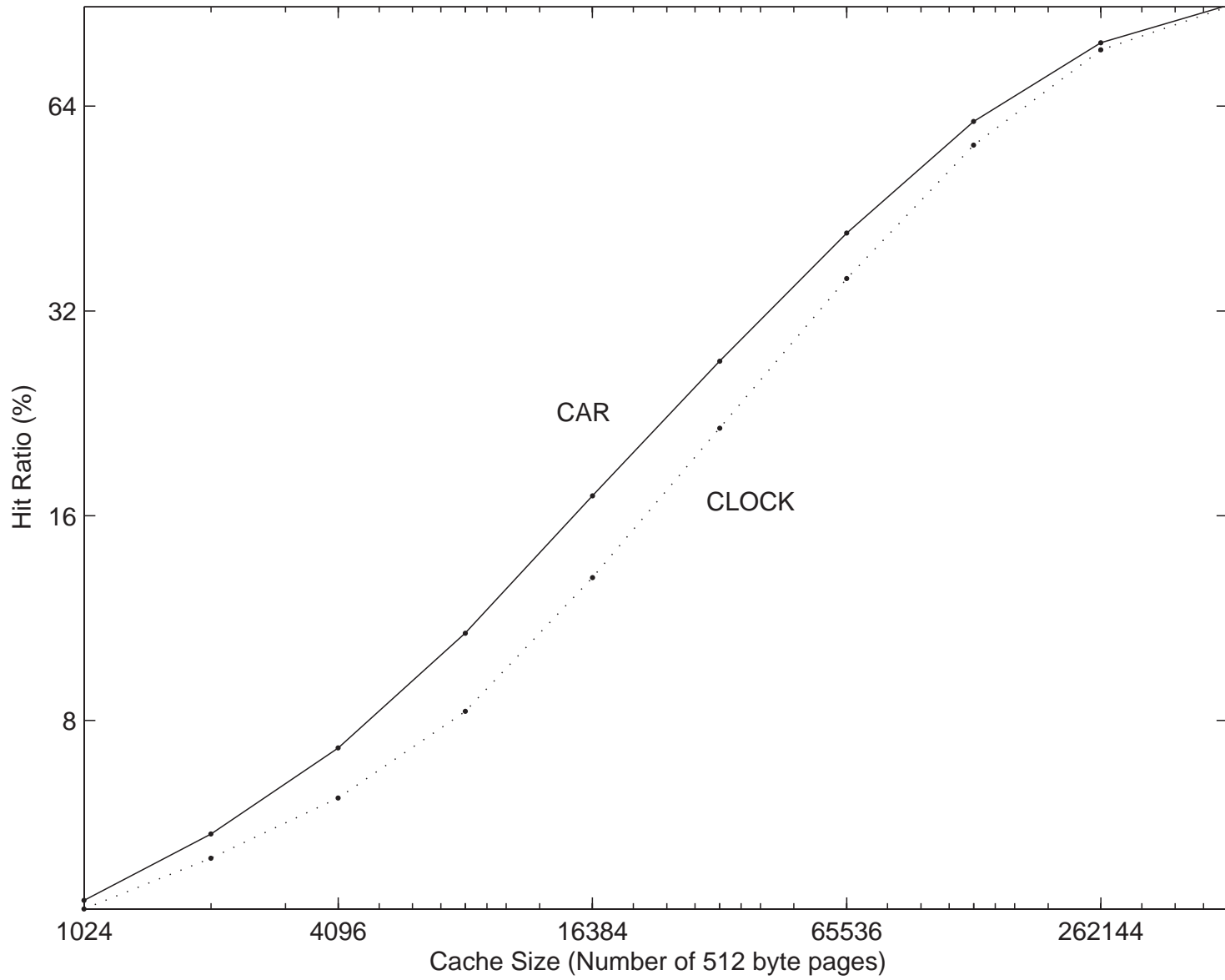
P9



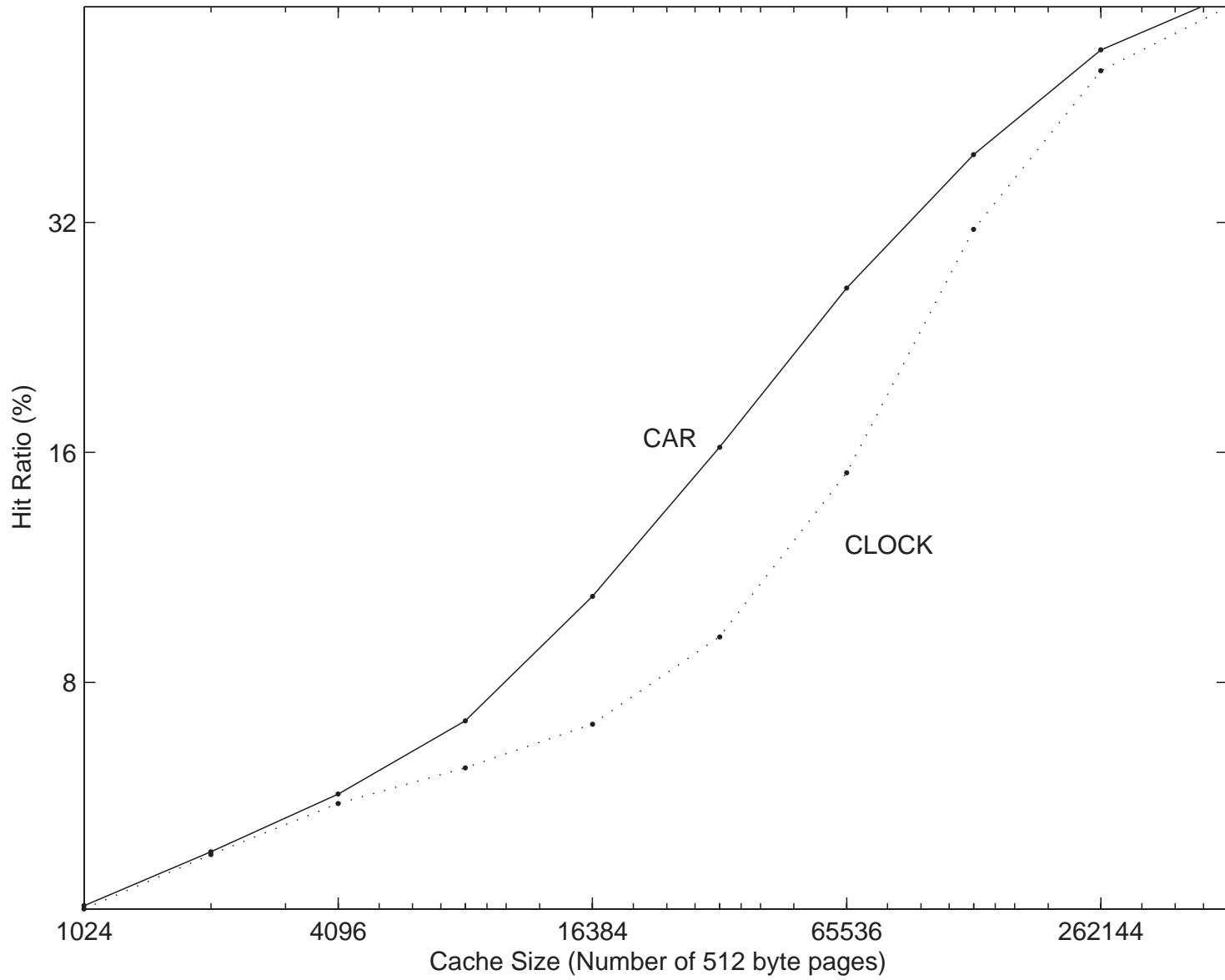
P10



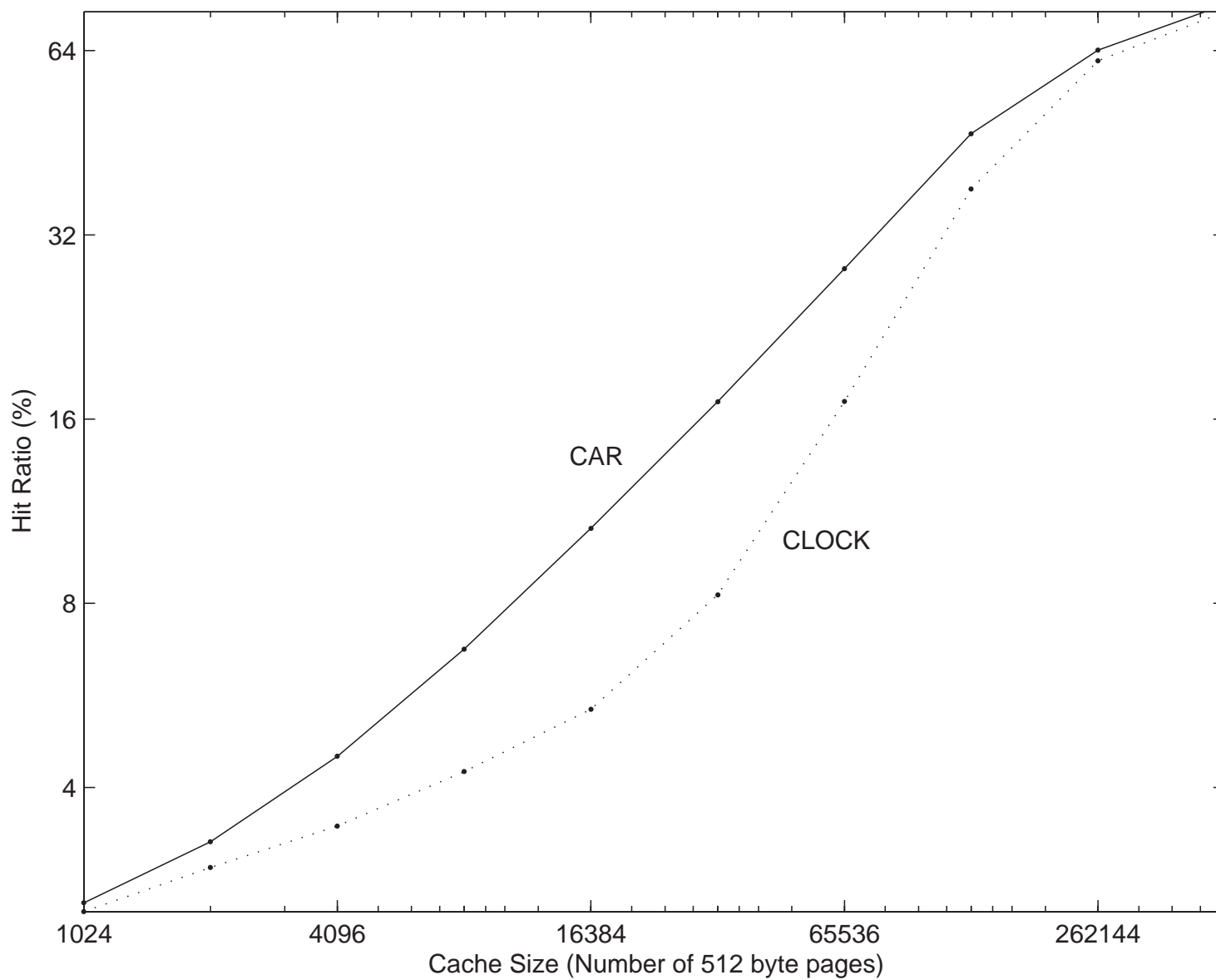
P11



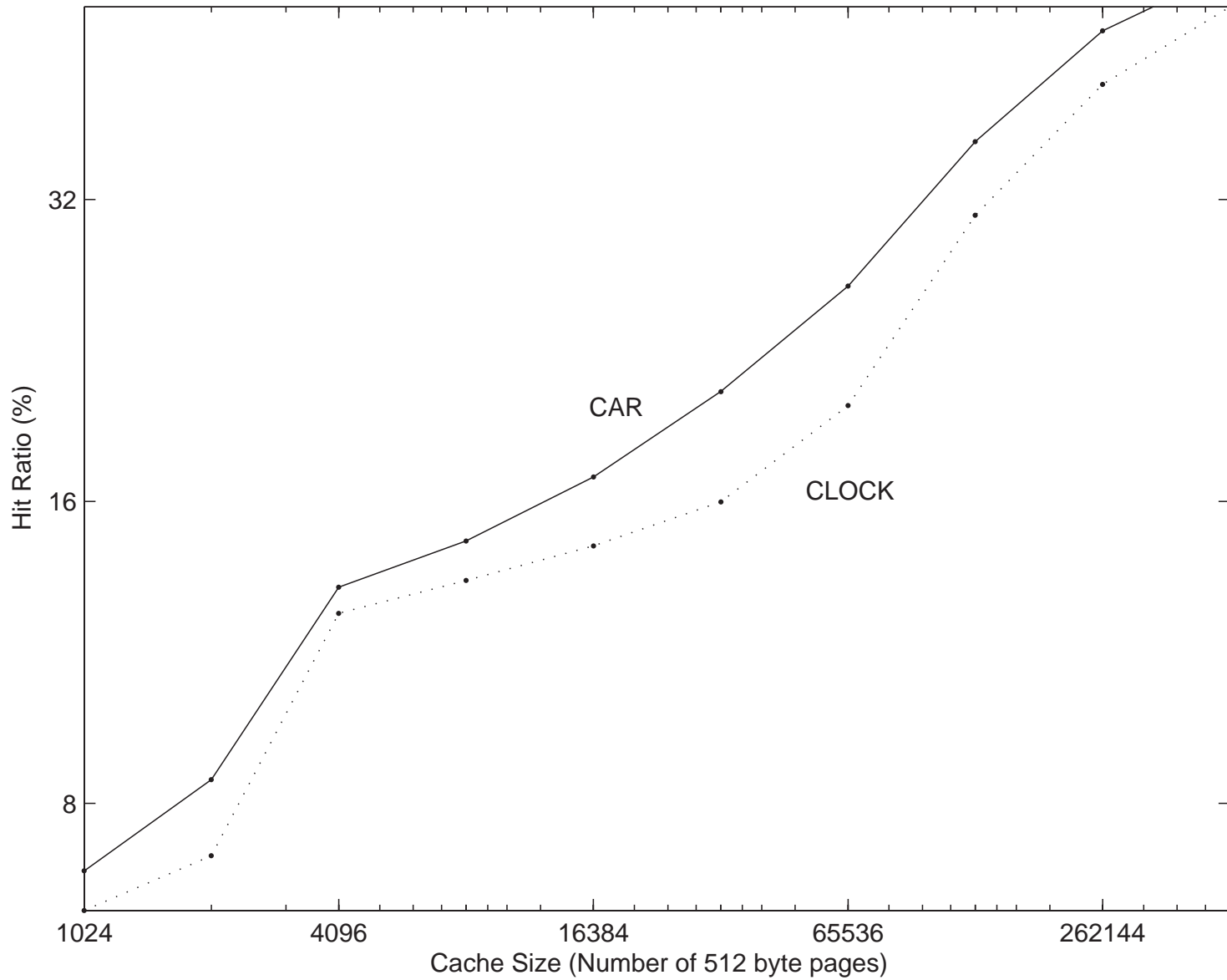
P12

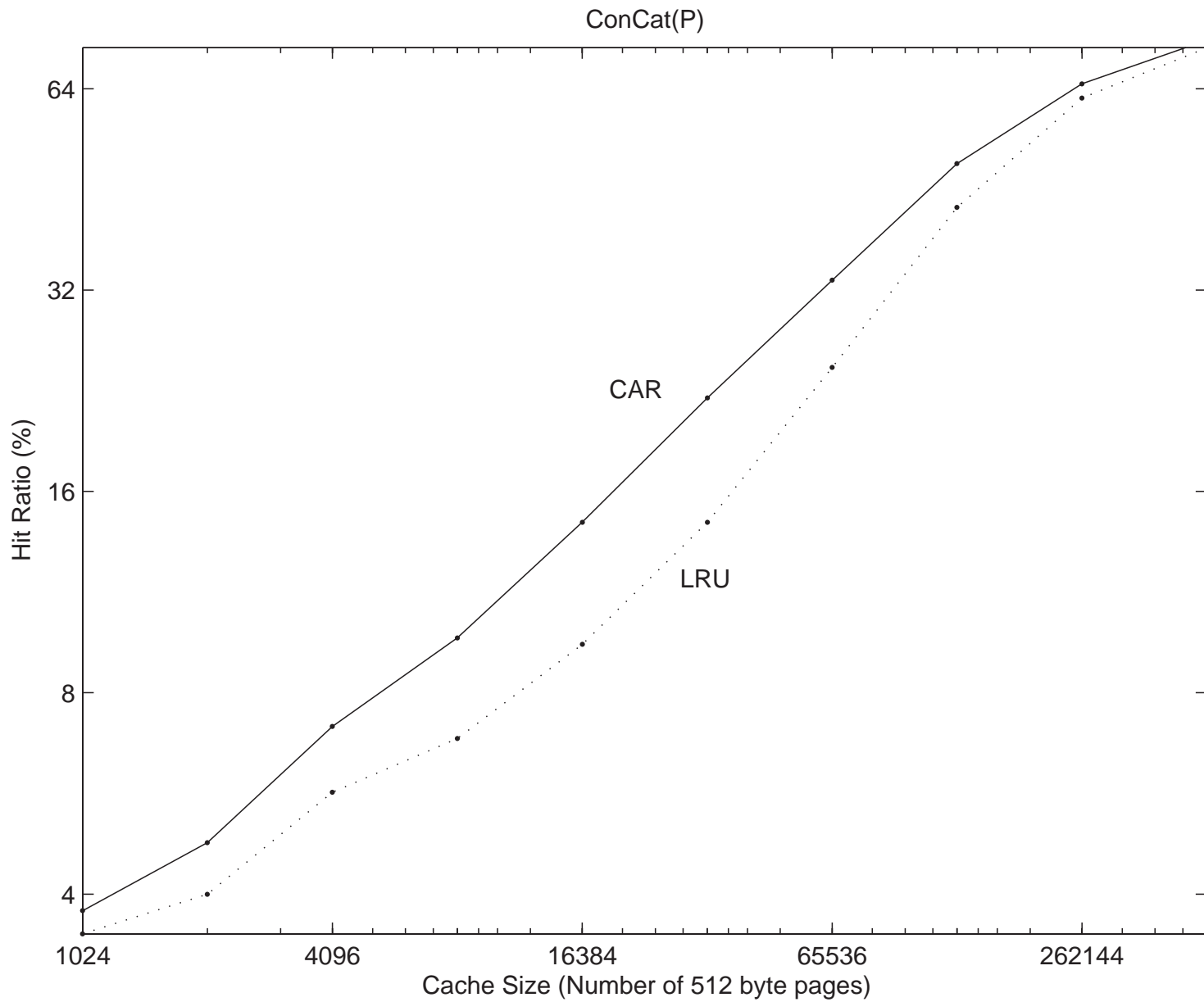


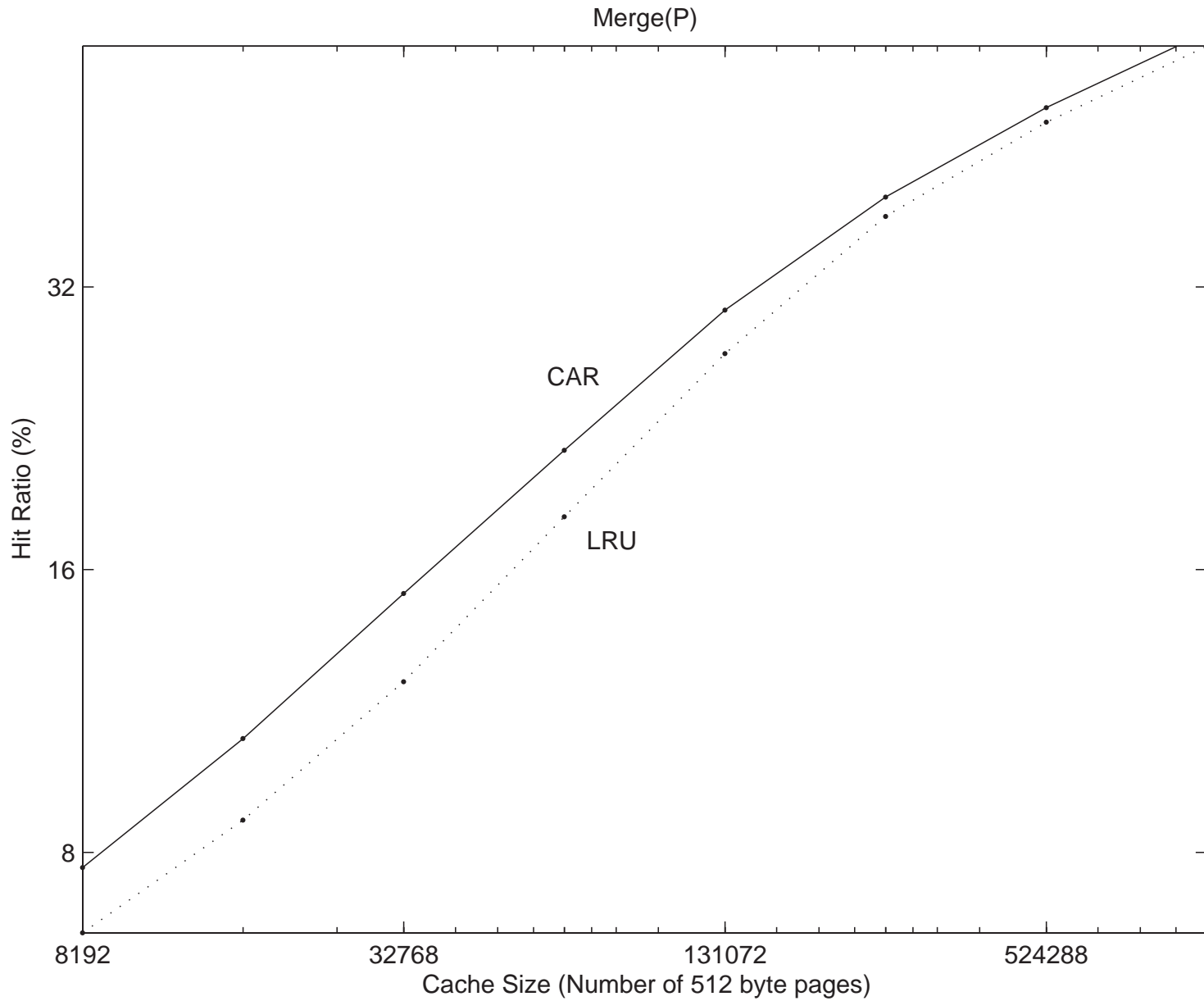
P13



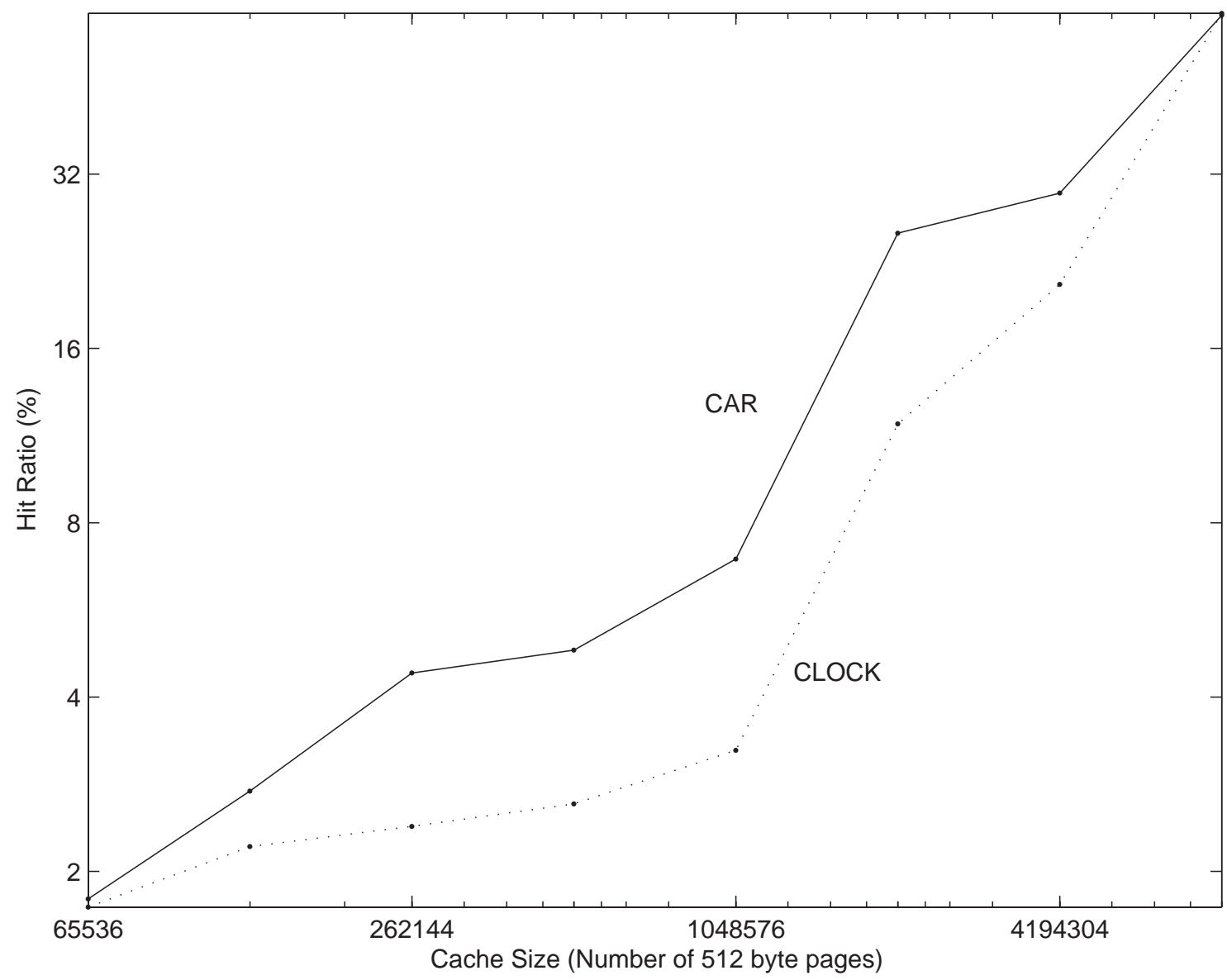
P14

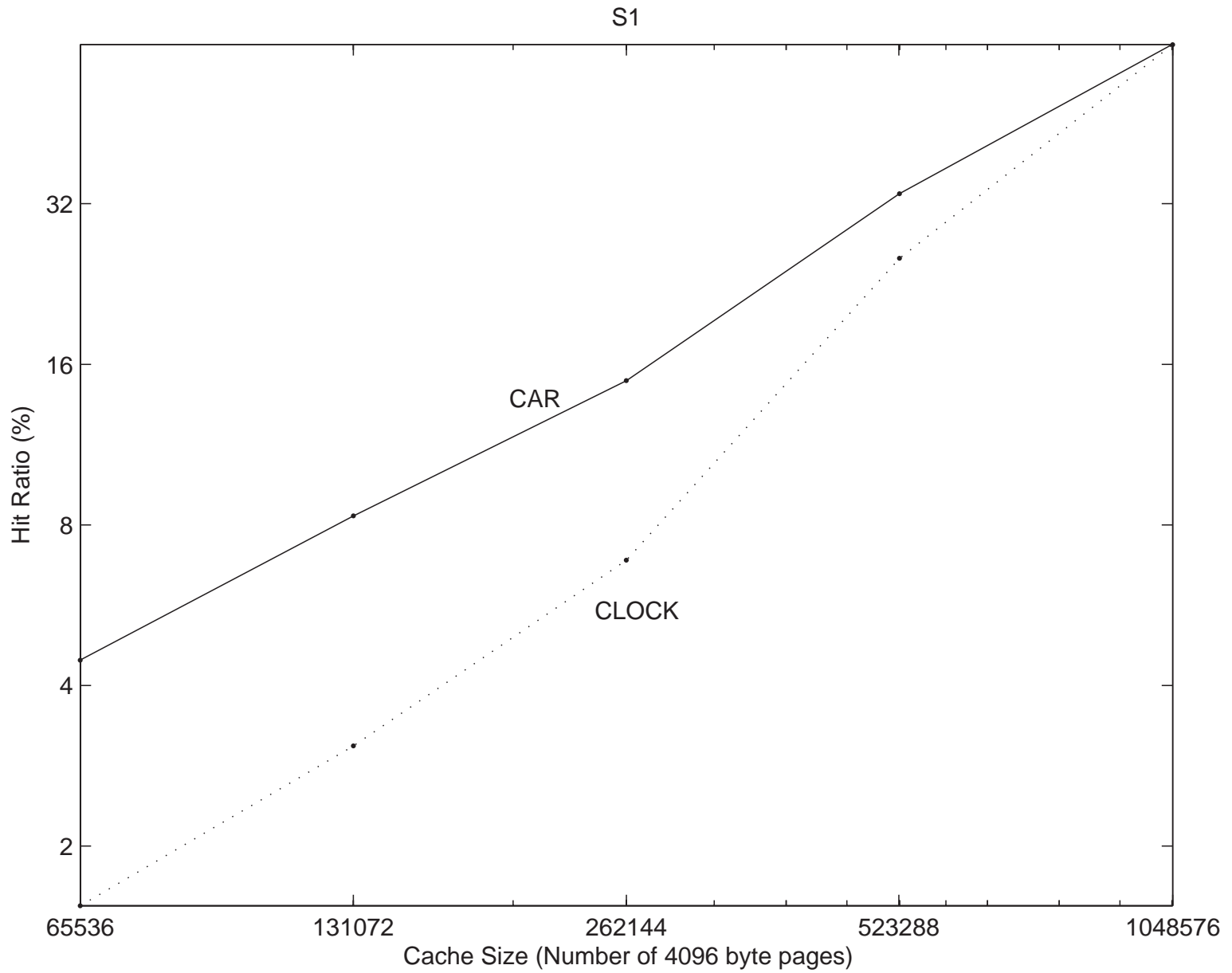


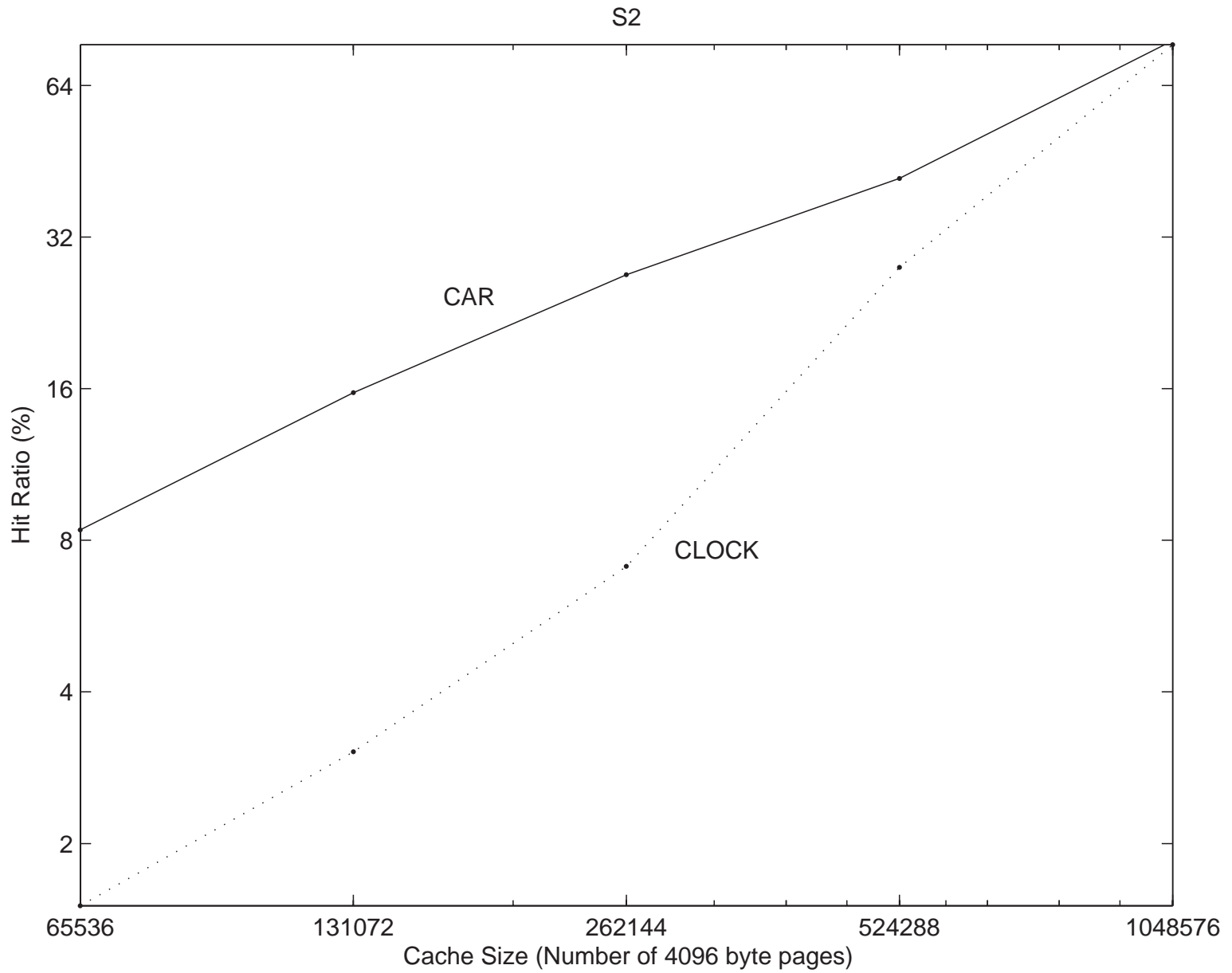


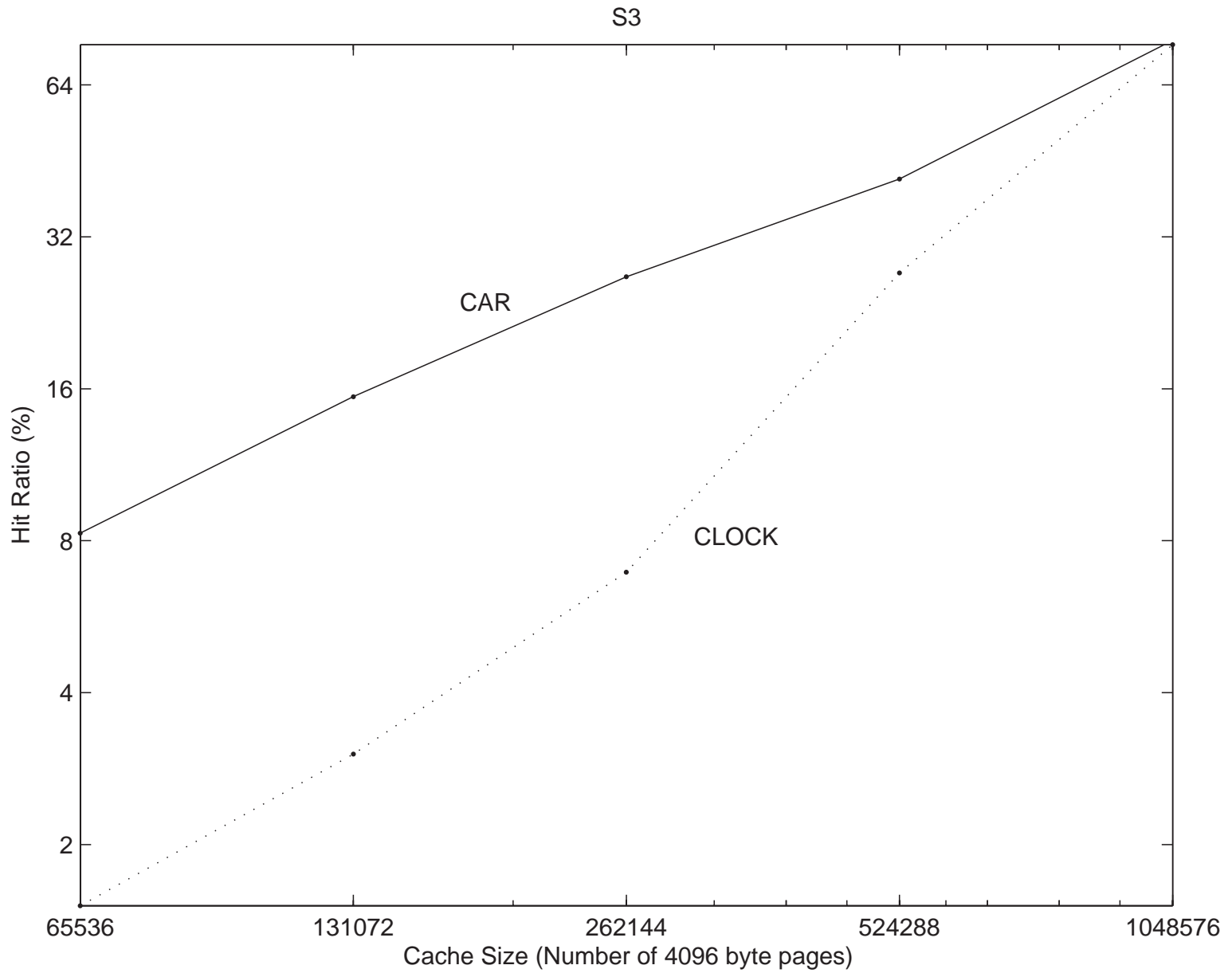


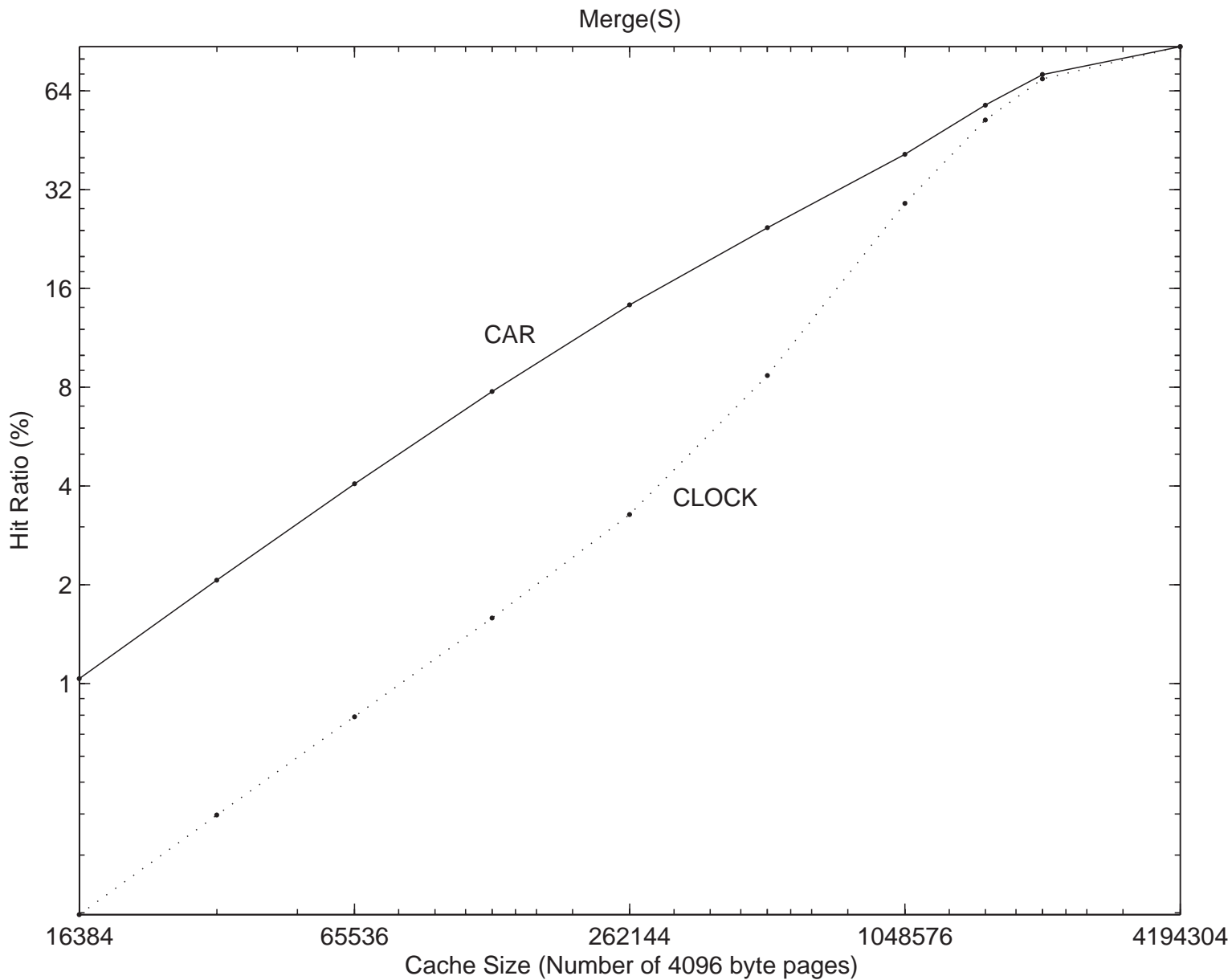
DS1

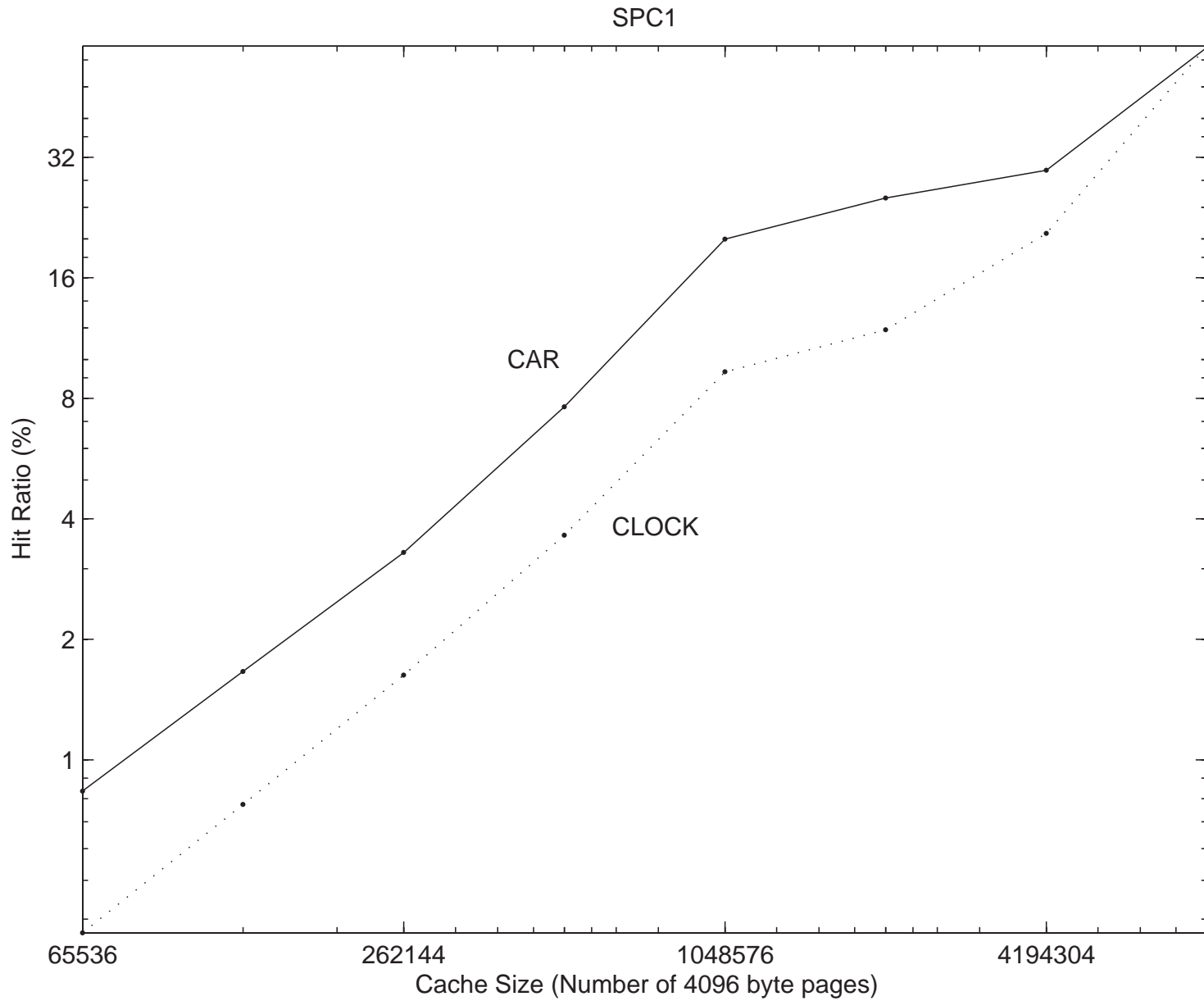








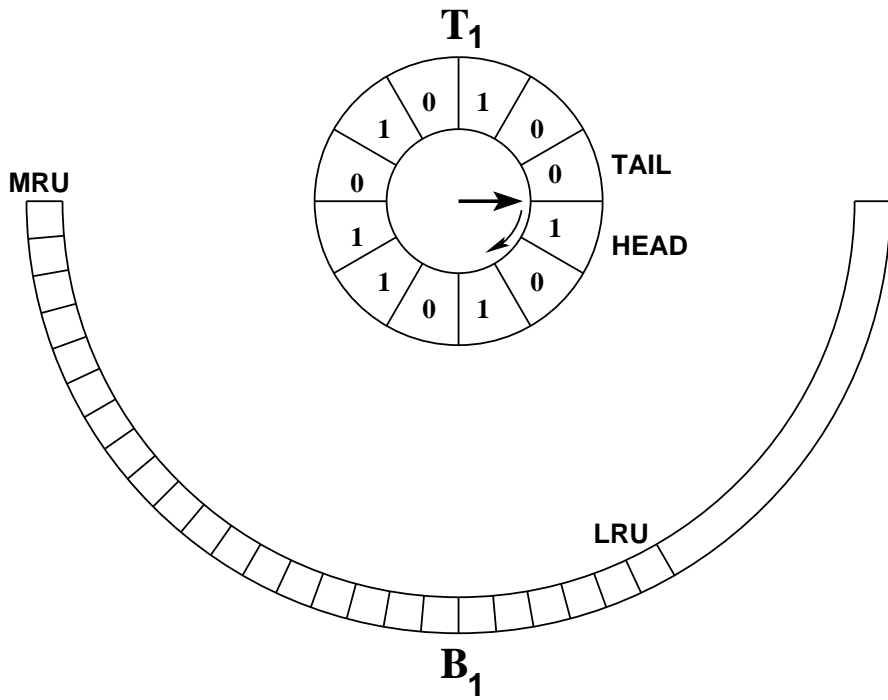




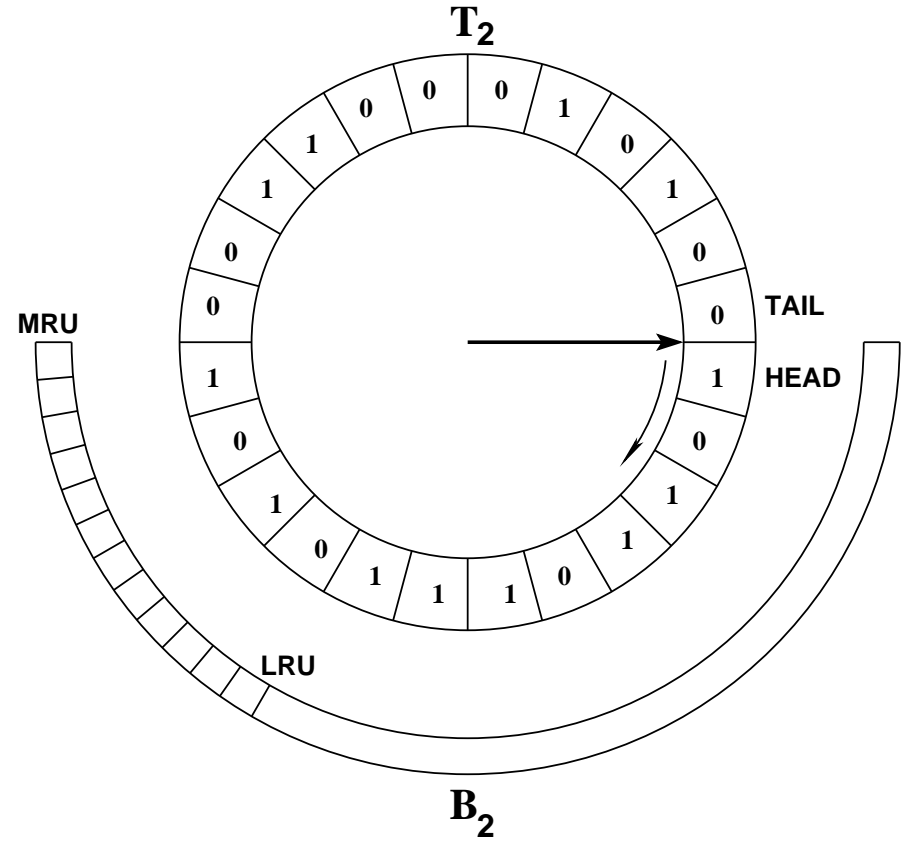
CAR: CONCLUSIONS

- **Simple and Low Overhead**
- **Self-tuning: captures “recency” and “frequency”**
- **Scan-Resistant**
- **Low Lock Contention and MRU Overhead**
- **Outperforms CLOCK on all workloads examined**
- **Comparable to (sometimes even better than) ARC!**

"Recency"



"Frequency"



CAR: Set-up

- **Clocks T1 and T2 contain cache pages**
- **B1 and B2 contain recently evicted history pages**
 - Alex Haley: “History is written by the winners”
 - CAR/ARC: “History is written by the losers”
- **Size of T1 roughly equals B2**
- **Size of T2 roughly equals B1**
- **T1(0) and B1 contain pages that have been seen exactly once recently = “Recency”**
- **T1(1), T2, and B2 contain pages that have been seen at least twice recently = “Frequency”**

CAR: Algorithm

- **HIT in T1 or T2: Set reference bit to 1**
- **MISS in B1: Set reference bit to 0, move to tail of T2, and increase target size of T1**
- **MISS in B2: Set reference bit to 0, move to tail of T2, and decrease target size of T1**
- **TOTAL MISS: Set reference bit to 0, move to tail of T1**
- **CACHE REPLACEMENT POLICY: Replace from T1 if larger than target; else from T2**
 - During Replacement in T1, if “1” page is found, make “0” and move to T2 tail, move evicted page to B1
 - During Replacement in T2, if “1” page is found, make “0”, move evicted page to B2
- **DIRECTORY REPLACEMENT POLICY: Replace from B1 if $T1+B1 = c$; else from B2**

CLOCK

- **HIT**: Set the reference bit to “1”
- **MISS**: Insert at the TAIL, initialize the reference bit to “0”
- **REPLACEMENT POLICY**:
 - Evict the first “0” page
 - Reset “1” pages to “0”—giving them a “second chance”
- **KEY INSIGHT**:
 - Reseting “1” to “0” is “delayed MRU”—removing it from hit path to miss path

