ARC : A Self-Tuning, Low Overhead Replacement Cache

Nimrod Megiddo, Dharmendra S. Modha

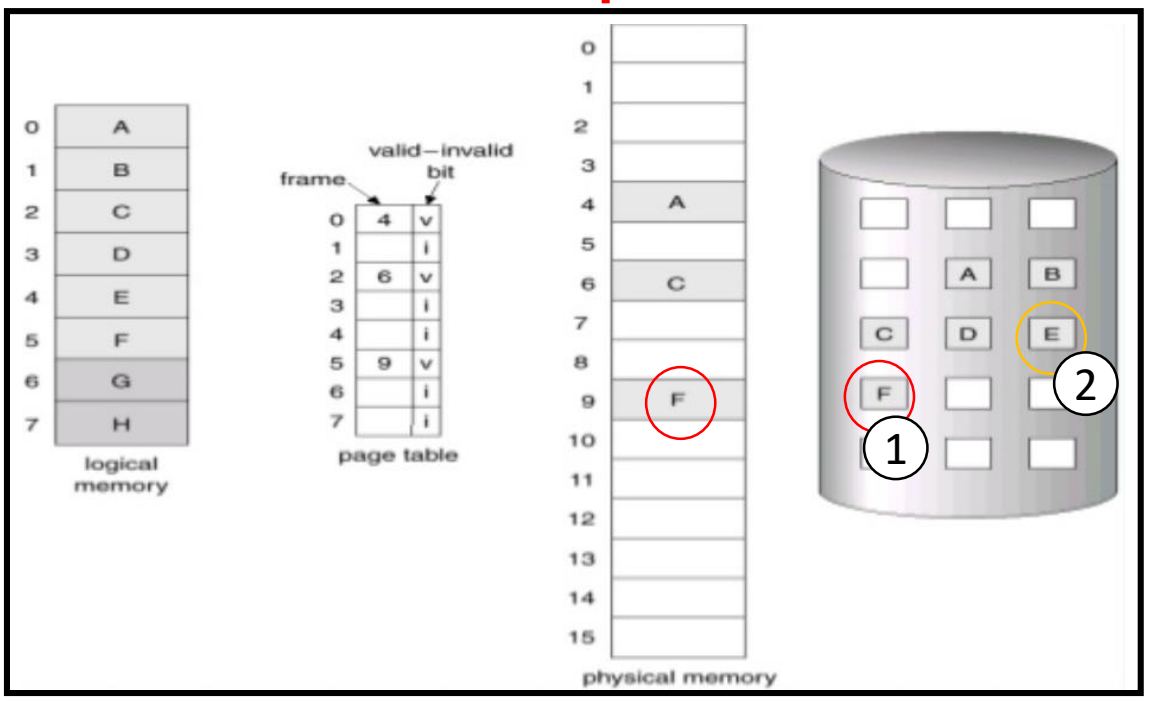
밑빠진독에코인붓기

Introduction

Before Cache Mangement Problem

Demand Paging vs Prefetching Page

Input Sequence : A C F B D E ……



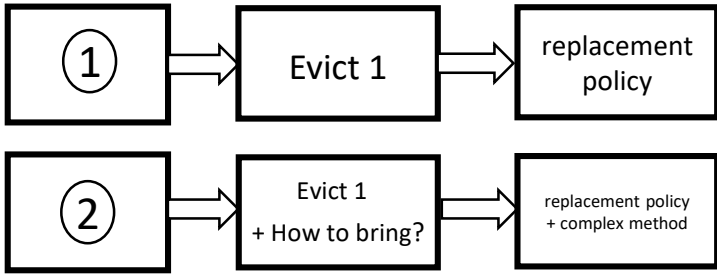
① **Demand Paging**

- Only Allocate when it requests

② **Prefetching**

- Predict Future for reasonable heuristics ex) bring near pages

Fault Occurs?



Problem Description

Cache Management Problem on Demand Paging Model

- **Main and the only Concern : Hit ratio**

Cache Replacement Algorithm

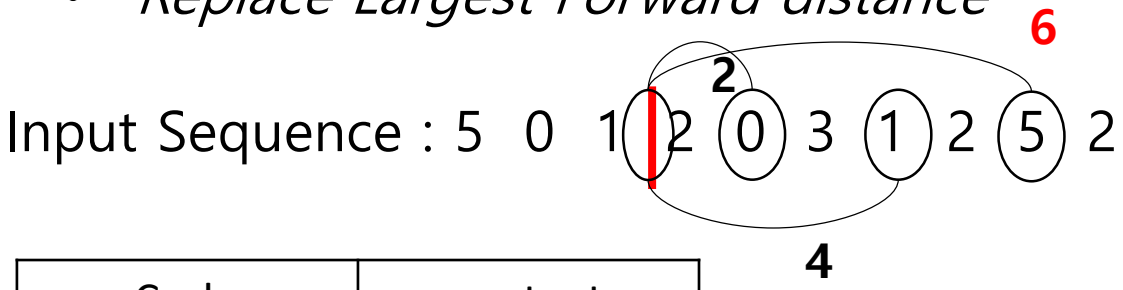
Goal : Better than *LRU*

- **Maximize hit rate**
- **Minimize Overhead**
  - **Time**
  - **Space**

Previous Work – Offline Optimal

Belady's *MIN*

- *Replace Largest Forward distance*



Cache	content
0	5
1	0
2	1

Evict!

But, Who knows future?

Before Previous Work

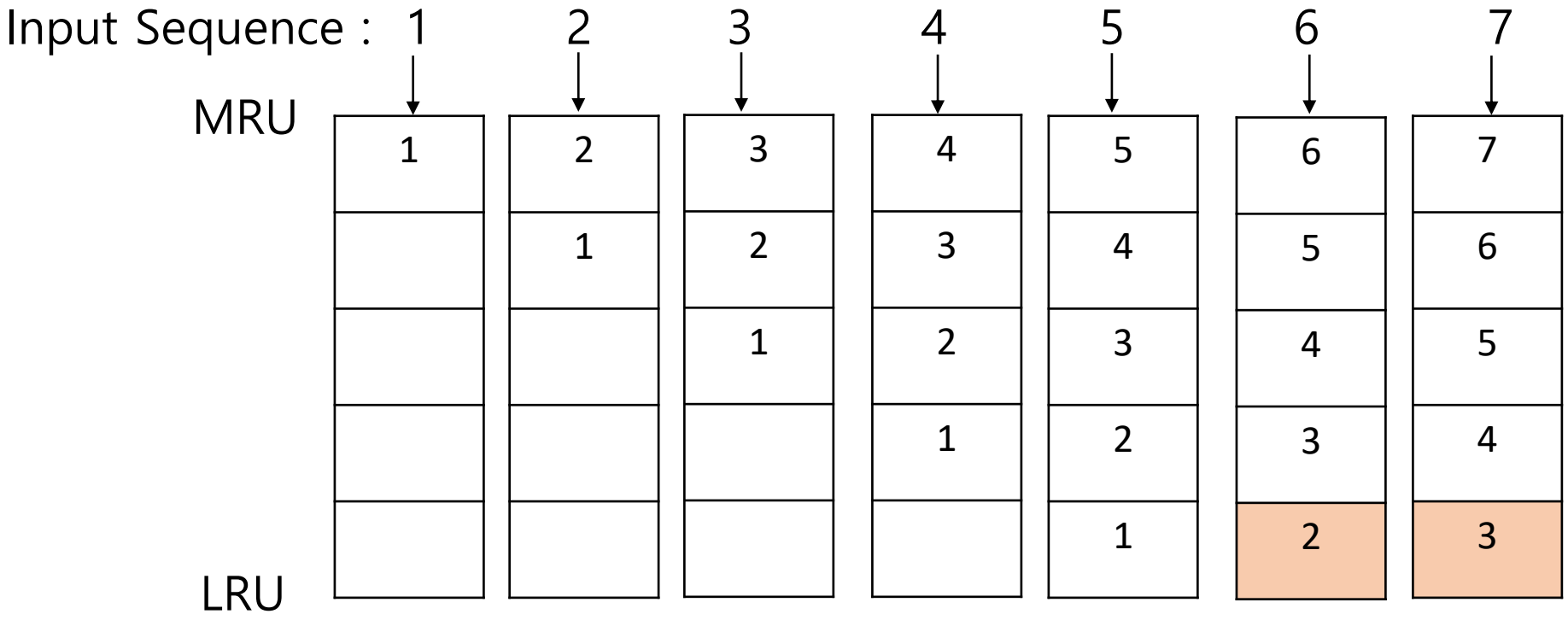
OLTP

c	LRU	ARC	FBR	LFU	LIRS	MQ	LRU-2	2Q	LRFU	MIN
	ONLINE						OFFLINE			
1000	32.83	38.93	36.96	27.98	34.80	37.86	39.30	40.48	40.52	53.61
2000	42.47	46.08	43.98	35.21	42.51	44.10	45.82	46.53	46.11	60.40
5000	53.65	55.25	53.53	44.76	47.14	54.39	54.78	55.70	56.73	68.27
10000	60.70	61.87	62.32	52.15	60.35	61.08	62.42	62.58	63.54	73.02
15000	64.63	65.40	65.66	56.22	63.99	64.81	65.22	65.82	67.06	75.13

Metric	*ARC*
Recency	O
Frequency	O
Time Complexity	O(1)
Scan Resistance	O
“Self-tunable”	O

Previous Work - Recency

LRU(Least Recently Used)



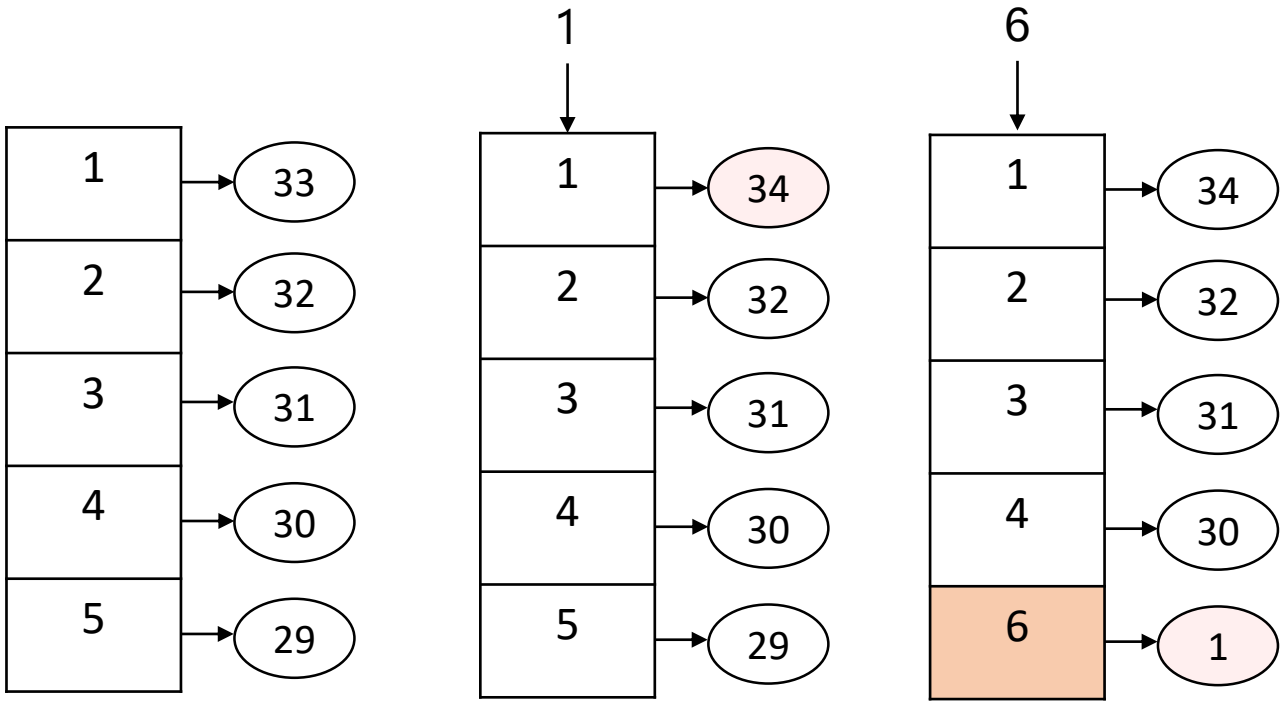
Data structure : Queue, Time complexity : O(1)

Old-Fashion, But Quite Good Performance

No Frequency Concerned

Previous Work - Frequency

LFU(Least Frequently Used)



Data structure : Priority Queue, Time complexity : O(logn)

No Recency Concerned

Previous Work - Frequency

LRU-2(LFU Extension)

Inter-arrival rate ↓ High-frequent !

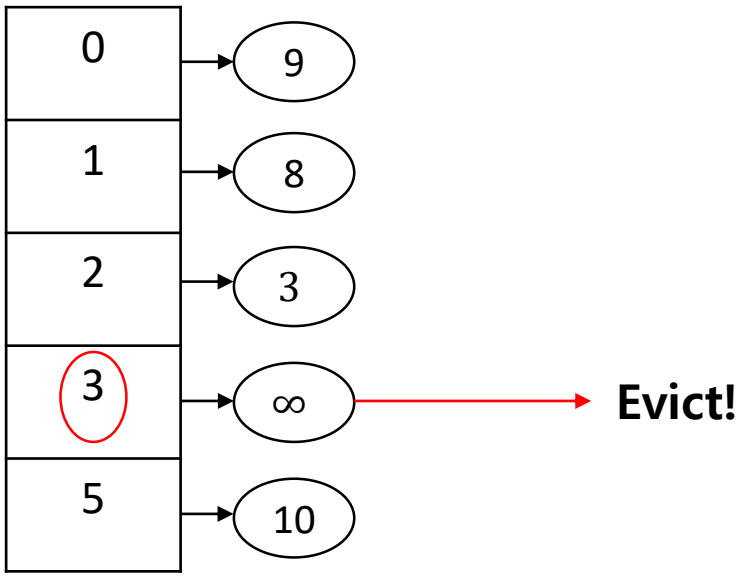
How to define 'Frequent'?

Inter-arrival rate

Input Sequence : 5 0 1 2 0 3 1 2 5 2 | 7

Time tick : 1 2 3 4 5 6 7 8 9 10

2nd Backward Distance



Problem
CIP(Correlated Information Period)

$x_0, \dots, x_i, x_{i+1}, \dots, x_j$

How far do we have to consider 2 pages are related?

Previous Work - Frequency

LRU-2(LFU Extension)

c	CIP/c 0.01	0.05	0.1	0.25	0.5	0.75	0.9	0.95	0.99
1024	0.87	1.01	1.41	3.03	3.77	4.00	4.07	4.07	4.07
2048	1.56	2.08	3.33	4.32	4.62	4.77	4.80	4.83	4.83
4096	2.94	4.45	5.16	5.64	5.81	5.79	5.70	5.65	5.61
8192	5.36	7.02	7.39	7.54	7.36	6.88	6.47	6.36	6.23
16384	9.53	10.55	10.67	10.47	9.47	8.38	7.60	7.28	7.15
32768	15.95	16.36	16.23	15.32	13.46	11.45	9.92	9.50	9.03
65536	25.79	25.66	25.12	23.64	21.33	18.26	15.82	14.99	14.58
131072	39.58	38.88	38.31	37.46	35.15	32.21	30.39	29.79	29.36
262144	53.43	53.04	52.99	52.09	51.73	49.42	48.73	49.20	49.11
524288	63.15	63.14	62.94	62.98	62.00	60.75	60.40	60.57	60.82

Hard to Predict best CIP Value only on 1 workload !

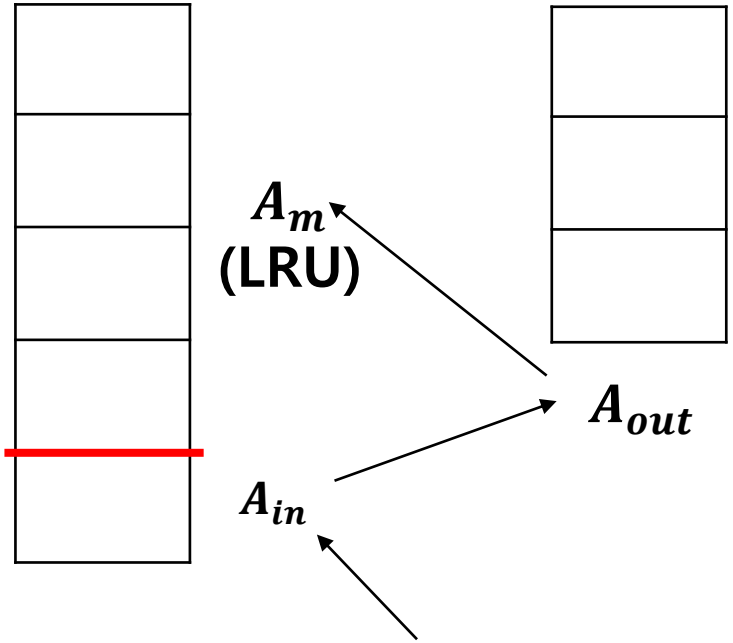
Metric	LRU-2
Time Complexity	O(logn)
Self-tunable	X

Previous Work - Frequency

2Q(LRU-2 Extension)

Input Sequence : 5 0 1 2 0 3 1 2 5 2 | 7

$K_{in} = 0.2c$
$K_{out} = 0.6c$



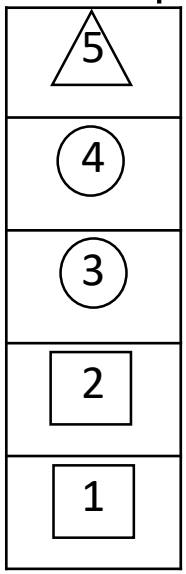
Still Have to tune some parameters

Metric	2Q
Time Complexity	O(1)
Self-tunable	X

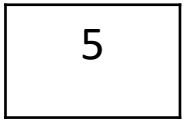
Previous Work - Frequency

LIRS(Low Inter-reference Recency Set)

Input Sequence : 1 4 2 3 2 1 4 1 5



LIRS stack



HIRS stack

$$c = 3$$
$$L_{hirs} = 1$$
$$L_{lirs} = 2$$

$$L_{lirs} + L_{hirs} = c$$
$$L_{lirs} = 0.001c$$

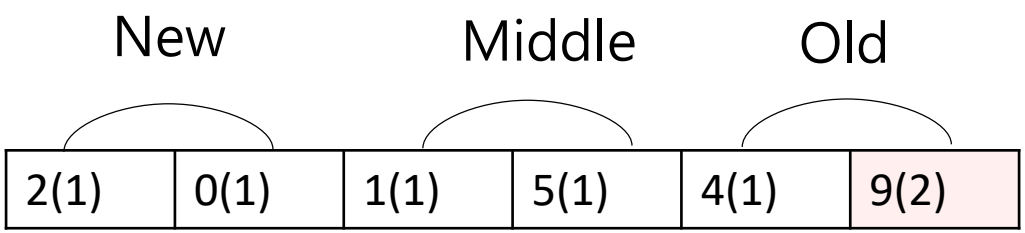
Still Have to tune some parameters

Metric	LIRS
Time Complexity	Stack-Pruning
Self-tunable	X

Previous Work - Frequency + Recency

FBR(Frequency Based replacement)

Input Sequence : 0 5 11



$\boldsymbol{Partition\ ratio\ of\ 3\ Sections}$

A_{max}(predetermined)

C_{max}(predetermined)

Still Have to tune some parameters

Metric	FBR
Self-tunable	X

Previous Work - Frequency + Recency

LRFU(Least Recently Frequently Used)

$$F(x) = \left(\frac{1}{p}\right)^{\lambda x}$$

Input Sequence : (1) 4 2 3 2 (1) 4 (1) 5

Time Tick : 1 2 3 4 5 6 7 8 9 10

$$C_{10}(1) = \text{F(10-1)} + \text{F(10-6)} + \text{F(10-8)}$$

$$if\ \lambda \to 0, \quad F(x) = 1 \longrightarrow LFU$$

$$if\ \lambda \to 1, \quad F(x) = \left(\frac{1}{p}\right)^{x} \longrightarrow LRU$$

$$\forall i \;\; \mathcal{F}(i) > \sum_{j=i+1}^{\infty} \mathcal{F}(j).$$

$$1 > \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots$$

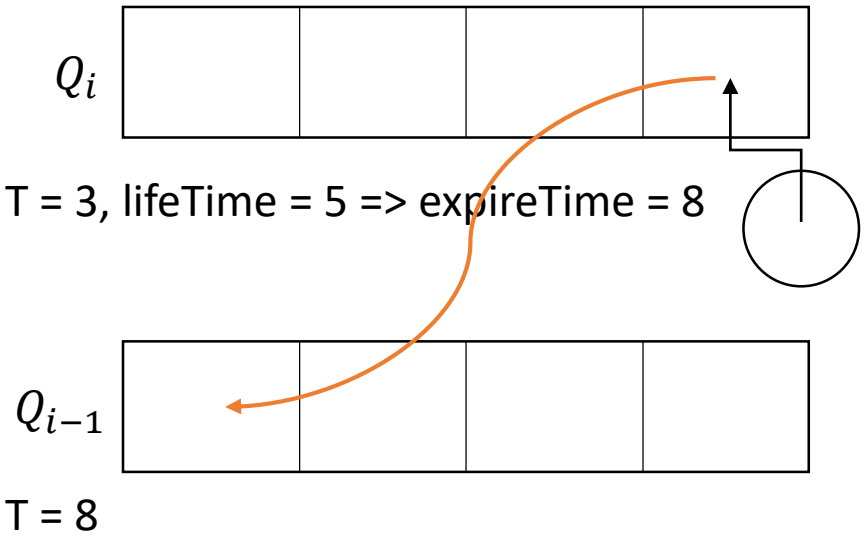
λ, p

Still Have to tune some parameters

Metric	LRFU
Self-tunable	X

Previous work - MQ

- Multi-Queue replacement policy
- Each Q_i contains $[2^i, 2^{i+1})$ times seen pages
- Each hit page is assigned lifetime
- If a page expired, it goes down to lower-level queue's MRU position



- Parameter tuning needed
  - # of queues, lifeTime, …
- Overhead for checking timestamp

Terms and basics

$\prod(c)$: Replacement Policies which manage *c* pages

L_1 : Set of pages seen only once recently

L_2 : Set of pages seen more than once recently

$$0 \leq L_1 + L_2 \leq 2c, \quad 0 \leq L_1 \leq c, \quad 0 \leq L_2 \leq 2c$$

Replace L_1 when L_1 is full, if not, replace L_2
-> make $|L_1|$ and $|L_2|$ similar

DBL(2c)

DBL$(2c)$

INPUT: The request stream $x_1, x_2, \ldots, x_t, \ldots$.
INITIALIZATION: Set $\ell_1 = 0$, $\ell_2 = 0$, $L_1 = \emptyset$ and $L_2 = \emptyset$.

For every $t \geq 1$ and any x_t, one and only one of the following two cases must occur.

Case I: x_t is in L_1 or L_2.
A cache hit has occurred. Make x_t the MRU page in L_2.

Case II: x_t is neither in L_1 nor in L_2.
A cache miss has occurred. Now, one and only one of the two cases must occur.

Case A: L_1 has exactly c pages.
Delete the LRU page in L_1 to make room for the new page, and make x_t the MRU page in L_1.

Case B: L_1 has less than c pages.
1) If the cache is full, that is, $(|L_1| + |L_2|) = 2c$, then delete the LRU page in L_2 to make room for the new page.
2) Insert x_t as the MRU page in L_1.

Cache hit : MRU update

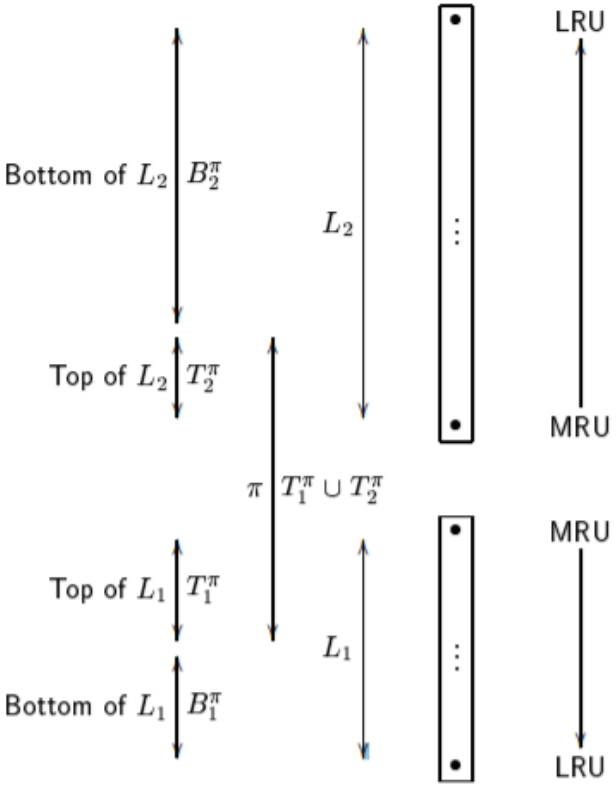
Cache miss :
- Case A : L_1 is full -> replace LRU of L_1
- Case B : L_1 isn't full
  - Case B-1 : L_1 + L_2 = $2c$ (cache full)
    -> replace LRU of L_2
  - Case B-2 : L_1 + L_2 < $2c$ (cache isn't full)
    -> insert into L_1

Summary : Fill L_1 up to c

π(c) – general structure

T_1^π : Top(MRU) of L_1 $\quad$ T_2^π : Top(MRU) of L_2

B_1^π : Bottom(LRU) of L_1 $\quad$ B_2^π : Bottom(LRU) of L_2



- $T_i^\pi \cap B_i^\pi = \emptyset,\ L_i = T_i^\pi \cup B_i^\pi$
- $|L_1 \cup L_2| < c \quad => \quad B_1^\pi \cup B_2^\pi = \emptyset$
- $|L_1 \cup L_2| \geq c \quad => \quad |T_1^\pi \cup T_2^\pi| = c$
- MRU(B) < LRU(T)
- Cache contains $T_1^\pi \cup T_2^\pi$

LRU(c)

- In DBL(2c)
  - If a page in L_1 evicted
    - L_1 must be full ($|L_1| = c$)
  - If a page in L_2 evicted
    - $|L_2| \geq c$

=> Evicted page is at most c MRU

Both DBL(2c) and LRU(c) properly keep c MRU (proved)

$FRC_P(c)$ (Fixed Replacement Cache)

- p is tunable parameter $(0 \leq p \leq c)$
- Follows all properties of π(c)
- Keeps $|T_1| = p,\ |T_2| = c - p$
- Replacement Policy
  - If $|T_1| > p$: Replace LRU of T_1
  - If $|T_1| < p$: Replace LRU of T_2
  - If $|T_1| = p$: (arbitrary)

ARC(c) (Adaptive Replacement Cache)

- Parameter p adaptively changes (No tuning needed)
- Cache miss and the page was in B_1 => increase $|T_1|$ (increase p)
- Cache miss and the page was in B_2 => increase $|T_2|$ (decrease p)
- The amount of change in parameter p is **learning rate**
  - If B_1 contains missed page => we should increase p
    - If $|B_1| \geq |B_2|$: increase p by 1
    - If $|B_1| < |B_2|$: increase p by $\frac{|B_2|}{|B_1|}$
  - Similar in B_2 hit case.

ARC(c) - algorithm

(x : Current requested stream)

- Case 1 : $x \in (T_1 \cup T_2)$
  - Cache hit. Move x into T_2 and update MRU
- Case 2 : $x \in B_1$
  - Increase p by **learning rate**
  - Move x into T_2 and update MRU
- Case 3 : $x \in B_2$
  - Decrease p by **learning rate**
  - Move x into T_2 and update MRU

ARC(c) - algorithm

- Case 4 : $x \notin (B_1 \cup B_2)$
  - Case 4-A : $|T_1 \cup B_1| = c$
    - Case 4-A-1 : $|T_1| < c$
      - Replace LRU of B_1
    - Case 4-A-2 : $|T_1| = c$
      - B_1 is empty -> Replace LRU of T_1
  - Case 4-B : $|T_1 \cup B_1| < c$
    - If $|T_1| + |T_2| + |B_1| + |B_2| \geq c$
      - Replace LRU of B_2 if $|T_1| + |T_2| + |B_1| + |B_2| = 2c$
  - Finally : Move x into T_1 and update MRU

ARC(c,k)

- Keeps recent k pages discarded from L_1 (not from L_2)
- All pages in the keeping set are more recent than **new** L_2 pages
  - Otherwise, discard the page from keeping set
- If request hits the keeping set, move that page into T_2 directly
- Previous algorithm denoted ARC($c, 0$)

Experimental result

SPC1 like

c	LRU	MQ	2Q	ARC
	ONLINE			
65536	0.37	0.37	0.66	0.82
131072	0.78	0.77	1.31	1.62
262144	1.63	1.65	2.59	3.23
524288	3.66	3.72	6.34	7.56
1048576	9.19	14.96	17.88	20.00

Merge(S)

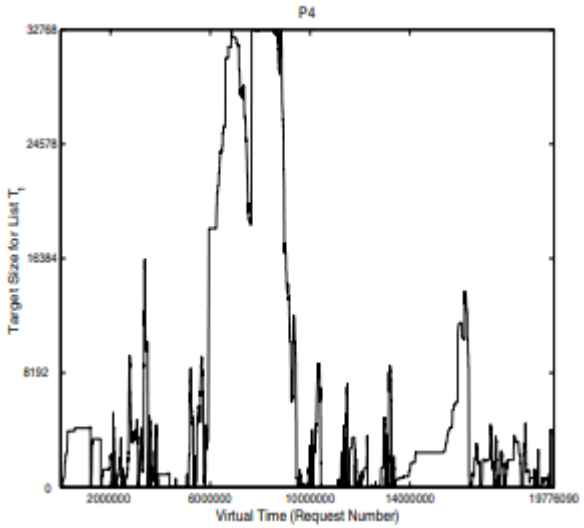
c	LRU	MQ	2Q	ARC
	ONLINE			
16384	0.20	0.20	0.73	1.04
32768	0.40	0.40	1.47	2.08
65536	0.79	0.79	2.85	4.07
131072	1.59	1.59	5.52	7.78
262144	3.23	4.04	10.36	14.30
524288	8.06	14.89	18.89	24.34
1048576	27.62	40.13	35.39	40.44
1572864	50.86	56.49	53.19	57.19
2097152	68.68	70.45	67.36	71.41
4194304	87.30	87.29	86.22	87.26

P4

c	LRU	MQ	ARC
	ONLINE		
1024	2.68	2.67	2.69
2048	2.97	2.96	2.98
4096	3.32	3.31	3.50
8192	3.65	3.65	4.17
16384	4.07	4.08	5.77
32768	5.24	5.21	11.24
65536	10.76	12.24	18.53
131072	21.43	24.54	27.42
262144	37.28	38.97	40.18
524288	48.19	49.65	53.37

Workload	c	space MB	LRU	ARC	FRC OFFLINE
P1	32768	16	16.55	28.26	29.39
P2	32768	16	18.47	27.38	27.61
P3	32768	16	3.57	17.12	17.60
P4	32768	16	5.24	11.24	9.11
P5	32768	16	6.73	14.27	14.29
P6	32768	16	4.24	23.84	22.62
P7	32768	16	3.45	13.77	14.01
P8	32768	16	17.18	27.51	28.92
P9	32768	16	8.28	19.73	20.28
P10	32768	16	2.48	9.46	9.63
P11	32768	16	20.92	26.48	26.57
P12	32768	16	8.93	15.94	15.97
P13	32768	16	7.83	16.60	16.81
P14	32768	16	15.73	20.52	20.55
ConCat	32768	16	14.38	21.67	21.63
Merge(P)	262144	128	38.05	39.91	39.40
DS1	2097152	1024	11.65	22.52	18.72
SPC1	1048576	4096	9.19	20.00	20.11
S1	524288	2048	23.71	33.43	34.00
S2	524288	2048	25.91	40.68	40.57
S3	524288	2048	25.26	40.44	40.29
Merge(S)	1048576	4096	27.62	40.44	40.18

Experimental result



Conclusion

- No parameter tuning needed
  - Empirically universal
- Less overhead
- Empirically good performance
- Robust, Scan-resistance

	LRU	LFU	LRU-2	LIRS	FBR	LRFU	MQ	ARC
Recency	O	X	X	X	O	O	O	O
Frequency	X	O	O	O	O	O	O	O
Constant Time	O	O	X	O	O	-	-	O
Scan Resistance	X	X	O	O	O	O	O	O
"Self-tunable"	O	O	X	X	X	X	X	O