Some mathematical facts about optimal cache replacement

Pierre Michaud

January 2017

This work

- Motivated by the reading of an old paper
  - Belady & Palermo, *On-line measurement of paging behavior by the multivalued MIN algorithm*, IBM Journal of R & D, 1974
  - I realized that many people (including myself) have cited Belady's 1966 paper without reading it

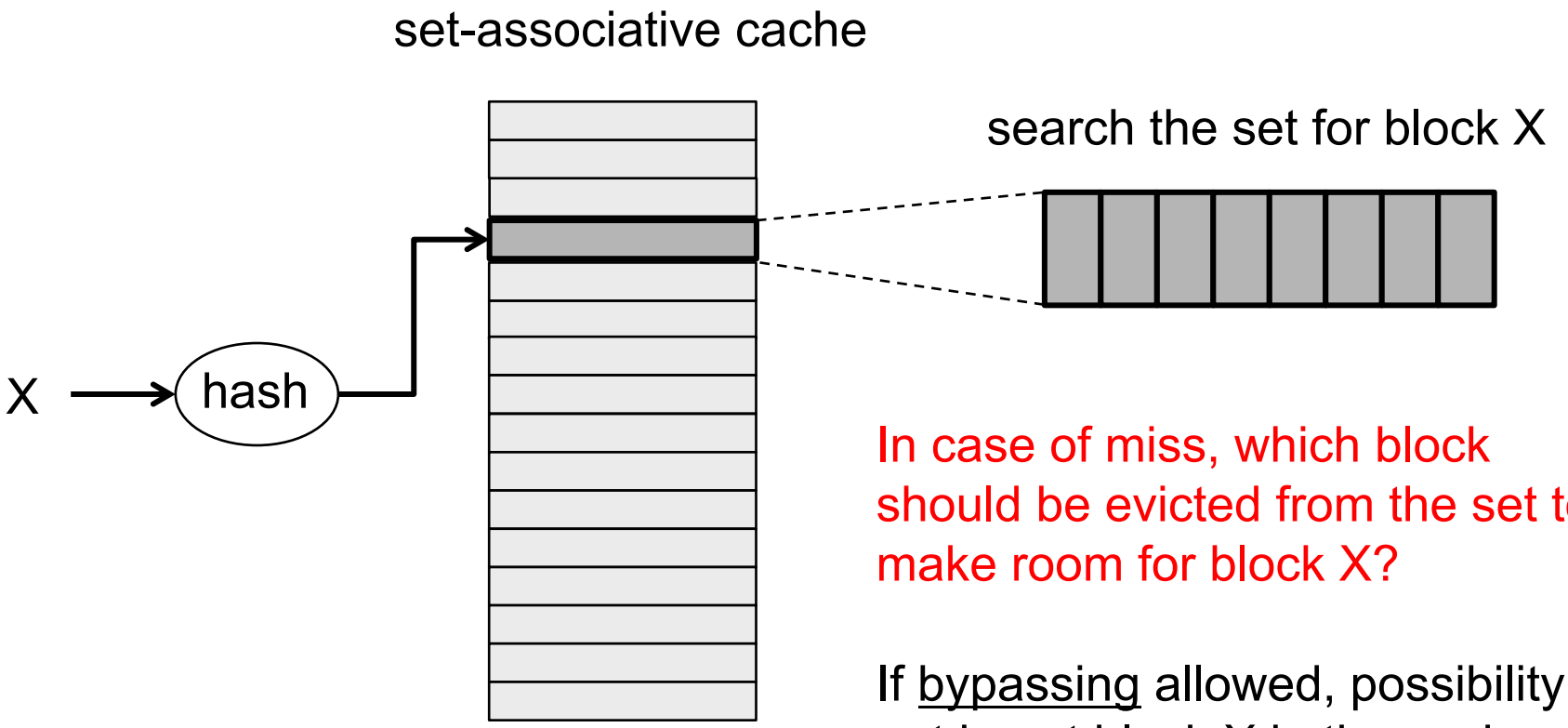
- Mathematical facts
  - theorems (and proofs)
  - independent of empirical properties of applications

This presentation

1. Brief tutorial

2. New findings

The cache replacement problem



In case of miss, which block should be evicted from the set to make room for block X?

If bypassing allowed, possibility to not insert block X in the cache

I assume fully-associative cache
(results apply to set-associative if focus on one set)

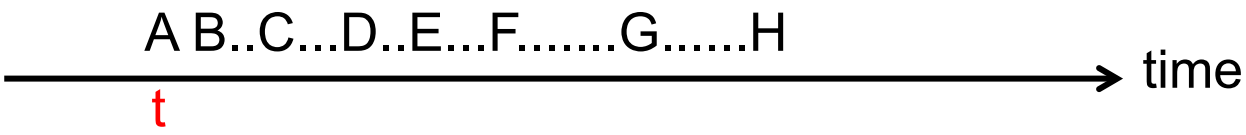
Optimal replacement

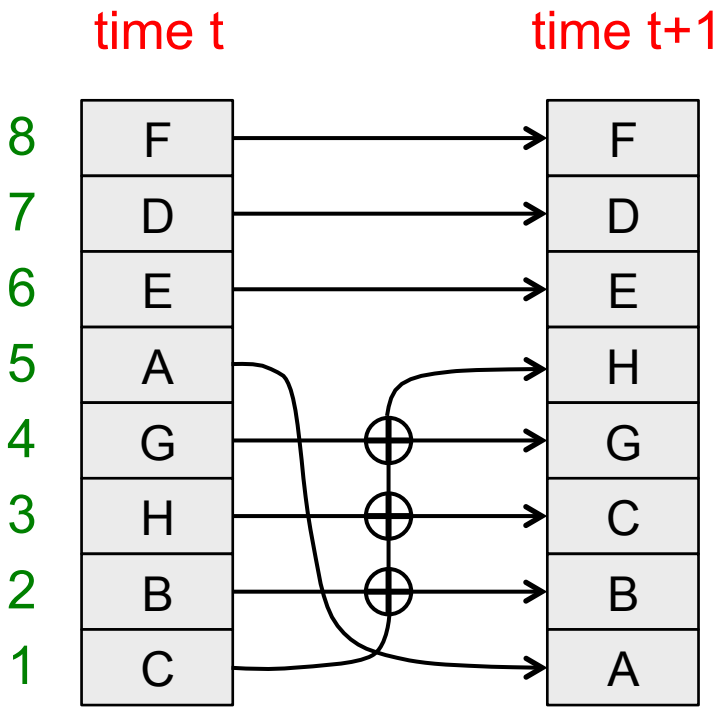
- Minimizes the number of cache misses
- Requires the oracle knowledge of future memory accesses
  - Belady, *A study of replacement algorithms for virtual-storage computers*, IBM Systems Journal, 1966
- Mattson's OPT policy
  - evict the block whose time of next access is farthest in the future
  - Mattson, Gecsei, Slutz & Traiger, *Evaluation techniques for storage hierarchies*, IBM Systems Journal, 1970
  - proved optimal by Mattson et al.

Mattson's stack

- OPT, like LRU, is a stack policy
  - At any time, the content of the cache of size N is a subset of the cache of size N+1
- Stack distance = size of the smallest cache for which the current memory access is a hit
  - LRU stack distance = reuse distance = number of distinct blocks accessed since the previous access to the current block
- Cumulative distribution of stack distances yields the number of misses for all cache sizes at once

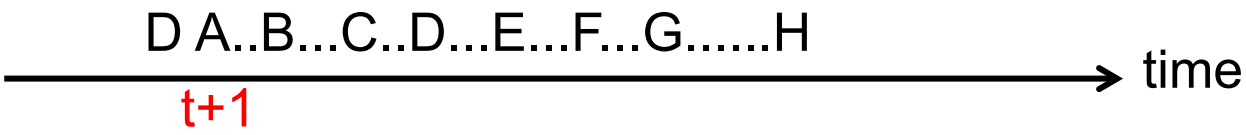
OPT stack

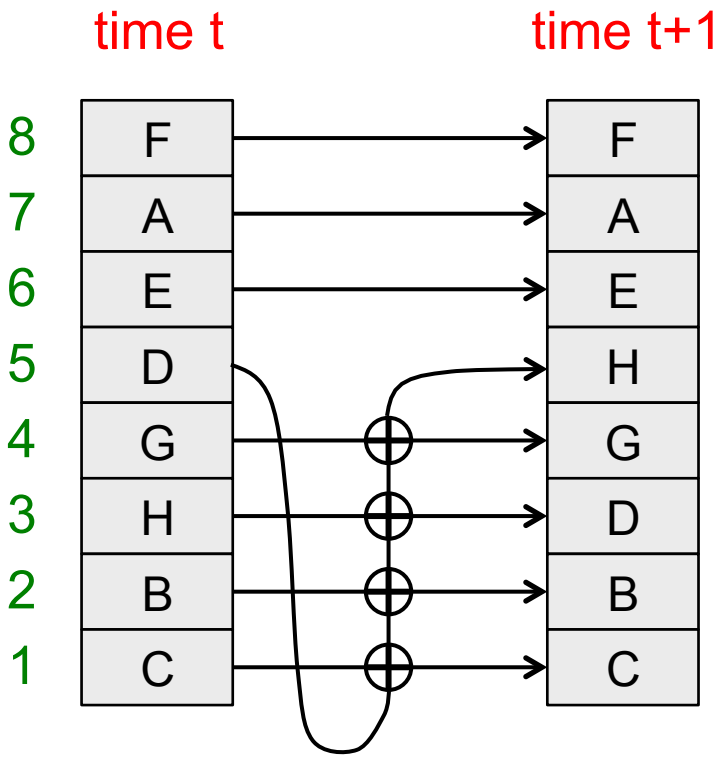




At time t, OPT stack distance = 5

OPT stack with bypassing allowed





Belady's MIN

- MIN algorithm computes the number of misses under optimal replacement

- MIN policy uses MIN algorithm to find optimal evictions
  - optimal evictions not unique
  - MIN policy different from Mattson's OPT, may evict different blocks

- 1966 paper
  - introduces MIN for a fixed cache size
  - optimality of MIN assumed, not proved

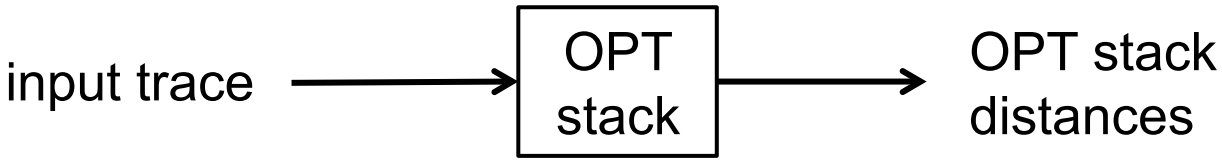
- 1974 paper
  - extends MIN to produce stack distances
  - 4 pages to prove that MIN gives same hits and misses as OPT

MIN is an online algorithm

- Hits and misses depend only on past (and present) accesses
  - but optimal evictions depend on the future
- Not stated explicitly in the 1966 paper, but it was there
- Emphasized later in the 1974 paper
- My starting point ➔ rederive this fact from OPT

New findings

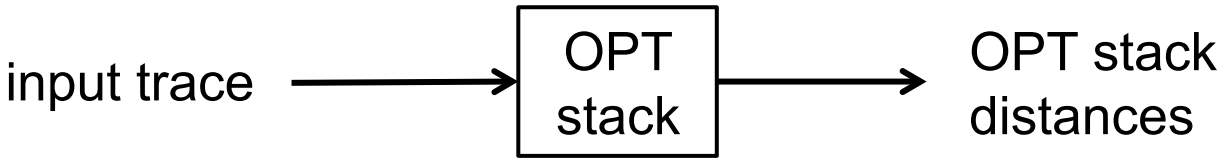
Trace of OPT stack distances



Example: 8 distinct blocks accessed randomly, bypassing allowed

1 2 3 1 2 4 1 1 2 1 3 4 2 5 1 2 3 6 1 2 4 7 1 3 2 4 8 1 5 2 1 1 6 1 1 2 7 1 3 4 1 2 5
6 1 3 7 1 4 1 2 8 3 1 2 4 5 1 3 2 6 4 7 1 2 3 1 4 5 2 8 1 3 2 1 3 1 2 1 4 5 2 1 2 6 1
7 3 4 8 2 1 1 3 5 4 1 6 2 3 4 1 7 2 3 1 2 5 3 4 6 1 8 1 2 1 3 4 5 1 7 6 2 1 3 4 5 6 1
1 2 3 1 2 3 4 5 7 1 2 3 1 6 2 4 3 8 1 2 3 5 4 1 6 7 8 2 1 3 4 1 5 2 1 3 6 4 1 5 2 3 1
7 2 8 1 3 2 4 1 1 2 1 3 4 1 1 5 6 1 2 3 4 1 5 1 2 7 8 3 1 6 2 4 5 3 1 7 2 1 3 2 4 1 3
5 1 1 6 2 1 3 4 1 8 2 1 3 4 2

Trace of OPT stack distances

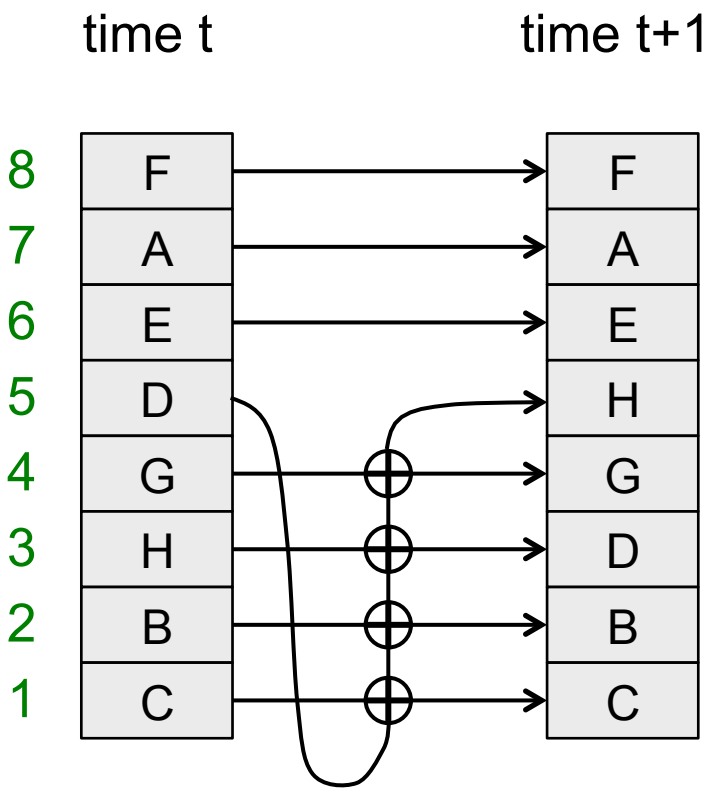


Example: 8 distinct blocks accessed randomly, bypassing allowed

1 2 3 1 2 4 1 1 2 1 3 4 2 5 1 2 3 6 1 2 4 7 1 3 2 4 8 1 5 2 1 1 6 1 1 2 7 1 3 4 1 2 5
6 1 3 7 1 4 1 2 8 3 1 2 4 5 1 3 2 6 4 7 1 2 3 1 4 5 2 8 1 3 2 1 3 1 2 1 4 5 2 1 2 6 1
7 3 4 8 2 1 1 3 5 4 1 6 2 3 4 1 7 2 3 1 2 5 3 4 6 1 8 1 2 1 3 4 5 1 7 6 2 1 3 4 5 6 1
1 2 3 1 2 3 4 5 7 1 2 3 1 6 2 4 3 8 1 2 3 5 4 1 6 7 8 2 1 3 4 1 5 2 1 3 6 4 1 5 2 3 1
7 2 8 1 3 2 4 1 1 2 1 3 4 1 1 5 6 1 2 3 4 1 5 1 2 7 8 3 1 6 2 4 5 3 1 7 2 1 3 2 4 1 3
5 1 1 6 2 1 3 4 1 8 2 1 3 4 2

The only distance that can be repeated in consecutive times is 1

Obvious...



H cannot be accessed at t+1
(blocks below H are accessed before H)

A very specific structure

1 2 3 1 2 4 1 1 2 1 3 4 2 5 1 2 3 6 1 2 4 7 1 3 2 4 8 1 5 2 1 1 6 1 1 2 7 1 3 4 1 2 5
6 1 3 7 1 4 1 2 8 3 1 2 4 5 1 3 2 6 4 7 1 2 3 1 4 5 2 8 1 3 2 1 3 1 2 1 4 5 2 1 2 6 1
7 3 4 8 2 1 1 3 5 4 1 6 2 3 4 1 7 2 3 1 2 5 3 4 6 1 8 1 2 1 3 4 5 1 7 6 2 1 3 4 5 6 1
1 2 3 1 2 3 4 5 7 1 2 3 1 6 2 4 3 8 1 2 3 5 4 1 6 7 8 2 1 3 4 1 5 2 1 3 6 4 1 5 2 3 1
7 2 8 1 3 2 4 1 1 2 1 3 4 1 1 5 6 1 2 3 4 1 5 1 2 7 8 3 1 6 2 4 5 3 1 7 2 1 3 2 4 1 3
5 1 1 6 2 1 3 4 1 8 2 1 3 4 2

Theorem (bypassing allowed)

1 2 3 1 2 4 1 1 2 1 3 4 2 5 1 2 3 6 1 2 4 7 1 3 2 4 8 1 5 2 1 1 6 1 1 2 7 1 3 4 1 2 5
6 1 3 7 1 4 1 2 8 3 1 2 4 5 1 3 2 6 4 7 1 2 3 1 4 5 2 8 1 3 2 1 3 1 2 1 4 5 2 1 2 6 1
7 3 4 8 2 1 1 3 5 4 1 6 2 3 4 1 7 2 3 1 2 5 3 4 6 1 8 1 2 1 3 4 5 1 7 6 2 1 3 4 5 6 1
1 2 3 1 2 3 4 5 7 1 2 3 1 6 2 4 3 8 1 2 3 5 4 1 6 7 8 2 1 3 4 1 5 2 1 3 6 4 1 5 2 3 1
7 2 8 1 3 2 4 1 1 2 1 3 4 1 1 5 6 1 2 3 4 1 5 1 2 7 8 3 1 6 2 4 5 3 1 7 2 1 3 2 4 1 3
5 1 1 6 2 1 3 4 1 8 2 1 3 4 2

Between 2 occurrences of a distance D > 1,
all distances less than D occur at least once

Bypassing not allowed

1 2 3 1 2 4 2 1 3 2 4 3 2 5 3 2 4 6 2 2 3 7 1 4 2 3 8 2 5 4 1 1 6 2 1 3 7 2 4 5 2
2 3 8 2 4 6 2 5 1 3 7 2 2 2 4 5 2 3 4 6 5 7 2 2 3 1 4 5 2 8 3 4 2 3 2 2 2 1 5 6 2
1 2 7 2 4 3 5 8 6 2 1 3 4 5 1 7 2 3 4 1 6 2 5 3 2 4 3 6 5 2 8 2 2 1 3 4 5 1 7 6 2
2 3 4 5 6 1 1 2 3 1 2 3 4 5 7 2 3 4 2 6 3 4 2 8 2 2 5 6 3 2 4 7 8 2 2 3 4 2 5 2 3
4 6 5 2 3 4 5 1 7 2 8 2 3 4 5 1 1 2 1 3 4 2 1 6 5 2 3 2 4 3 5 1 2 7 8 3 2 6 3 4 5
2 1 7 3 2 3 2 4 3 2 5 2 1 6 3 2 3 4 1 8 2 1 3 4 5

Between 2 occurrences of a distance D > 2,
all distances greater than 1 and less than D occur at least once

Observation

Example: 8 distinct blocks, bypassing allowed, cache of size 5
Misses occur for OPT stack distances 6,7,8

1 2 3 1 2 4 1 1 2 1 3 4 2 5 1 2 3 6 1 2 4 7 1 3 2 4 8 1 5 2 1 1 6 1 1 2 7 1 3 4 1 2 5
6 1 3 7 1 4 1 2 8 3 1 2 4 5 1 3 2 6 4 7 1 2 3 1 4 5 2 8 1 3 2 1 3 1 2 1 4 5 2 1 2 6 1
7 3 4 8 2 1 1 3 5 4 1 6 2 3 4 1 7 2 3 1 2 5 3 4 6 1 8 1 2 1 3 4 5 1 7 6 2 1 3 4 5 6 1
1 2 3 1 2 3 4 5 7 1 2 3 1 6 2 4 3 8 1 2 3 5 4 1 6 7 8 2 1 3 4 1 5 2 1 3 6 4 1 5 2 3 1
7 2 8 1 3 2 4 1 1 2 1 3 4 1 1 5 6 1 2 3 4 1 5 1 2 7 8 3 1 6 2 4 5 3 1 7 2 1 3 2 4 1 3
5 1 1 6 2 1 3 4 1 8 2 1 3 4 2

In a time interval containing 4 misses, distance 6,7 or 8 occurs twice
➔ This interval must contain at least 5 hits (distances 1,2,3,4,5)

Upper bound for the OPT miss ratio

N distinct blocks
cache size C
N > C

The long-term OPT miss ratio cannot exceed $\frac{N-C}{N}$ with bypassing

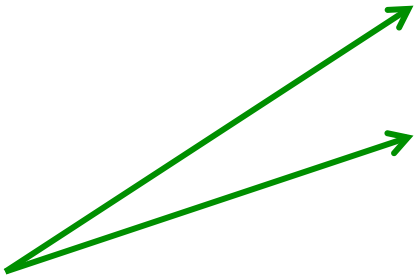
$\frac{N-C}{N-1}$ without bypassing

Upper bound for the OPT miss ratio

N distinct blocks
cache size C
N > C

The long-term OPT miss ratio cannot exceed $\frac{N-C}{N}$ with bypassing

$\frac{N-C}{N-1}$ without bypassing

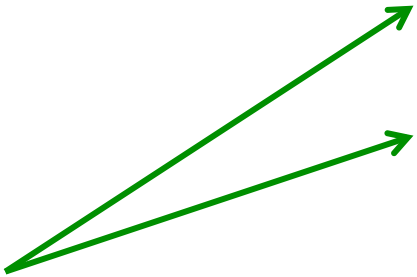


Miss ratios for a circular access pattern (ABC... ABC...)

Upper bound for the OPT miss ratio

N distinct blocks
cache size C
N > C

The long-term OPT miss ratio cannot exceed $\frac{N-C}{N}$ with bypassing

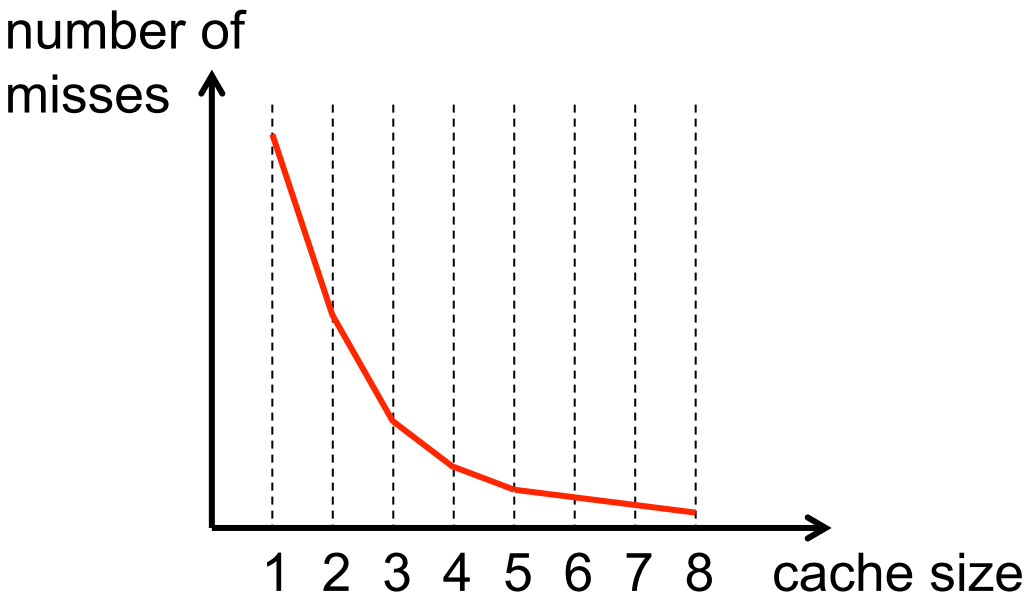


$\frac{N-C}{N-1}$ without bypassing

Miss ratios for a circular access pattern (ABC... ABC...)

Circular access patterns are the worst

Convexity of the OPT miss curve



Beckmann & Sanchez, *Talus: a simple way to remove cliffs in cache performance*, HPCA 2015

they made some assumptions about applications behavior

Convexity of the OPT miss curve

Always true

1 2 3 1 2 4 1 1 2 1 3 4 2 5 1 2 3 6 1 2 4 7 1 3 2 4 8 1 5 2 1 1 6 1 1 2 7 1 3 4 1 2 5
6 1 3 7 1 4 1 2 8 3 1 2 4 5 1 3 2 6 4 7 1 2 3 1 4 5 2 8 1 3 2 1 3 1 2 1 4 5 2 1 2 6 1
7 3 4 8 2 1 1 3 5 4 1 6 2 3 4 1 7 2 3 1 2 5 3 4 6 1 8 1 2 1 3 4 5 1 7 6 2 1 3 4 5 6 1
1 2 3 1 2 3 4 5 7 1 2 3 1 6 2 4 3 8 1 2 3 5 4 1 6 7 8 2 1 3 4 1 5 2 1 3 6 4 1 5 2 3 1
7 2 8 1 3 2 4 1 1 2 1 3 4 1 1 5 6 1 2 3 4 1 5 1 2 7 8 3 1 6 2 4 5 3 1 7 2 1 3 2 4 1 3
5 1 1 6 2 1 3 4 1 8 2 1 3 4 2

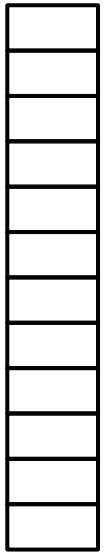
A new algorithm: OPT Tokens

- Gives the number of misses for a fixed cache size
  - Exact same misses as OPT and MIN
- Does not tell which block to evict
  - If you need this information, use OPT (or MIN)
- Online algorithm, like MIN
  - but different

Stuff needed

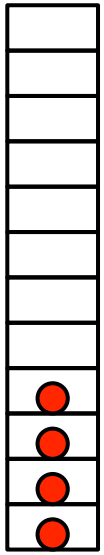
Stuff needed

- one "ladder"
  - height of the ladder = number of distinct blocks



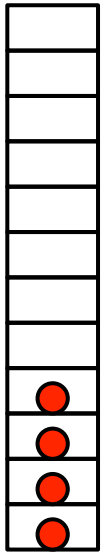
Stuff needed

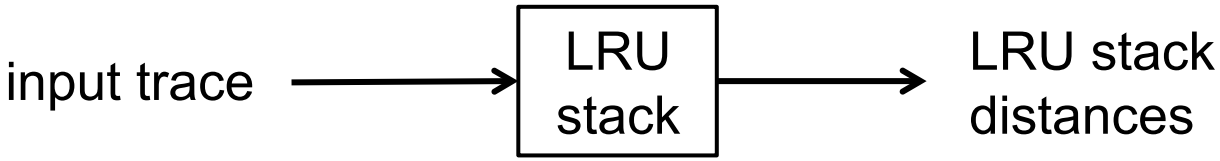
- one "ladder"
  - height of the ladder = number of distinct blocks
- C tokens
  - C = cache size (in blocks)
  - each token is at a different height



Stuff needed

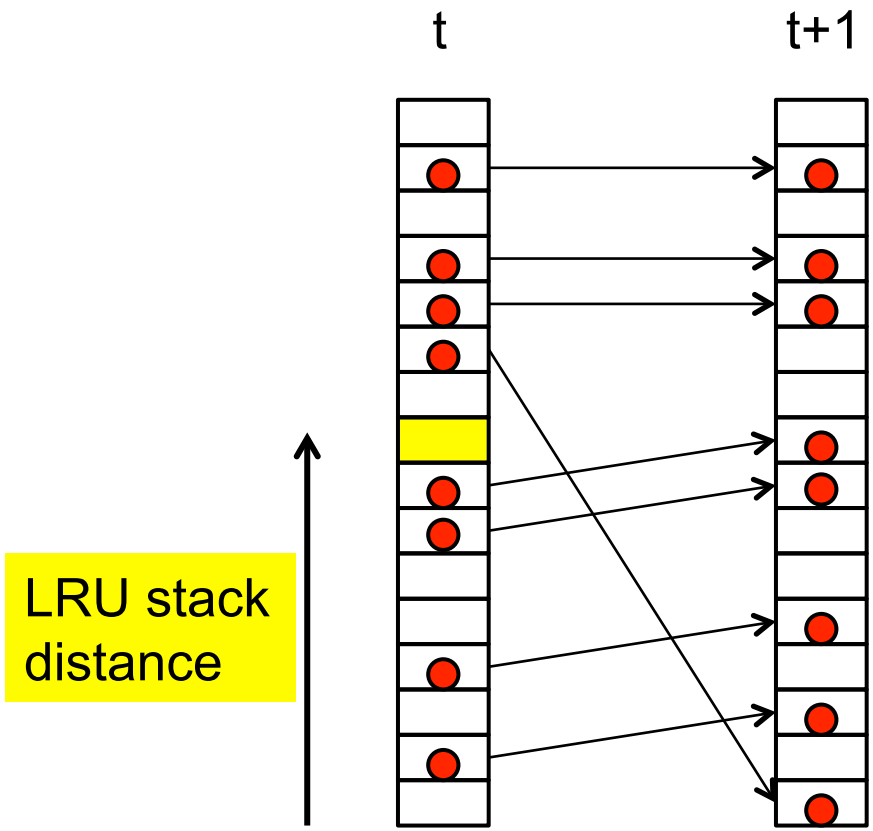
- one "ladder"
  - height of the ladder = number of distinct blocks
- C tokens
  - C = cache size (in blocks)
  - each token is at a different height
- LRU stack distances





Hit

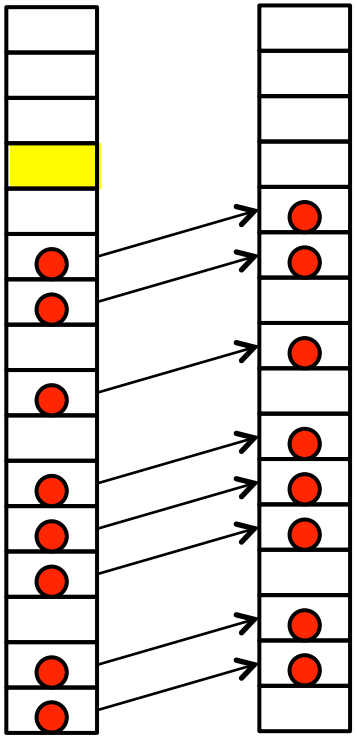
The LRU stack distance does not exceed the height of the highest token



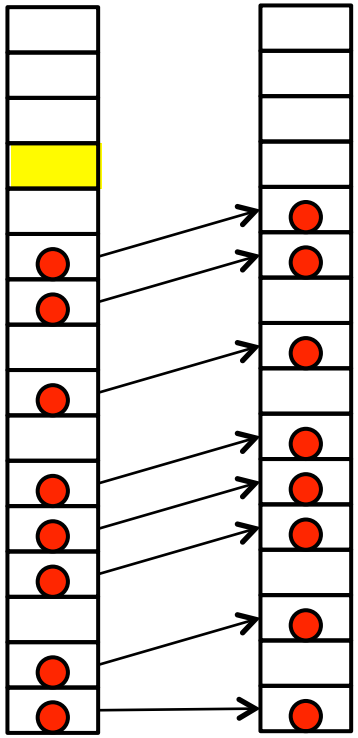
Miss

The LRU stack distance exceeds the height of the highest token

with bypassing



without bypassing



Example

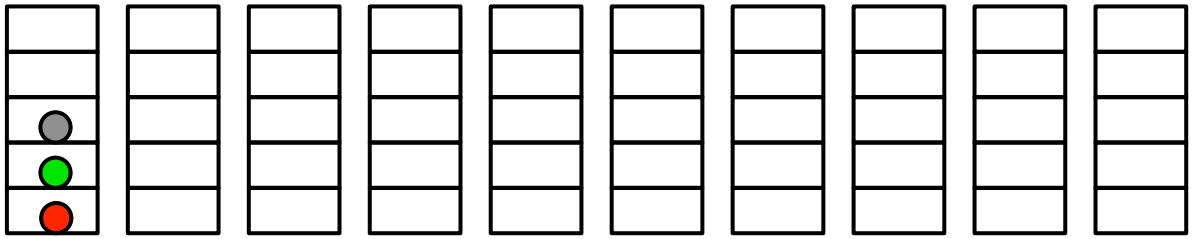
trace ABCDE ABCDE, cache size = 3, bypassing

	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5

Example

trace ABCDE ABCDE, cache size = 3, bypassing

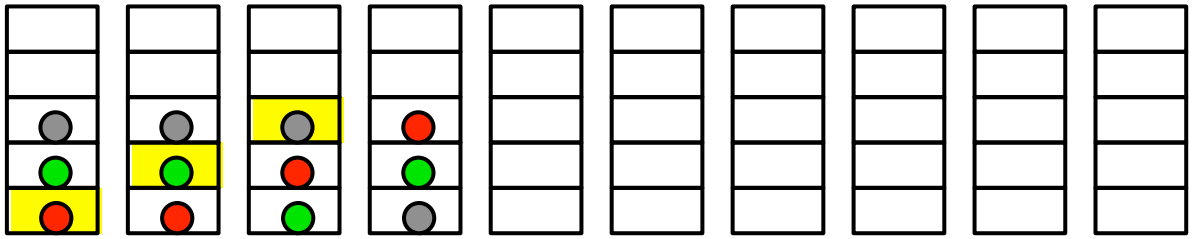
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



Example

trace ABCDE ABCDE, cache size = 3, bypassing

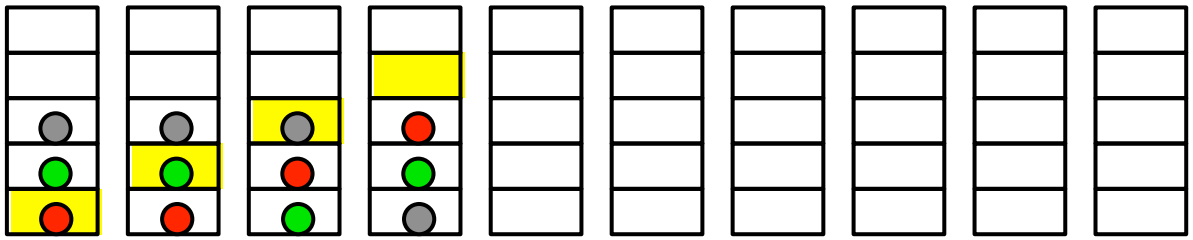
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



Example

trace ABCDE ABCDE, cache size = 3, bypassing

	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5

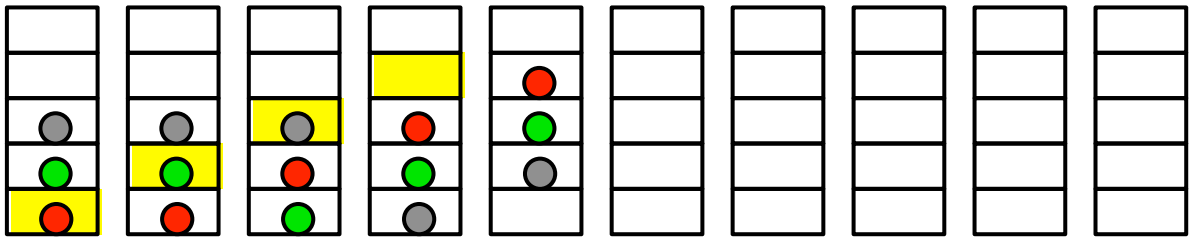


M

Example

trace ABCDE ABCDE, cache size = 3, bypassing

	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5

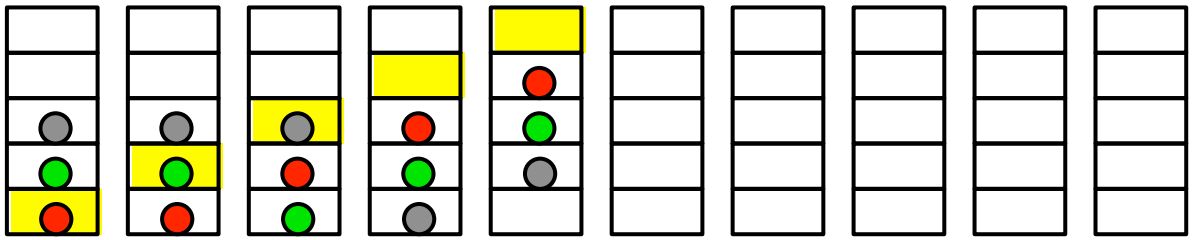


M

Example

trace ABCDE ABCDE, cache size = 3, bypassing

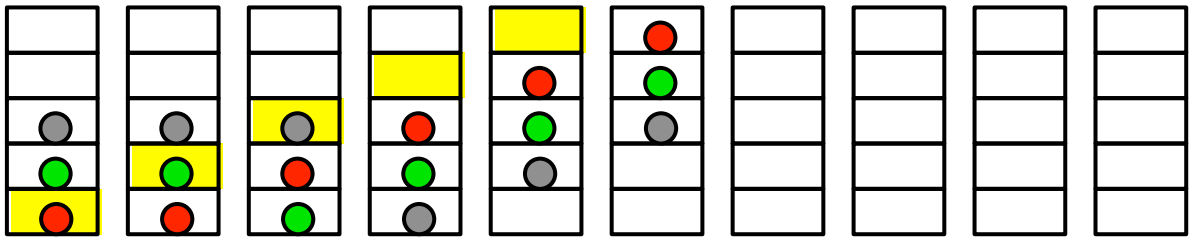
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



Example

trace ABCDE ABCDE, cache size = 3, bypassing

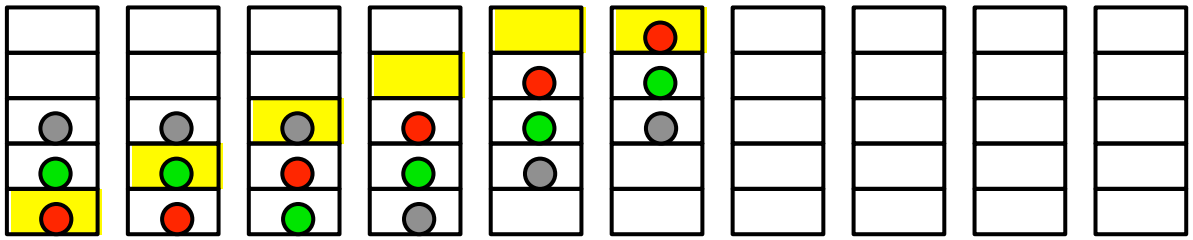
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



Example

trace ABCDE ABCDE, cache size = 3, bypassing

	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5

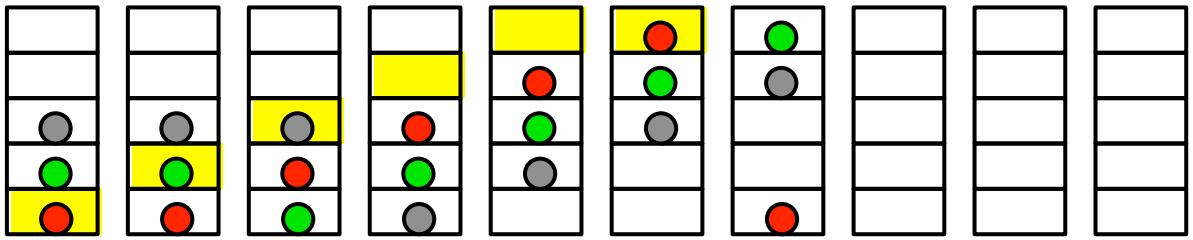


M M H

Example

trace ABCDE ABCDE, cache size = 3, bypassing

	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5

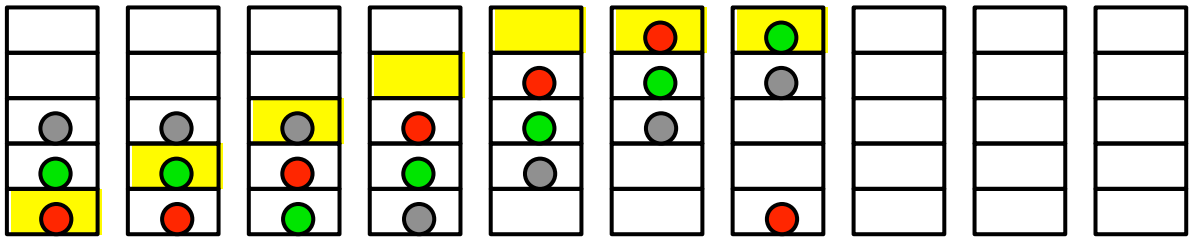


M M H

Example

trace ABCDE ABCDE, cache size = 3, bypassing

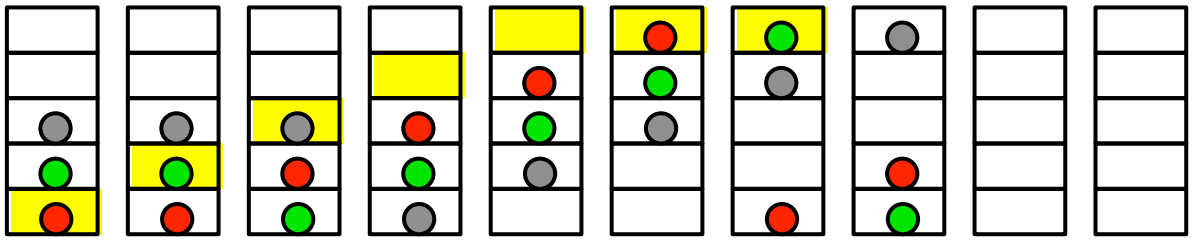
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



Example

trace ABCDE ABCDE, cache size = 3, bypassing

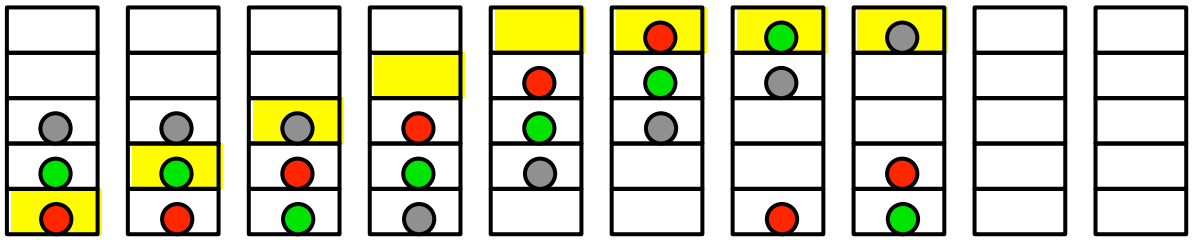
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



Example

trace ABCDE ABCDE, cache size = 3, bypassing

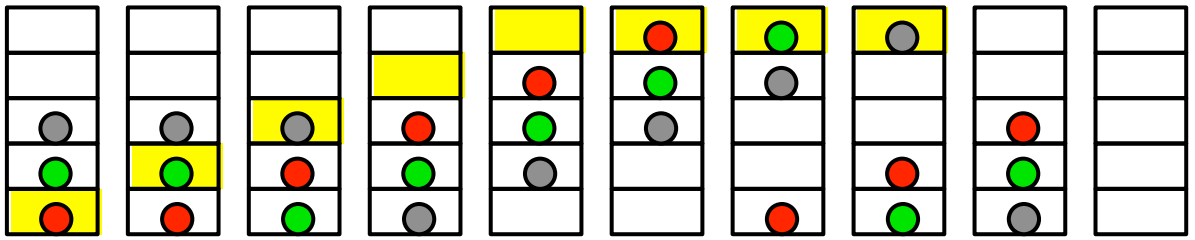
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



Example

trace ABCDE ABCDE, cache size = 3, bypassing

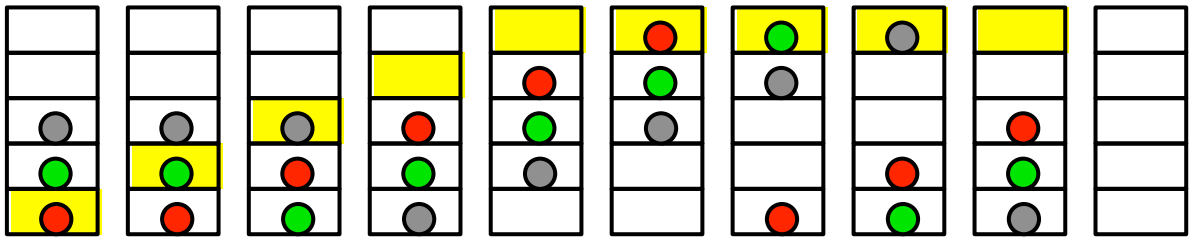
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



Example

trace ABCDE ABCDE, cache size = 3, bypassing

	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5

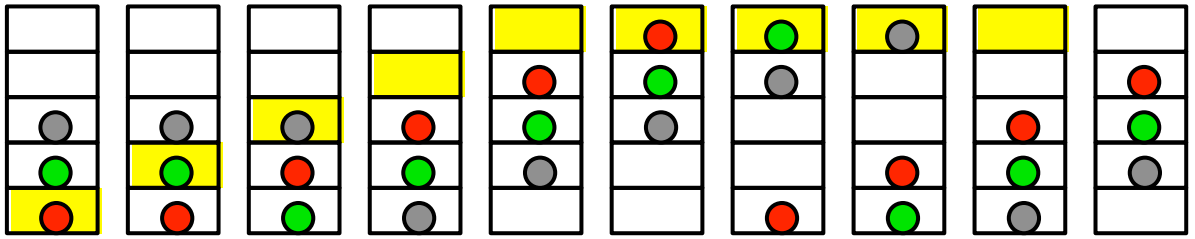


M M H H H M

Example

trace ABCDE ABCDE, cache size = 3, bypassing

	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5

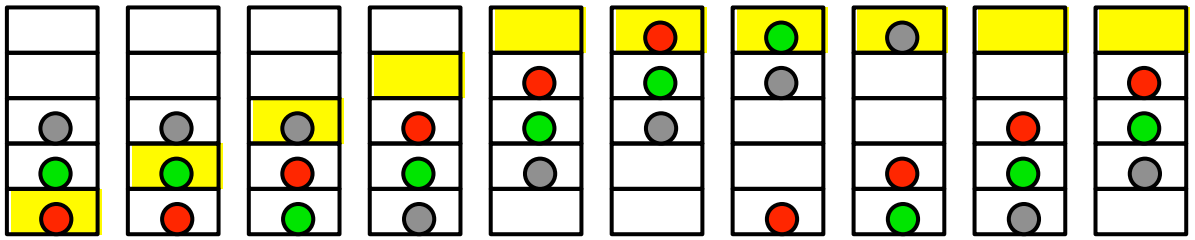


M M H H H M

Example

trace ABCDE ABCDE, cache size = 3, bypassing

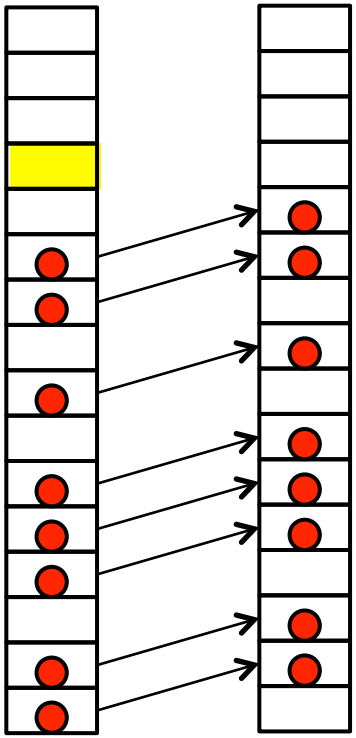
	A	B	C	D	E	A	B	C	D	E
LRU stack distance	1	2	3	4	5	5	5	5	5	5



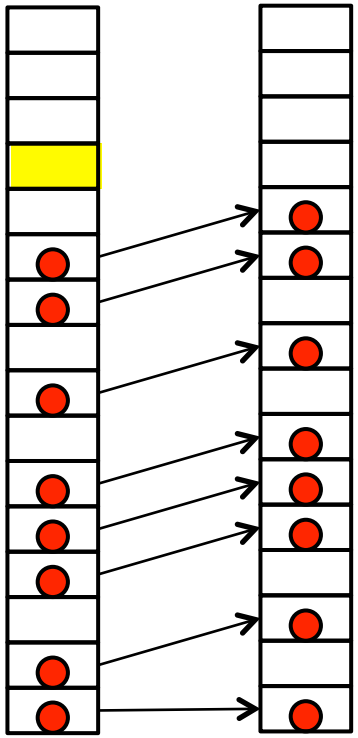
Bypassing vs. no bypassing

Identical except for one token

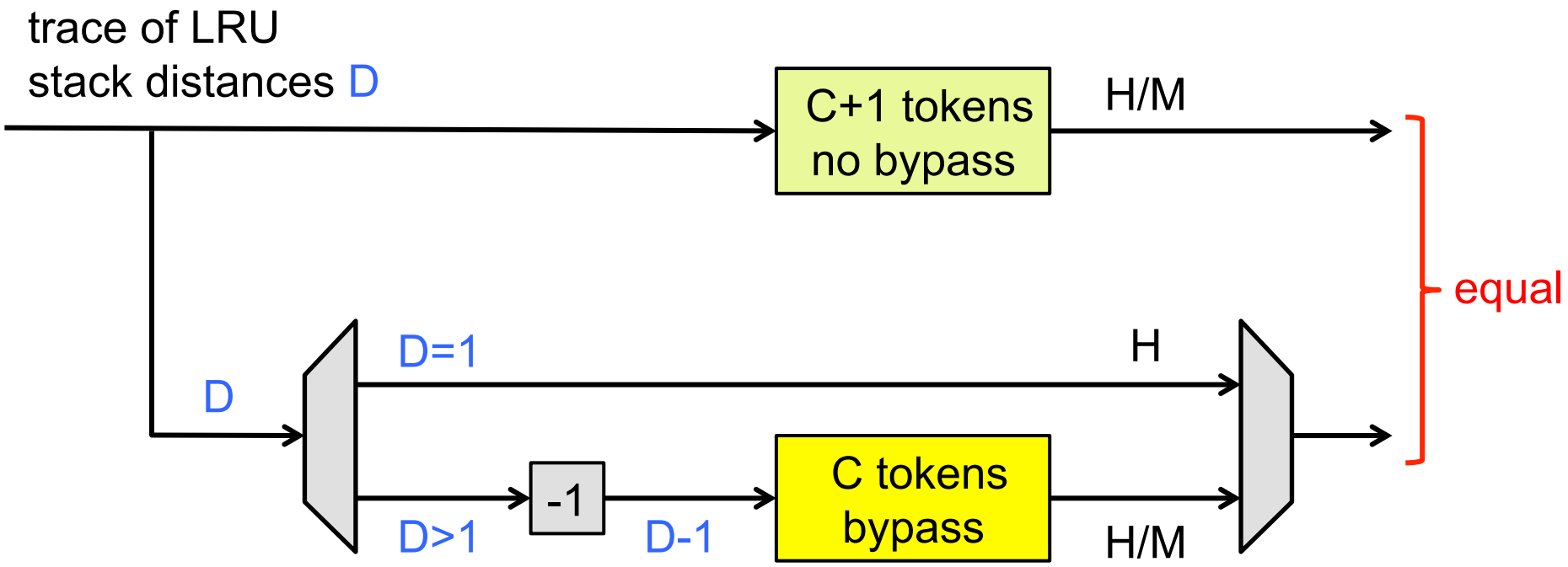
with bypassing



without bypassing



Equivalence



More results in the paper

Thanks for your attention!