LIRS: Low Inter-reference Recency Set Replacement for VM and Buffer Caches

Xiaodong Zhang

Ohio State University

In collaborations with

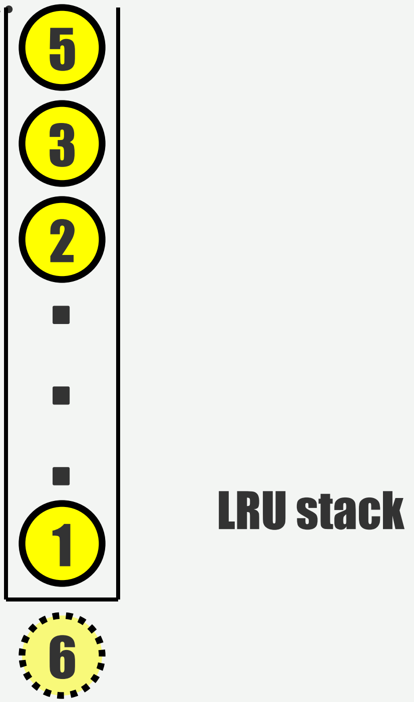
Song Jiang (Wayne State University)

Least Recent Used (LRU) Replacement

- LRU is most commonly used replacement for data management.
- Blocks are ordered by recency in the LRU stack.
- Blocks enter from the top, and leave from bottom.

The stack is long, the bottom is the only exit.

A block evicted from the bottom of the stack should have been evicted much earlier !



The Problem of LRU Replacement

Inability to cope with weak access locality

- File scanning: one-time accessed blocks are not replaced timely; (e.g. 50% disk data in NCAR only used once).
- Loop-like accesses: blocks to be accessed soonest can be unfortunately replaced;
- Accesses with distinct frequencies: Frequently accessed blocks can be unfortunately replaced.

Reasons for LRU to Fail but Powerful

- Why LRU fails sometimes?
  - A recently used block will not necessarily be used again or soon.
  - The prediction is based on a single source information.
- Why it is so widely used?
  - Simplicity: an easy and simple data structure.
  - Works well for accesses following LRU assumption.

The Challenges of Addressing the LRU problem

Two types of efforts to improve/replace LRU have been made:

- Case by case; or
- Building complex structure with high runtime overhead

Our contributions in SIGMETRICS’02 (Jiang and Zhang)

- Address the limits of LRU fundamentally.
- Retain the low overhead and strong locality merits of LRU.
- Widely adopted in buffer management in production systems.

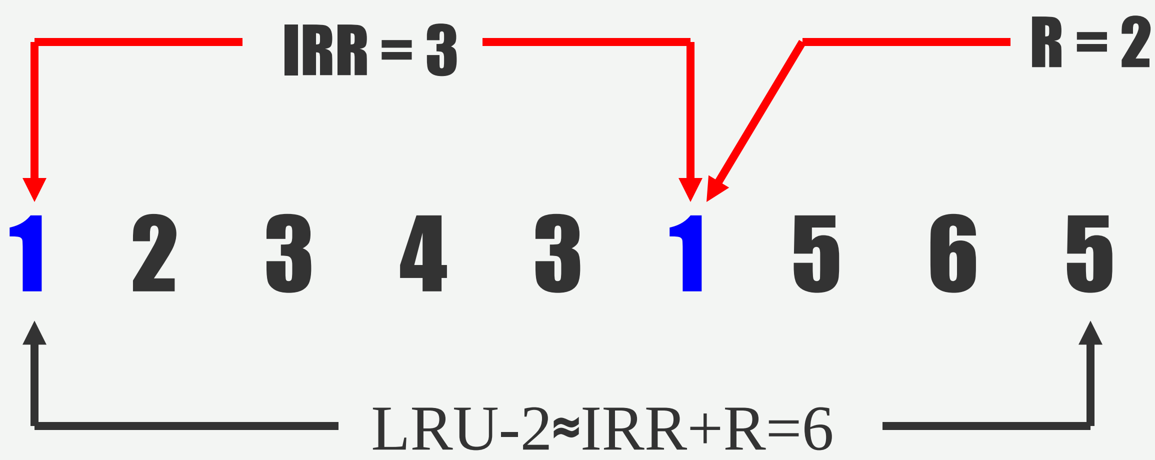
Related Work

- Aided by user-level hints
  - Application-hinted caching and prefetching [OSDI, SOSP, ...]
  - rely on users` understanding of data access patterns.
- Detection and adaptation of access regularities
  - SEQ, EELRU, DEAR, AFC, UBM [OSDI, SIGMETRICS …]
  - case-by-case oriented approaches
- Tracing and utilizing deeper history information
  - LRFU, LRU-k, 2Q, ARC (VLDB, SIGMETRICS, SIGMOD, FAST …)
  - Implementation, runtime overhead, and suboptimal performance

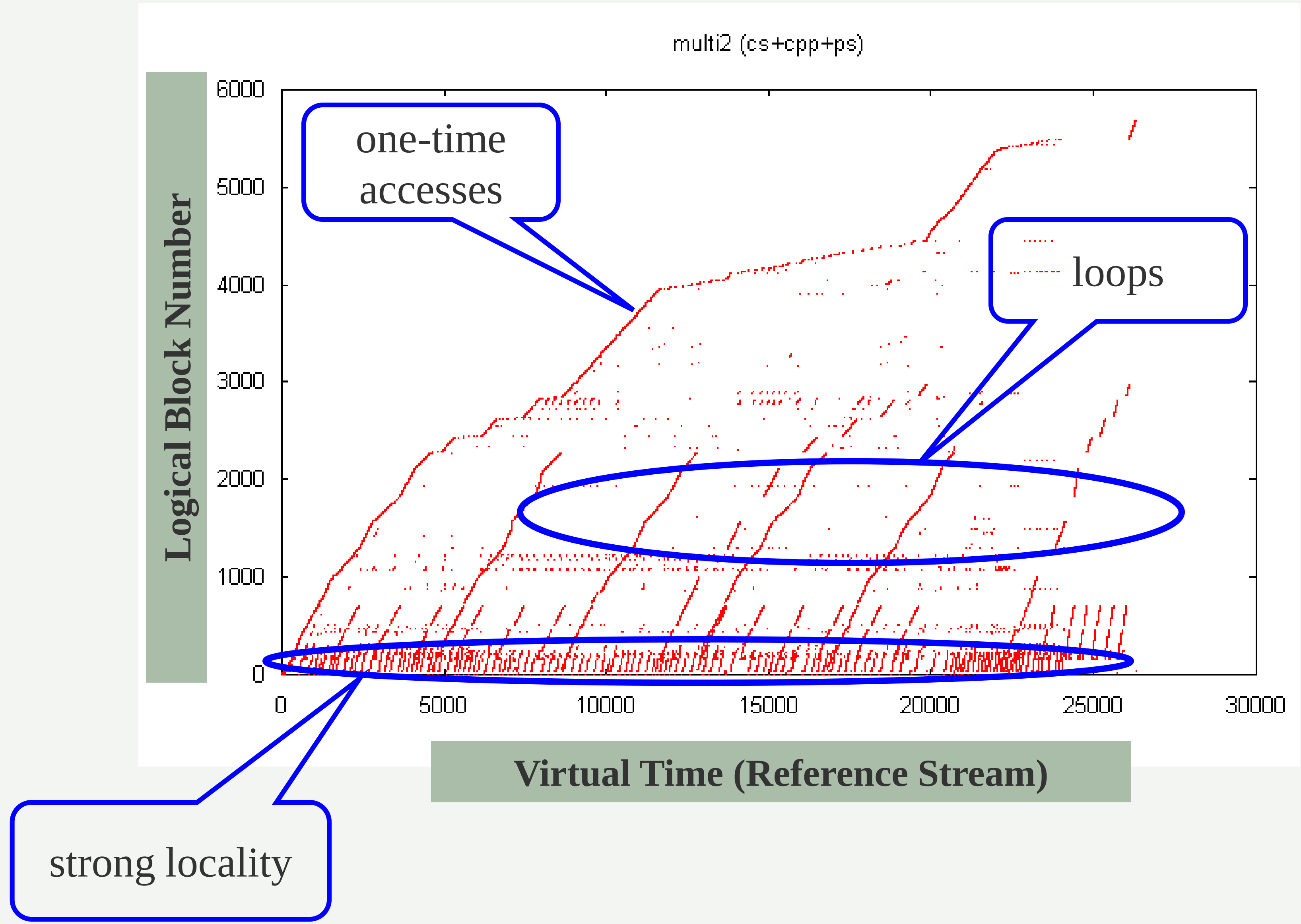
Inter-Reference Recency (IRR)

IRR (= ``**reuse distance**", 1970) of a block: the number of other unique blocks accessed between two consecutive references to the block.

Recency: the number of other unique blocks accessed from last reference to the current time.

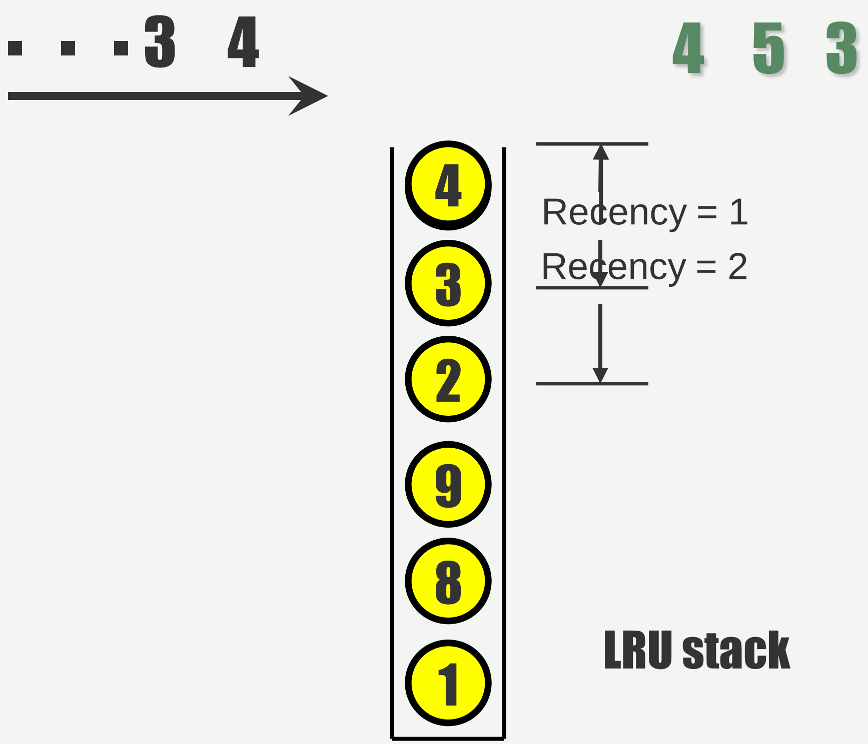


Diverse Locality Patterns on Access Map



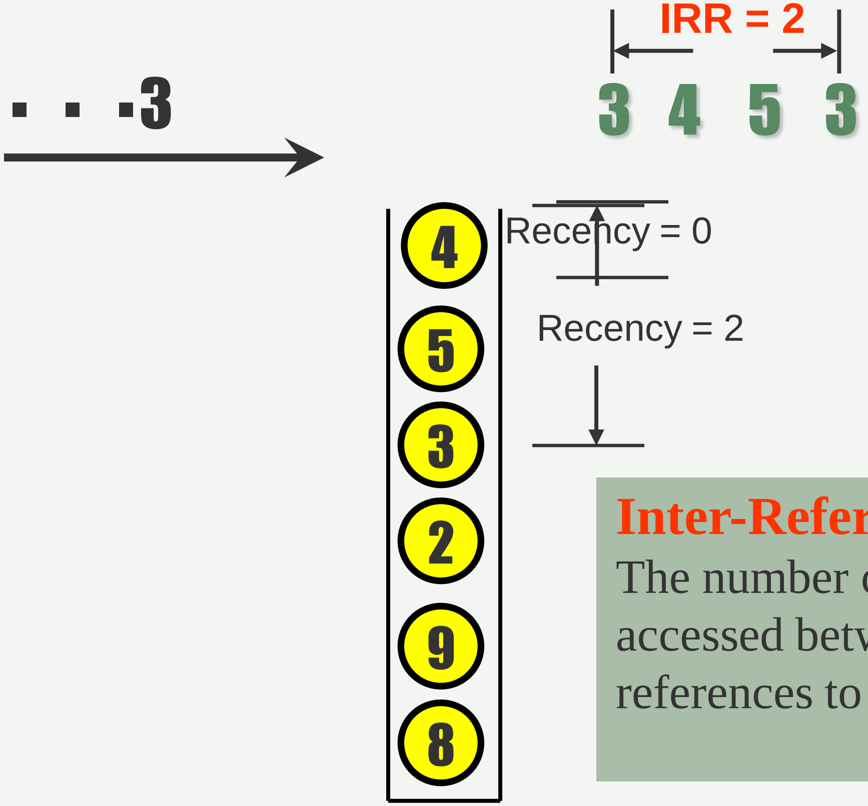
Locality Quantification Limit in LRU Stack

- Blocks are ordered by recency;
- Blocks enter from the stack top, and leave from its bottom;



LRU Stack

- Blocks are ordered by recency in the LRU stack;
- Blocks enter from the stack top, and leave from its bottom;



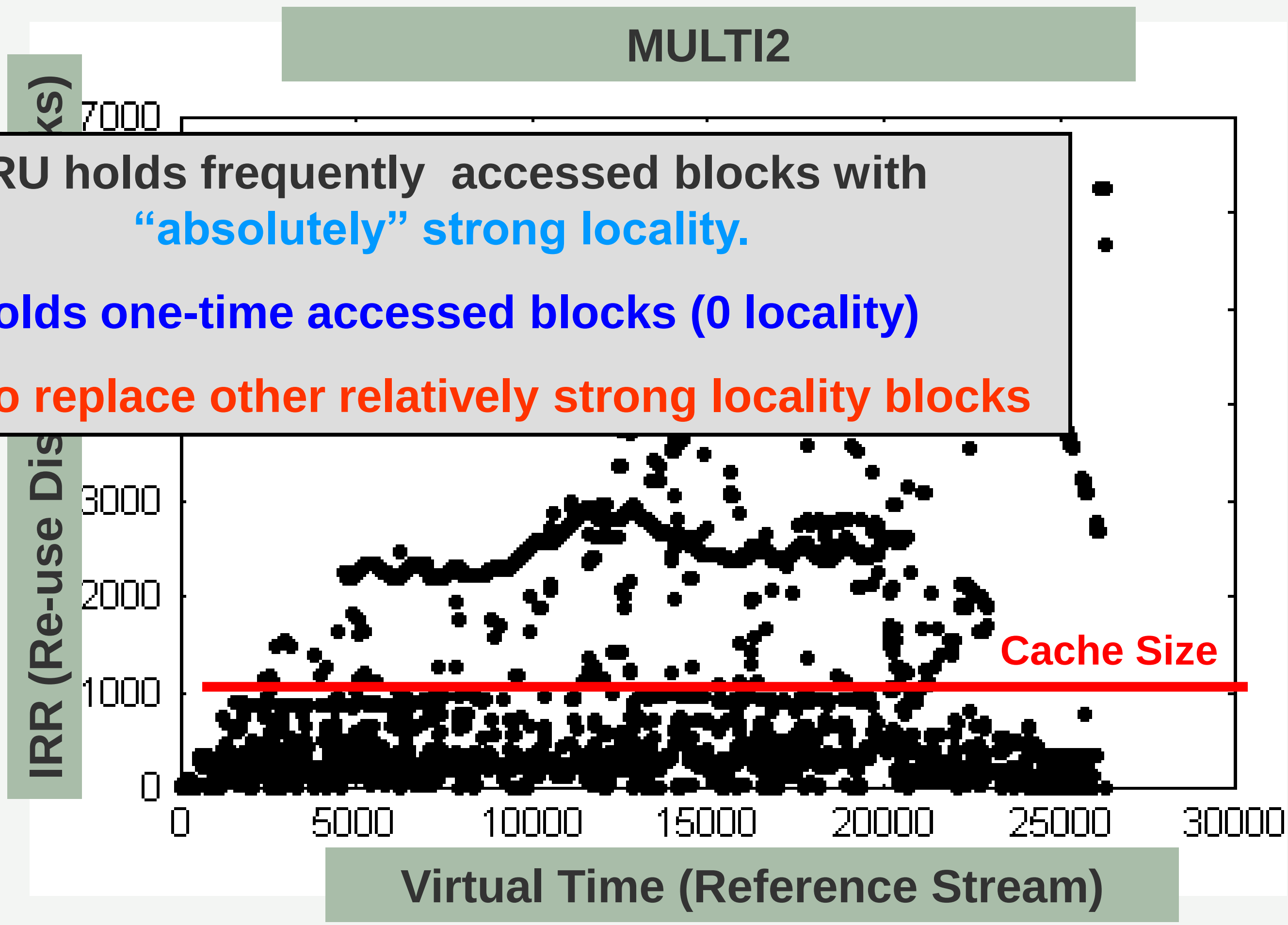
Inter-Reference Recency (IRR)
The number of other unique blocks accessed between two consecutive references to the block.

Locality Strength

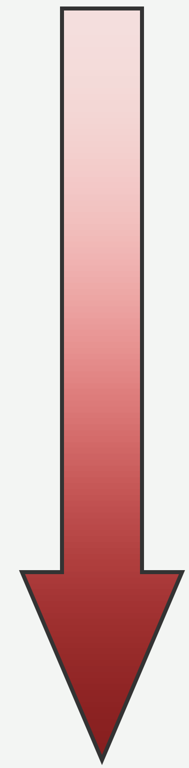
LRU holds frequently accessed blocks with
“absolutely” strong locality.

holds one-time accessed blocks (0 locality)

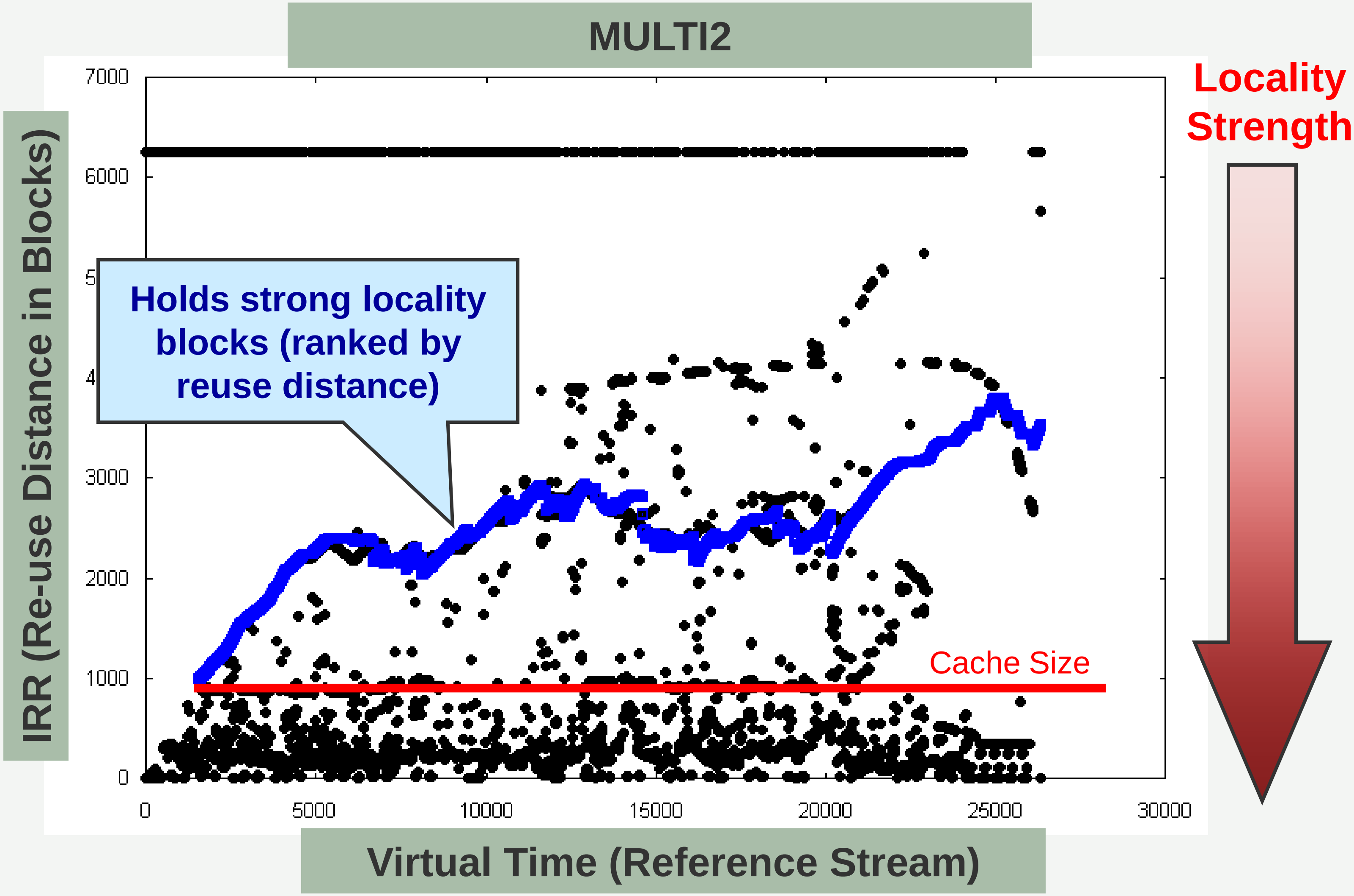
Likely to replace other relatively strong locality blocks



Locality Strength



Looking for Blocks with Strong Locality

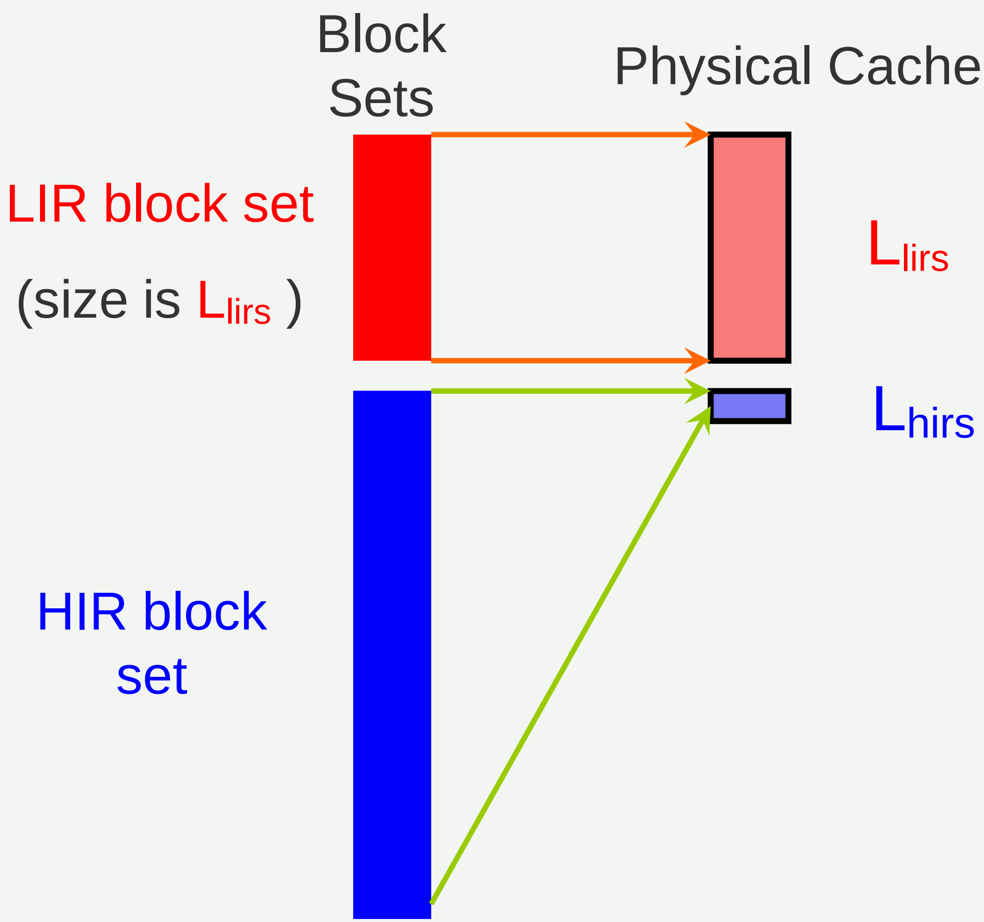


Basic Ideas of LIRS

- A high reuse distance (IRR) block is not used often.
  - High IRR blocks are selected for replacement.
- Recency is used as a second reference.
- LIRS: Low Inter-reference Recency Set algorithm
  - Keep Low reuse distance (IRR) blocks in buffer cache.
- Foundations of LIRS:
  - effectively use multiple sources of access information.
  - Responsively determine and change the status of each block.
  - Low cost implementations.

Data Structure: Keep LIR Blocks in Cache

Low **IRR** (LIR) blocks and **H**igh **IRR** (HIR) blocks



Cache size

$L = L_{lirs} + L_{hirs}$

Replacement Operations of LIRS

$L_{lirs}=2$, $L_{hirs}=1$

	V time / Blocks	1	2	3	4	5	6	7	8	9	10	R	IRR
LIR	A	X					X		X			1	**1**
LIR	B			X		X						3	**1**
	C				X							4	inf
	D		X					X				2	3
HIR	E									X		**0**	inf

LIR block set = {A, B}, HIR block set = {C, D, E}

E becomes a resident HIR determined by its low recency

Which Block is replaced ? <u>Replace an HIR Block</u>

D is referenced at time 10

V time / Blocks	1	2	3	4	5	6	7	8	9	10	R	IRR
A	X					X		X			1	1
B			X		X						3	1
C				X							4	inf
D		X					X			X	0	3
replaced → E									X		1	Inf

The resident HIR block E is replaced !

How is LIR Set Updated? LIR Block Recency is Used

V time / Blocks	1	2	3	4	5	6	7	8	9	10	R	IRR
A	X					X		X			2	1
B			X		X						3	1
C				X							4	inf
D		X					X			X	0	2
									X		1	inf

Which set, HIR or LIR should D belong to?

Compare its IRR with recency of LIR.

Recency reflects the most updated status.

After D is Referenced at Time 10

	V time / Blocks	1	2	3	4	5	6	7	8	9	10	R	IRR
LIR	A	X					X		X			2	1
HIR	B			X		X						3	1
	C				X							4	inf
LIR	D		X					X			X	0	2
	E									X		1	Inf

D enters LIR set, and B is demoted to HIR set

Because D`s IRR< R_{max} in LIR set

The Power of LIRS Replacement

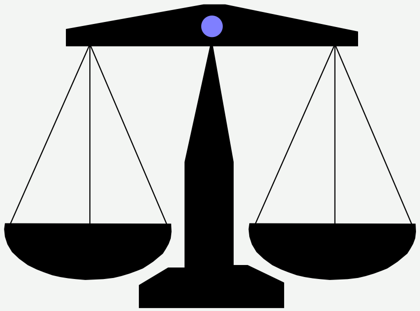
Capability to cope with weak access locality

- File scanning: one-time access blocks will be replaced timely; (due to their high IRRs)
- Loop-like accesses: blocks to be accessed soonest will NOT be replaced; (due to an MRU effect of HIR blocks)
- Accesses with distinct frequencies: Frequently accessed blocks in short reuse distance will NOT be replaced. (dynamic status changes)

LIRS Efficiency: *O(1)*

IRR** **HIR

(New IRR of a HIR block)



Rmax

(Maximum Recency of LIR blocks)

Can *O(LIRS) = O(LRU) = O(1)?*

YES! this efficiency is achieved by our LIRS stack.

- Both recencies and useful IRRs are automatically recorded.
- ***Rmax*** of the block in the stack bottom is larger than IRRs of others.
- No comparison operations are needed.

LIRS Operations

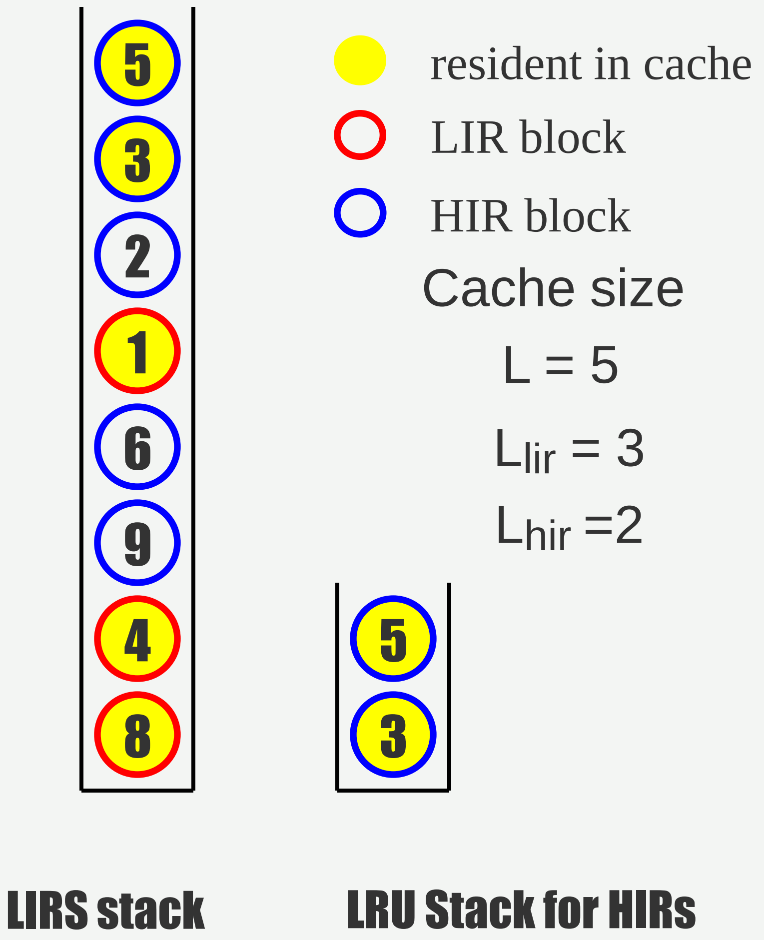
• Initialization: All the referenced blocks are given an LIR status until LIR block set is full.

We place resident HIR blocks in a small LRU Stack.

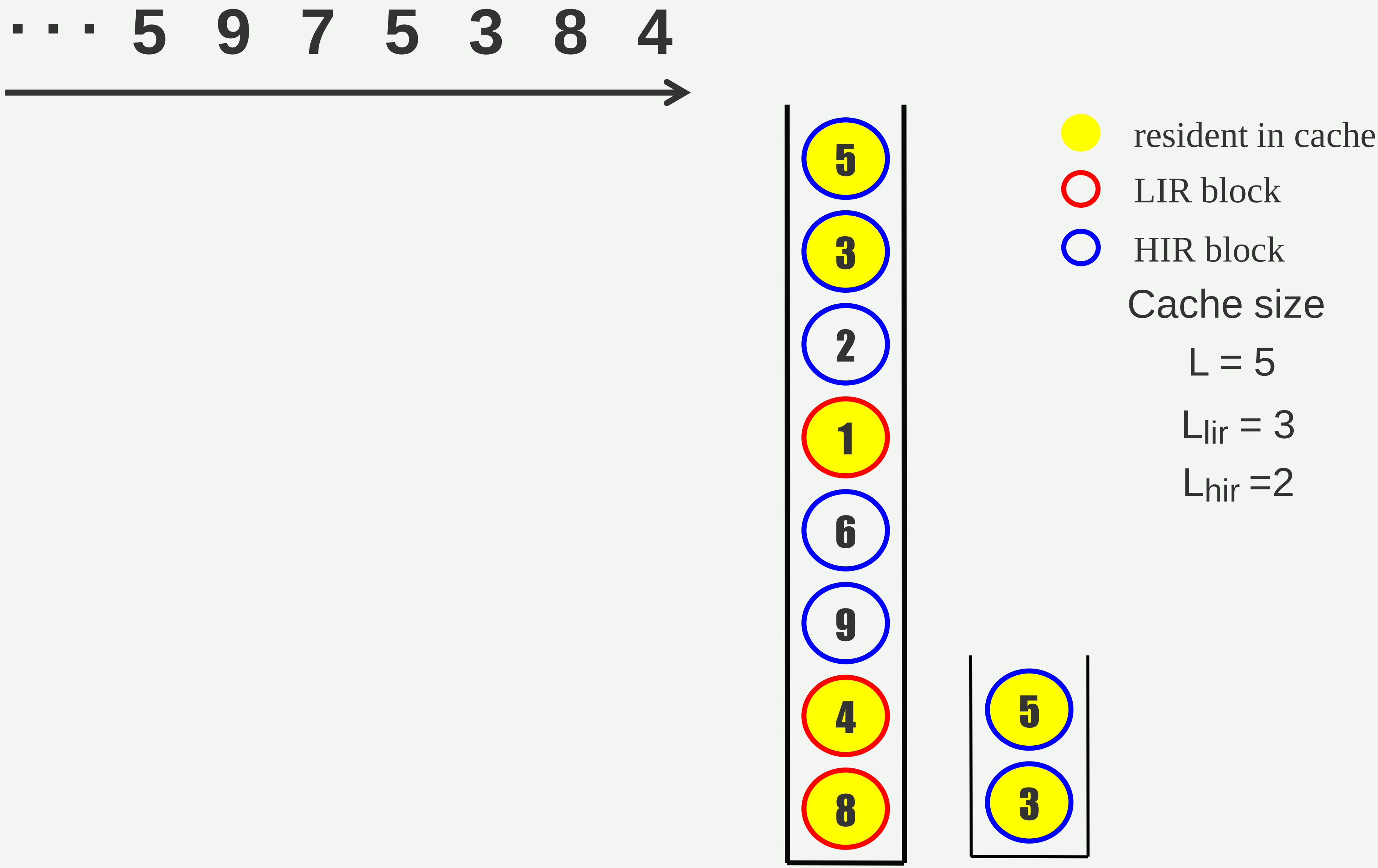
• Upon accessing an LIR block (a hit)

• Upon accessing a resident HIR block (a hit)

• Upon accessing a non-resident HIR block (a miss)

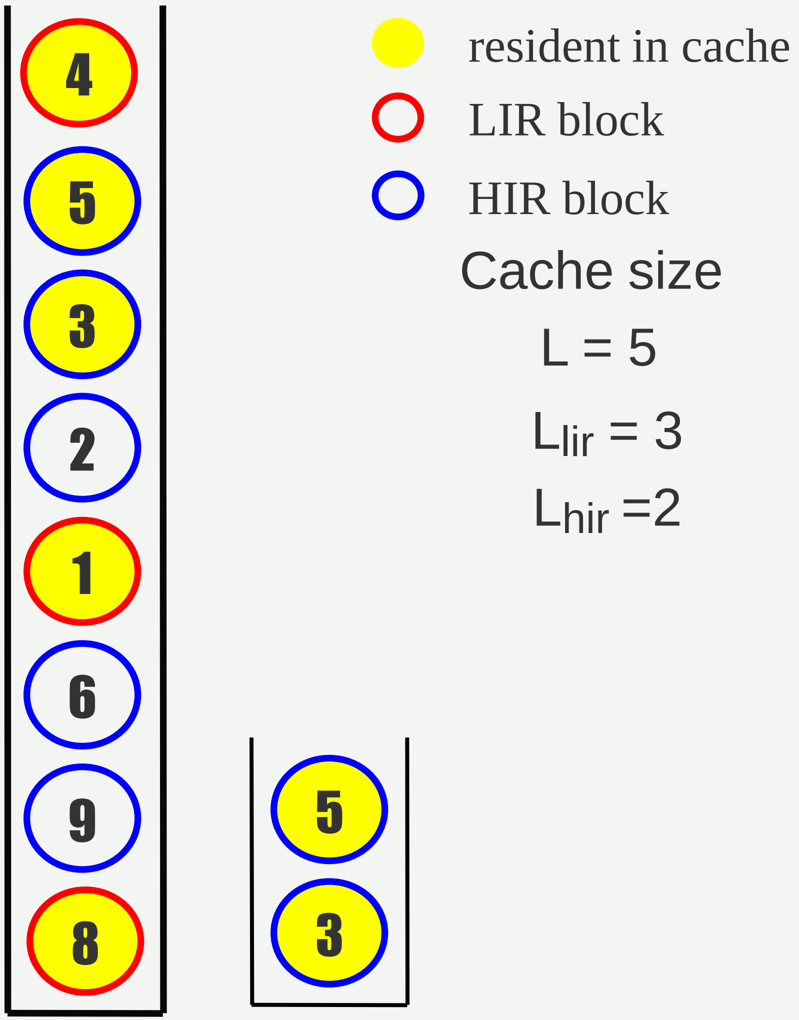


Access an LIR Block (a Hit)

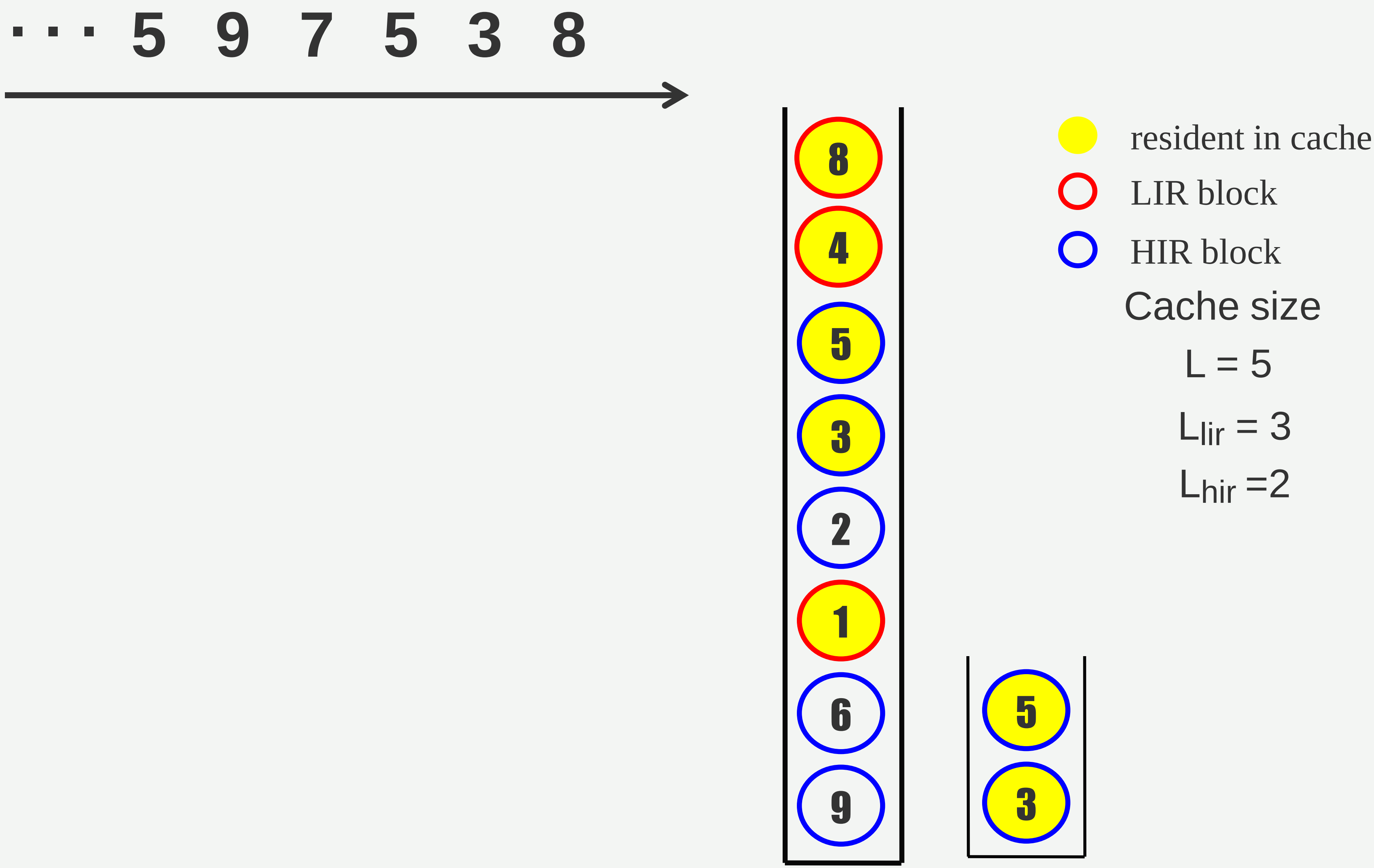


Access an LIR Block (a Hit)

· · · 5 9 7 5 3 8

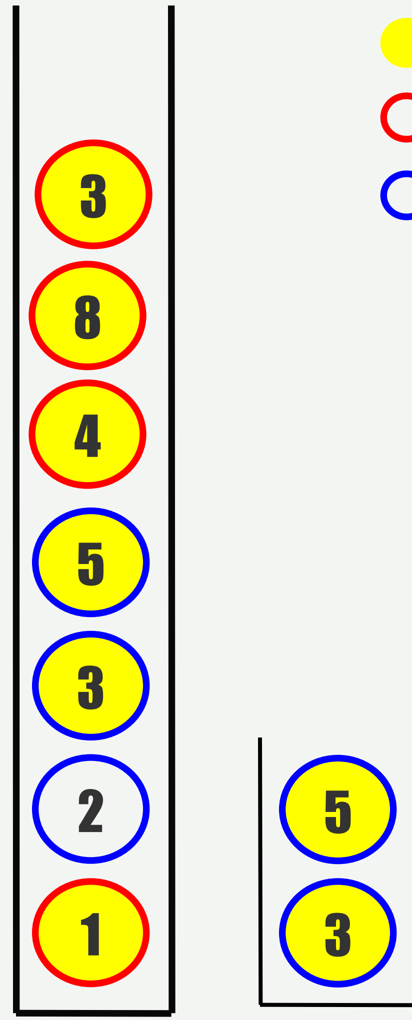


Access an LIR block (a Hit)



Access a Resident HIR Block (a Hit)

. . . 5 9 7 5 3



resident in cache

LIR block

HIR block

Cache size

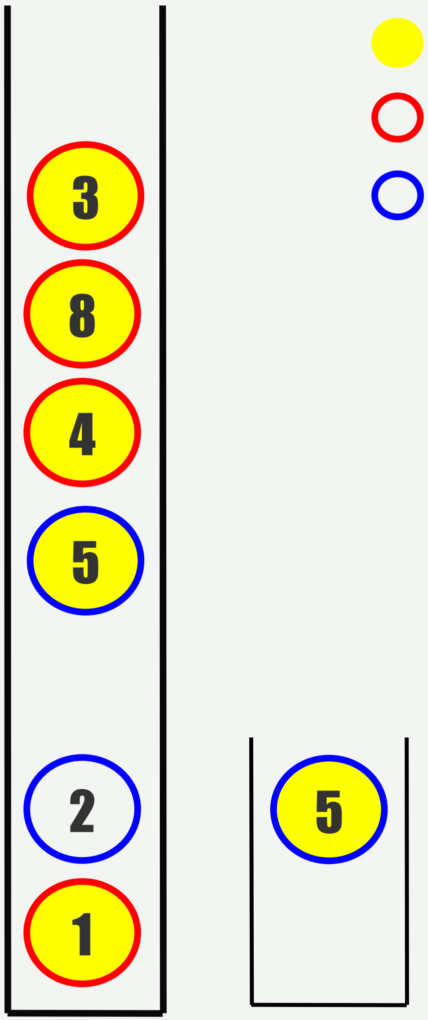
L = 5

L_{lir} = 3

L_{hir} =2

Access a Resident HIR Block (a Hit)

· · · 5 9 7 5 3



resident in cache

LIR block

HIR block

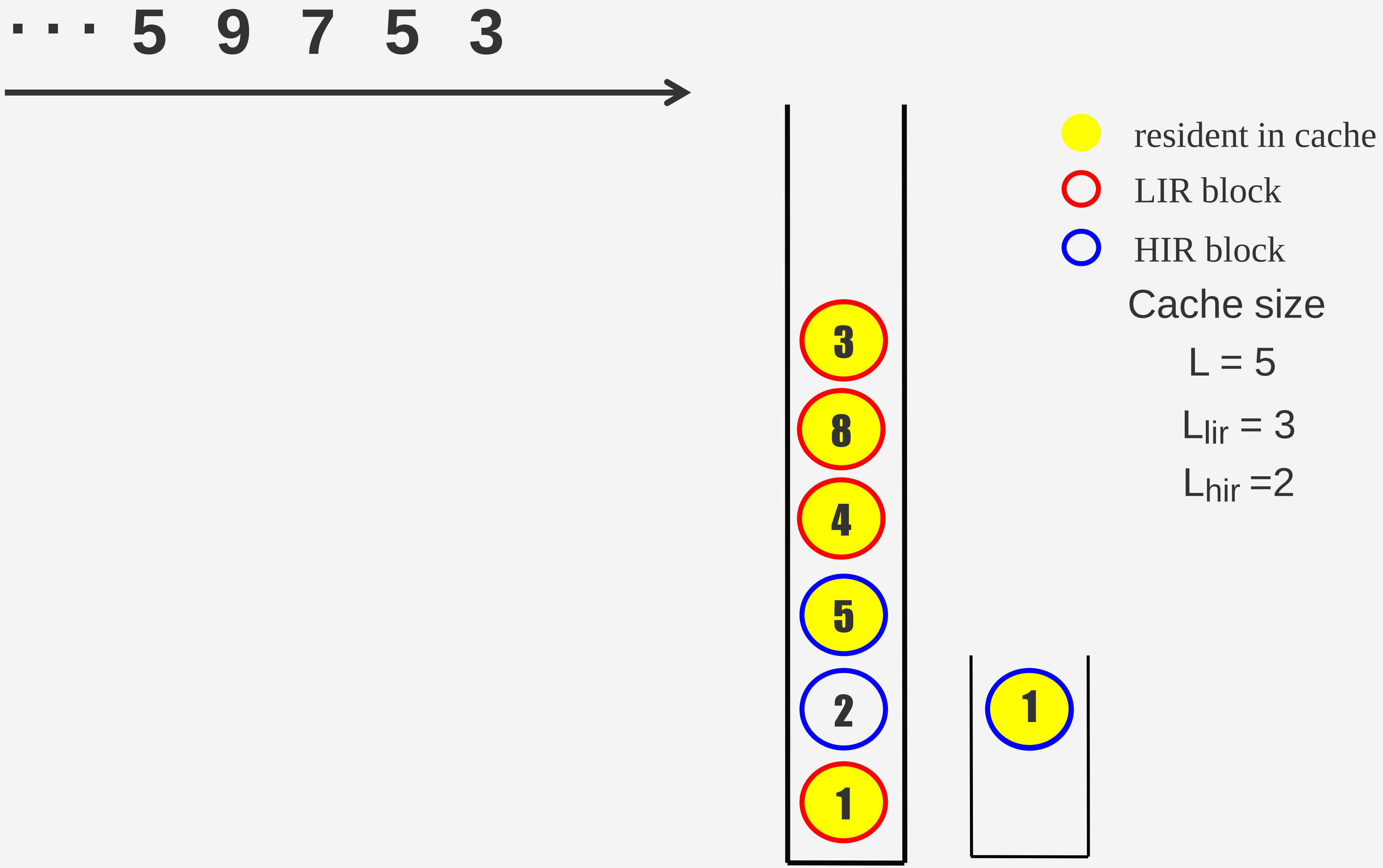
Cache size

L = 5

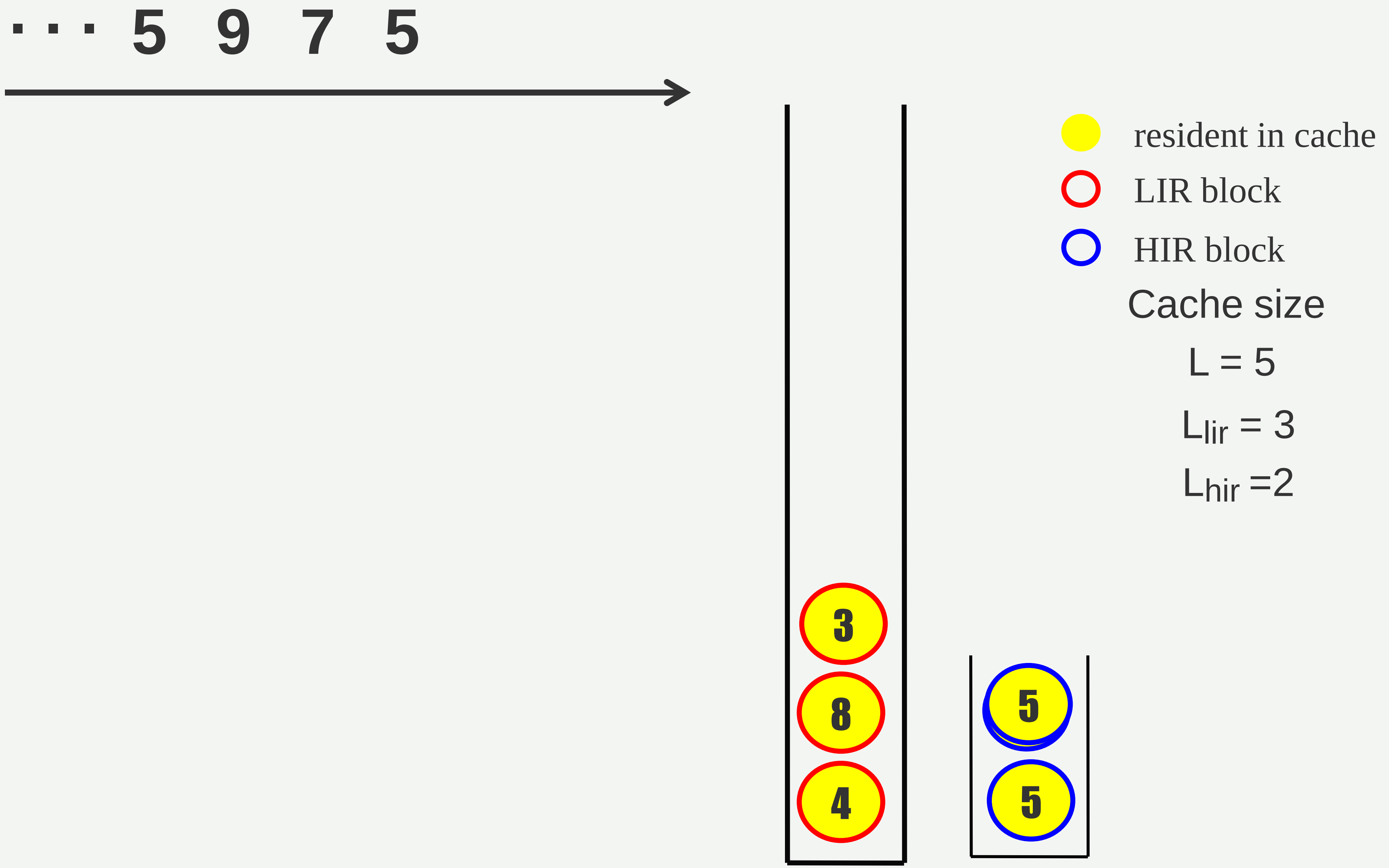
L_{lir} = 3

L_{hir} =2

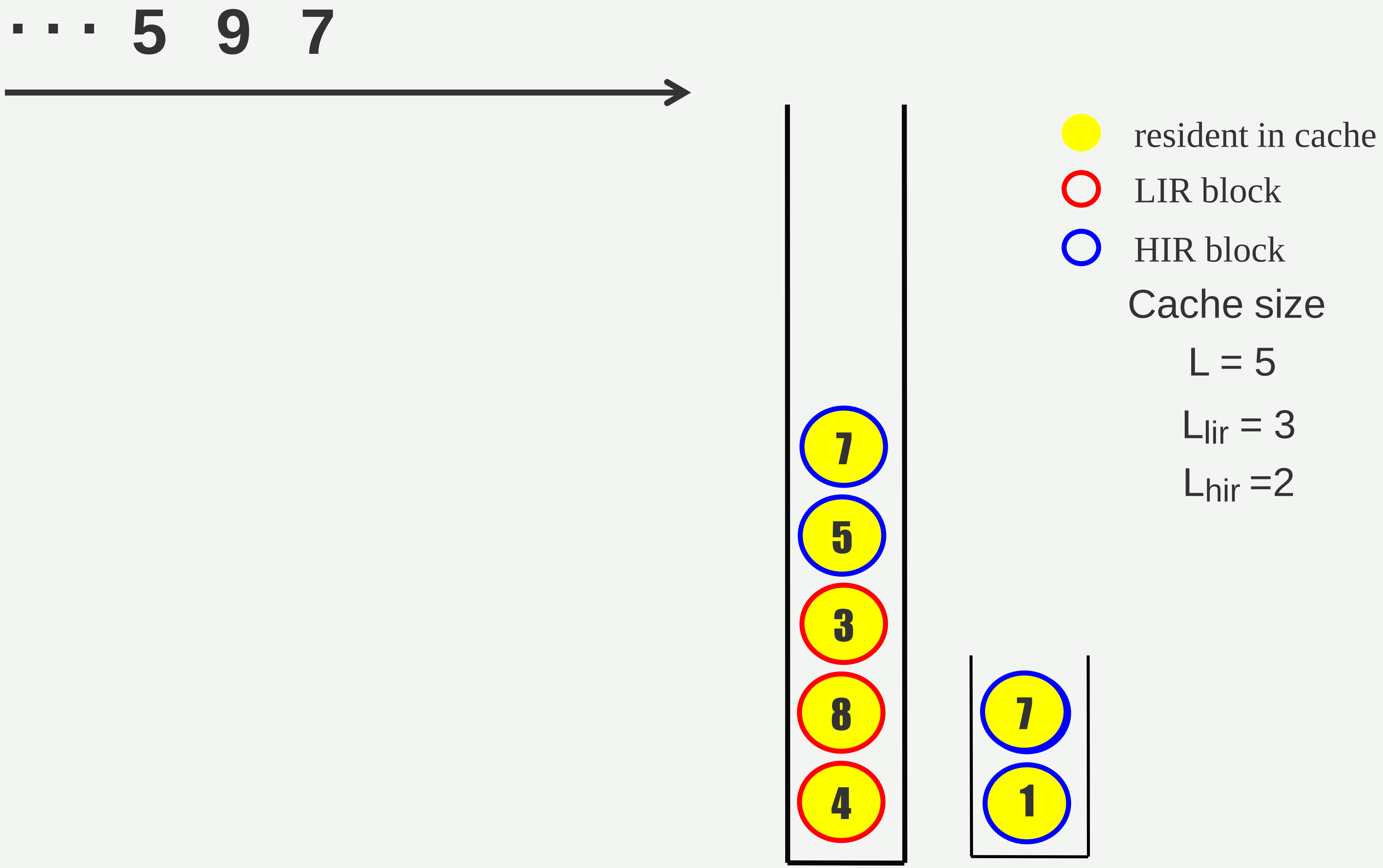
Access a Resident HIR Block (a Hit)



Access a Resident HIR Block (a Hit)



Access a Non-Resident HIR block (a Miss)



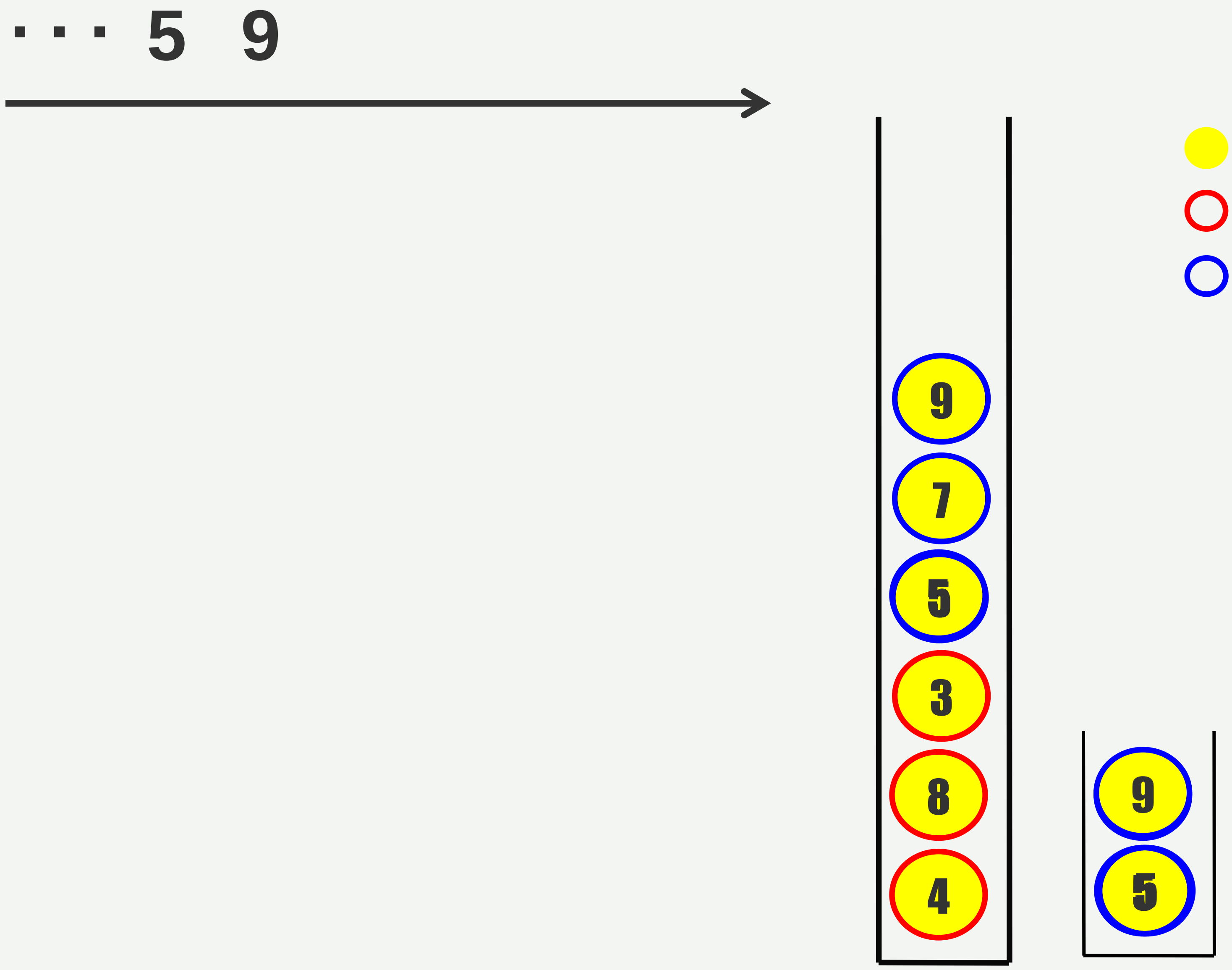
Cache size

L = 5

L_{lir} = 3

L_{hir} =2

Access a Non-Resident HIR block (a Miss)



resident in cache

LIR block

HIR block

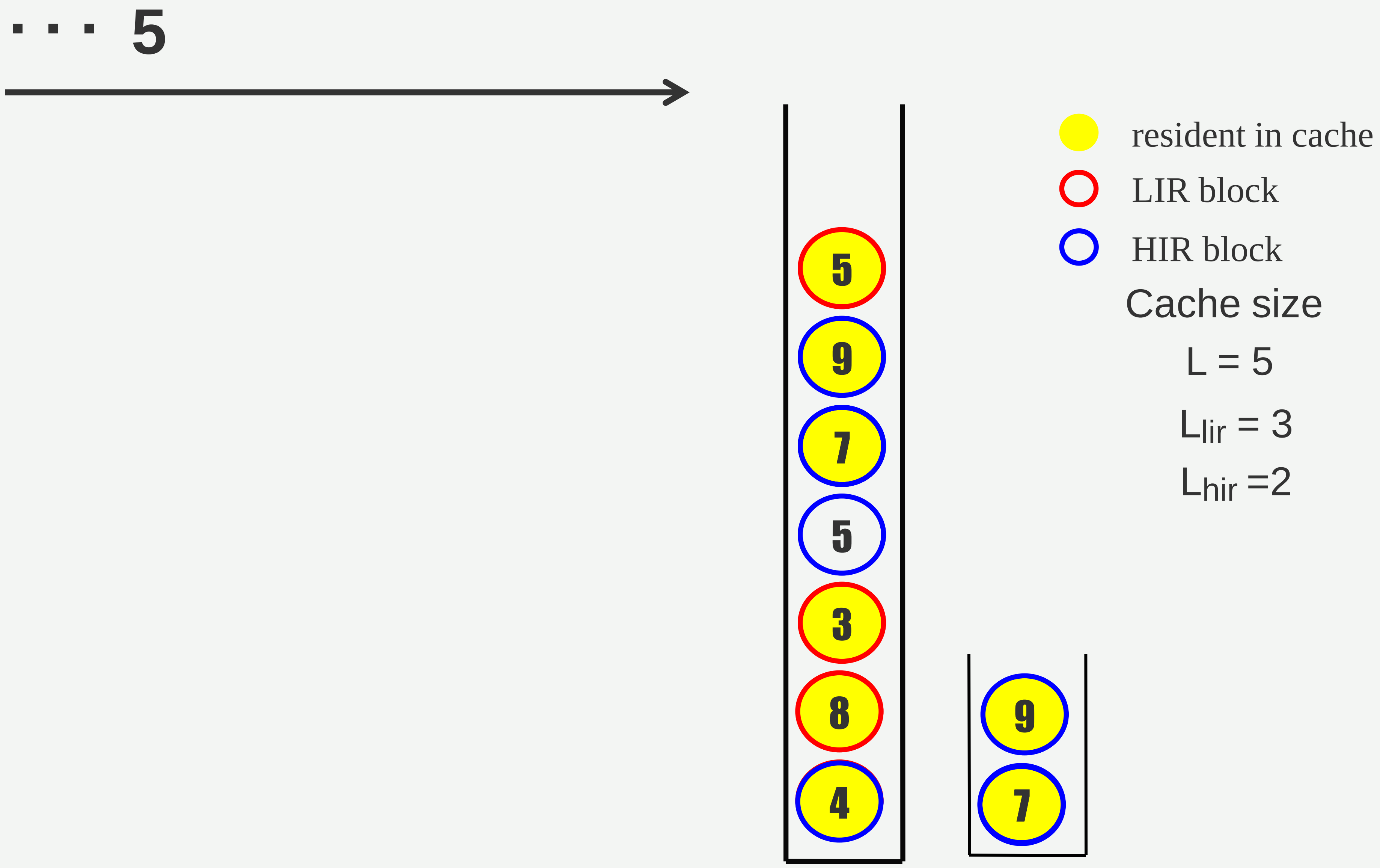
Cache size

L = 5

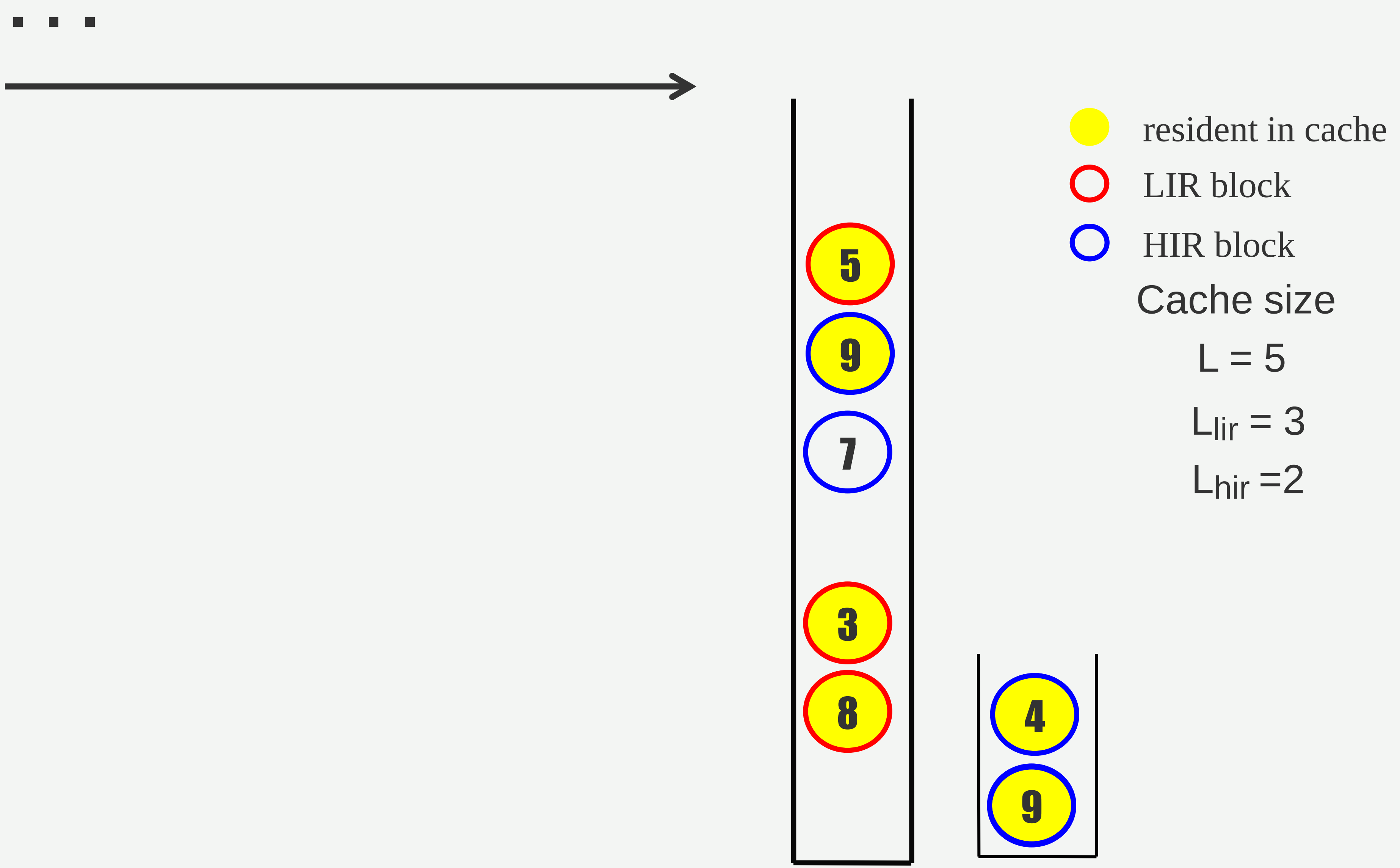
L_{lir} = 3

L_{hir} =2

Access a Non-Resident HIR block (a Miss)



Access a Non-Resident HIR block (a Miss)



LIRS Stack Simplifies Replacement

- Recency is ordered in stack with *Rmax* LIR block in bottom
- No need to keep track of each HIR block`s IRR because
  - A newly accessed HIR block`s IRRs in stack = recency < R_{max}.
- A small LRU stack is used to store resident HIR blocks.
- Additional operations of pruning and demoting are constant.
- Although LIRS operations are much more dynamic than LRU, its complexity is identical to LRU.

Performance Evaluation

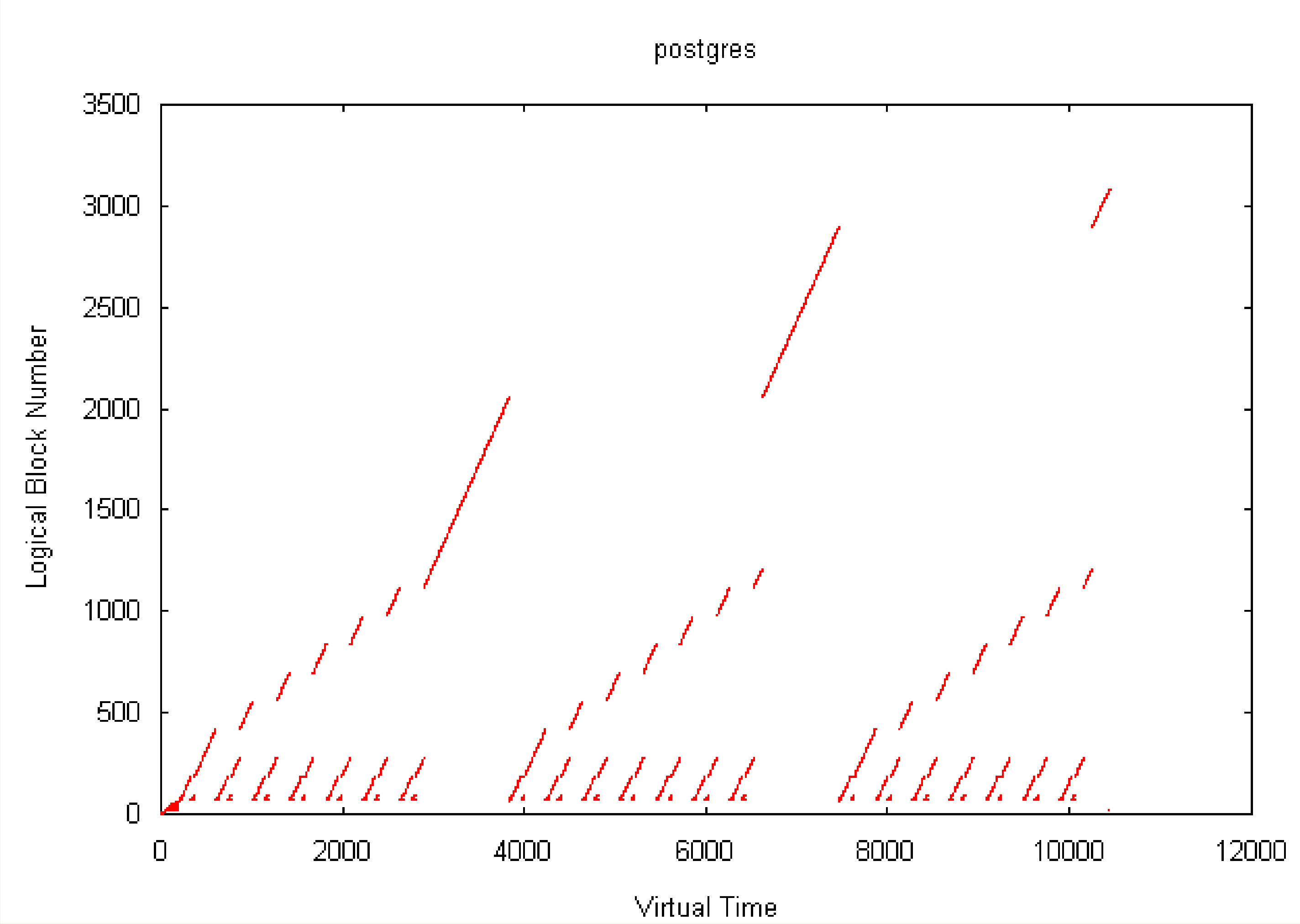
- Trace-driven simulation on different patterns shows
  - LIRS outperforms existing replacement algorithms in almost all the cases.
  - The performance of LIRS is not sensitive to its only parameter L_{hirs}.
  - Performance is not affected even when LIRS stack size is bounded.
  - The time/space overhead is as low as LRU.
  - LRU can be regarded as a special case of LIRS.

Selected Workload Traces

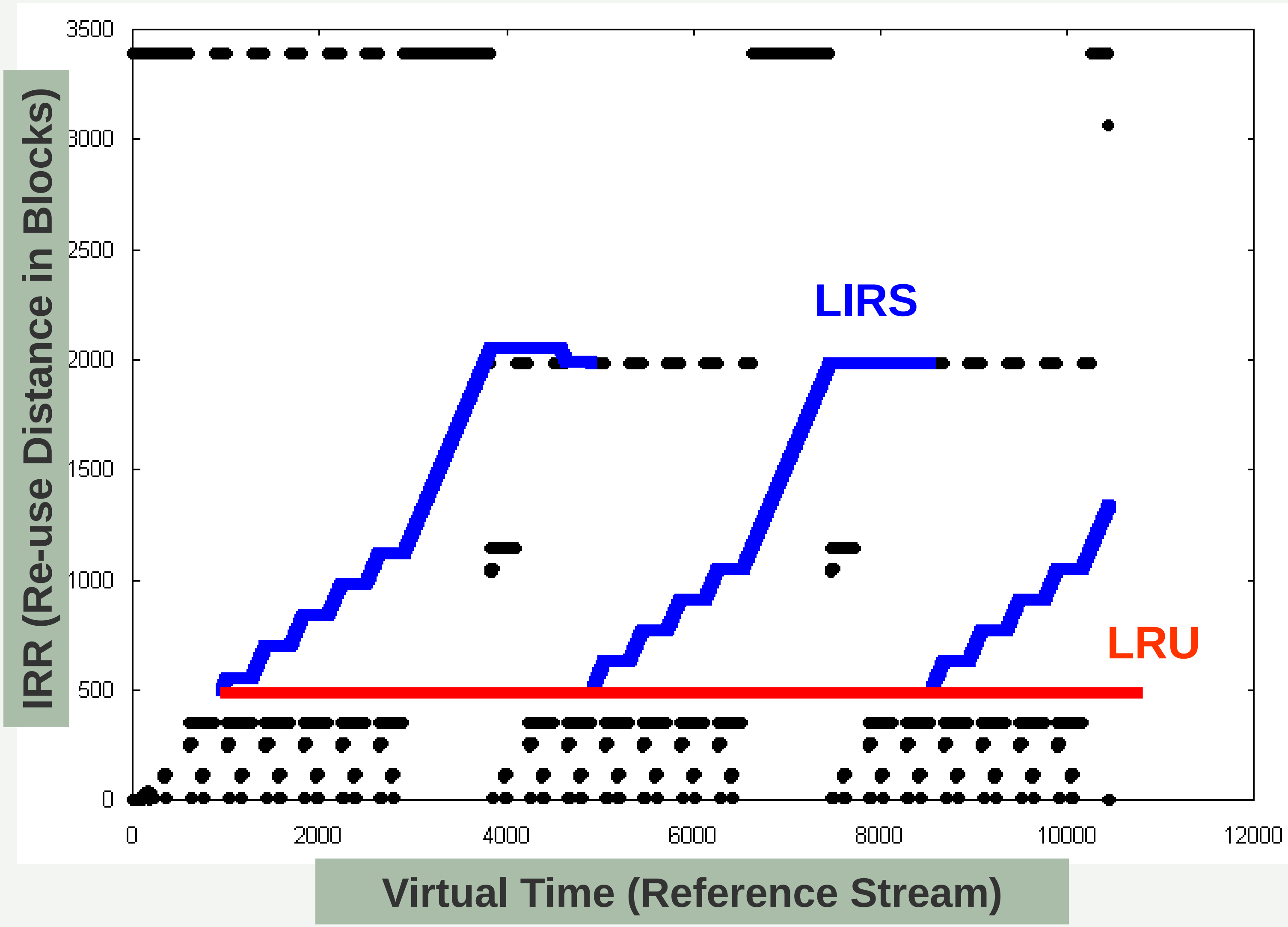
- **2-pools** is a synthetic trace to simulate the distinct frequency case.
- **cpp** is a GNU C compiler pre-processor trace
- **cs** is an interactive C source program examination tool trace.
- **glimpse** is a text information retrieval utility trace.
- **link** is a UNIX link-editor trace.
- **postgres** is a trace of join queries among four relations in a relational database system
- **sprite** is from the Sprite network file system
- **mulit1**: by executing 2 workloads, cs and cpp, together.
- **multi2:** by executing 3 workloads, cs, cpp, and postgres, together.
- **multi3:** by executing 4 workloads, cpp, gnuplot, glimpse, and postgres, together

(1) various patterns, (2) non-regular accesses , (3) large traces.

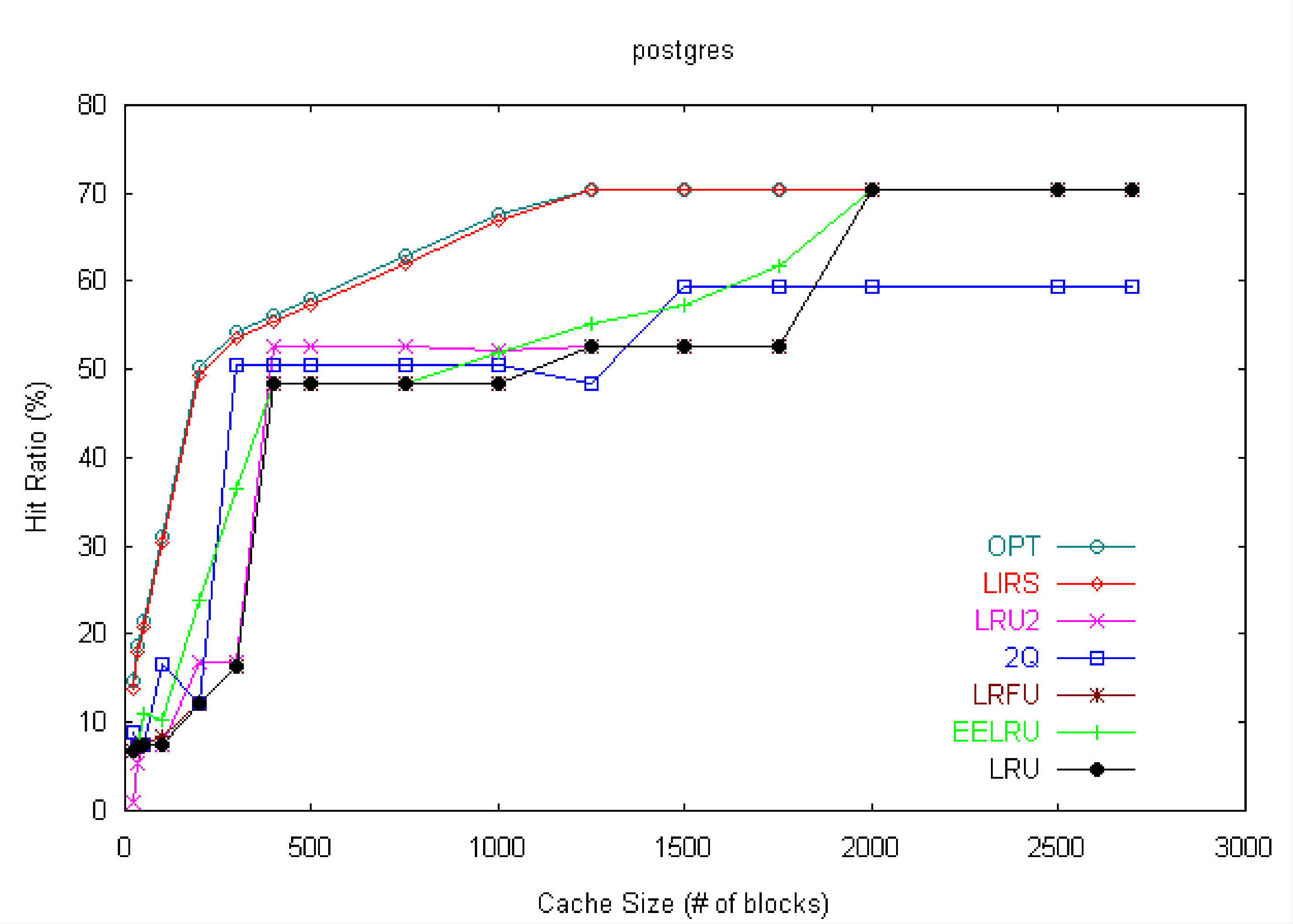
Looping Pattern: postgres (Time-space map)



Looping Pattern: postgres (IRR Map)



Looping Pattern: postgres (Hit Rates)



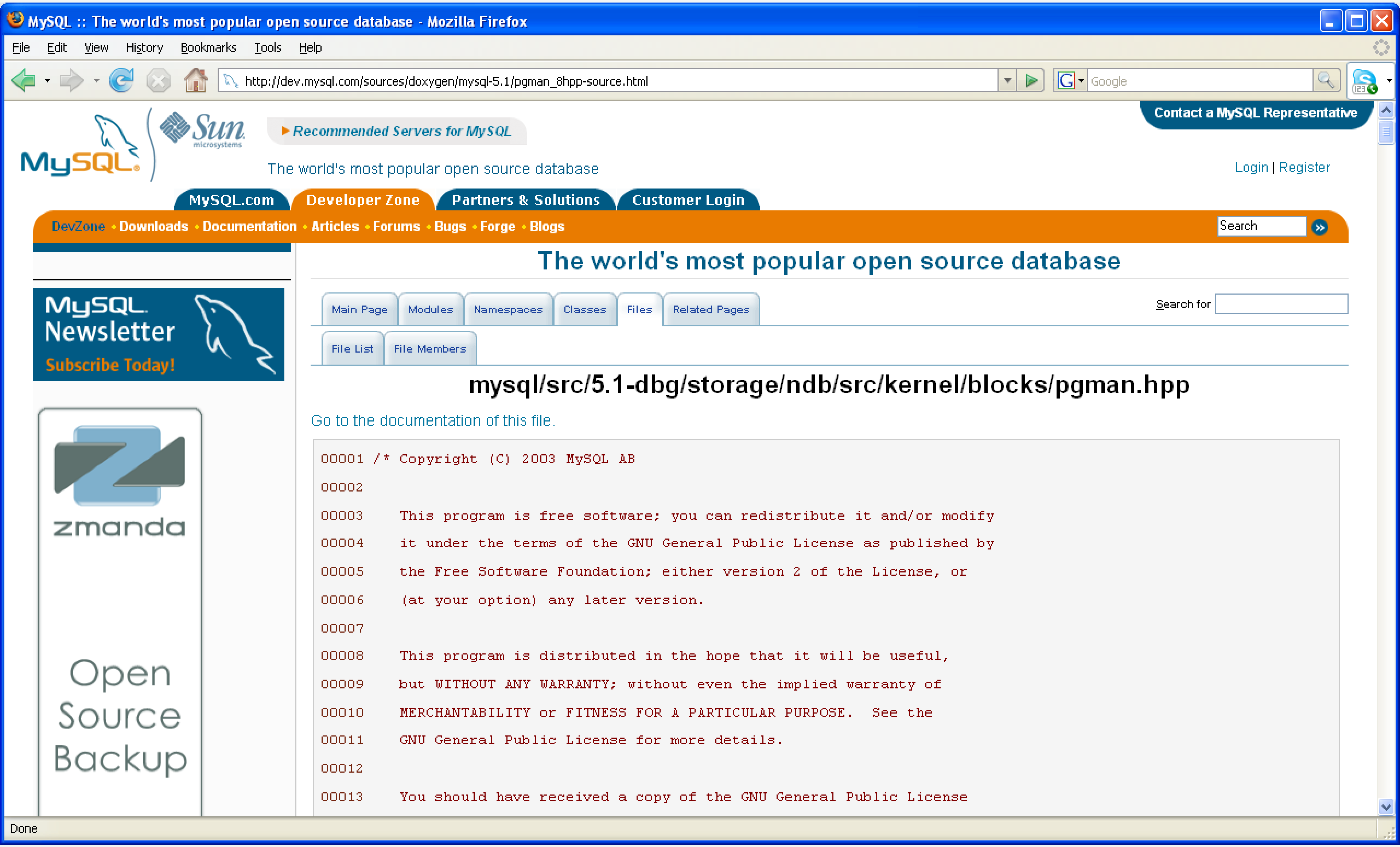
Impact of LIRS

- LIRS is a benchmark to compare replacement algorithms
  - **Reuse distance is first used in buffer management**
  - **A paper in SIGMETRICS'05 confirmed that LIRS outperforms all the other replacement.**
  - **LIRS has become a topic to teach in both graduate and undergraduate classes of OS, performance evaluation, and databases at many US universities.**
  - **A high number of citations to the LIRS paper.**
- Linux Memory Management group has established an Internet Forum on Advanced Replacement, including LIRS

LIRS has been adopted in MySQL

- MySQL is the most widely used relational database
  - 11 million installations in the world
  - The busiest Internet services use MySQL to maintain their databases for high volume Web sites: **google, YouTube, wikipedia, facebook, …**
  - LIRS is managing the **buffer pool** of MySQL
  - The adoption is the most recent version **(5.1),** November 2008.
- LIRS is documented as **Jiang-Zhang caching algorithm** in MySQL.

LIRS-MySQL-jiang-zhang.mht


MySQL :: The world's most popular open source database - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dev.mysql.com/sources/doxygen/mysql-5.1/pgman_8hpp-source.html
Google
Recommended Servers for MySQL
Contact a MySQL Representative
The world's most popular open source database
Login | Register
MySQL.com
Developer Zone
Partners & Solutions
Customer Login
DevZone • Downloads • Documentation • Articles • Forums • Bugs • Forge • Blogs
Search
The world's most popular open source database
MySQL Newsletter
Subscribe Today!
Main Page
Modules
Namespaces
Classes
Files
Related Pages
Search for
File List
File Members
mysql/src/5.1-dbg/storage/ndb/src/kernel/blocks/pgman.hpp
Go to the documentation of this file.
00001 /* Copyright (C) 2003 MySQL AB
00002
00003 This program is free software; you can redistribute it and/or modify
00004 it under the terms of the GNU General Public License as published by
00005 the Free Software Foundation; either version 2 of the License, or
00006 (at your option) any later version.
00007
00008 This program is distributed in the hope that it will be useful,
00009 but WITHOUT ANY WARRANTY; without even the implied warranty of
00010 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00011 GNU General Public License for more details.
00012
00013 You should have received a copy of the GNU General Public License
zmanda
Open Source Backup
Done

MySQL :: The world's most popular open source database - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://dev.mysql.com/sources/doxygen/mysql-5.1/pgman_8hpp-source.html

Google

```
 * A local check point (LCP) periodically performs a complete pageout of
 * dirty pages.  It must iterate over a list which will cover all pages
 * which had been dirty since LCP start.
 *
 * A clean page is a candidate ("victim") for being "unmapped" and
 * "evicted" from the cache, to allow another page to become resident.
 * This process is called "page replacement".
 *
 * PAGE REPLACEMENT
 *
 * Page replacement uses the LIRS algorithm (Jiang-Zhang).
 *
 * The "recency" of a page is the time between now and the last request
 * for the page.  The "inter-reference recency" (IRR) of a page is the
 * time between the last 2 requests for the page.  "Time" is advanced by
 * request for any page.
 *
 * Page entries are divided into "hot" ("lir") and "cold" ("hir").  Here
 * lir/hir refers to low/high IRR.  Hot pages are always resident but
 * cold pages need not be.
 *
 * Number of hot pages is limited to slightly less than number of cache
 * pages.  Until this number is reached, all used cache pages are hot.
 * Then the algorithm described next is applied.  The algorithm avoids
 * storing any of the actual recency values.
```

Done

LIRS is adopted in Java Library

- LIRS has been adopted in Infinispan, a Java-based data grid
- LIRS is being adopted in Java Class of
  - ConcurrentLinkedHashMap
  - as a software cache management facility