SDC

STORAGE DEVELOPER CONFERENCE

SNIA ■ SANTA CLARA, 2016

Multi-Chance Adaptive Replacement Cache(MC-ARC)

Shailendra Tripathi

Architect,

Tegile Systems

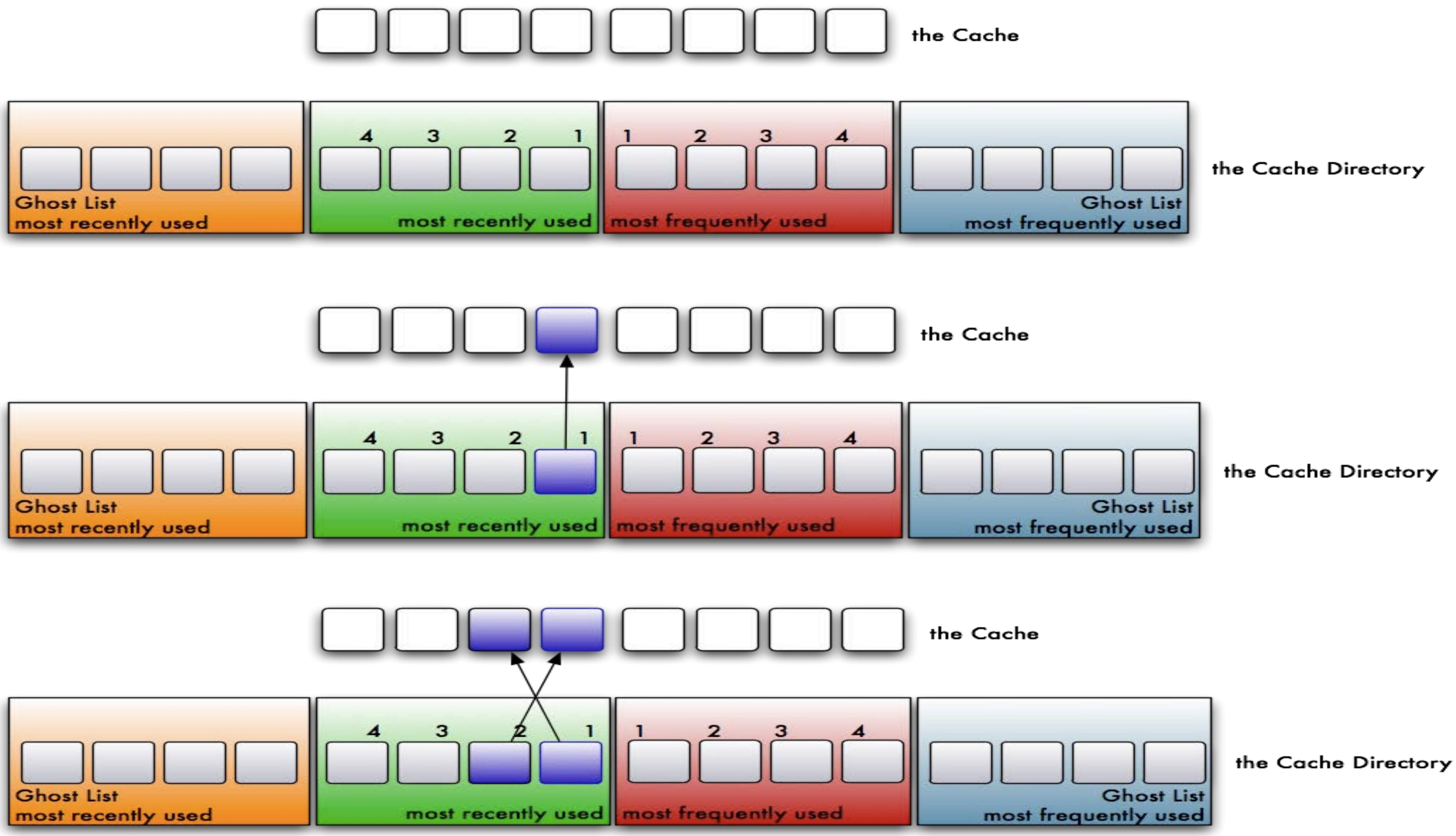
Outline

- Adaptive Replacement Cache (ARC)
- Scaling Issue of ARC
- Design Goals
- MC-ARC
- Test Results
- References
- Q & A

Adaptive Replacement Cache (ARC)

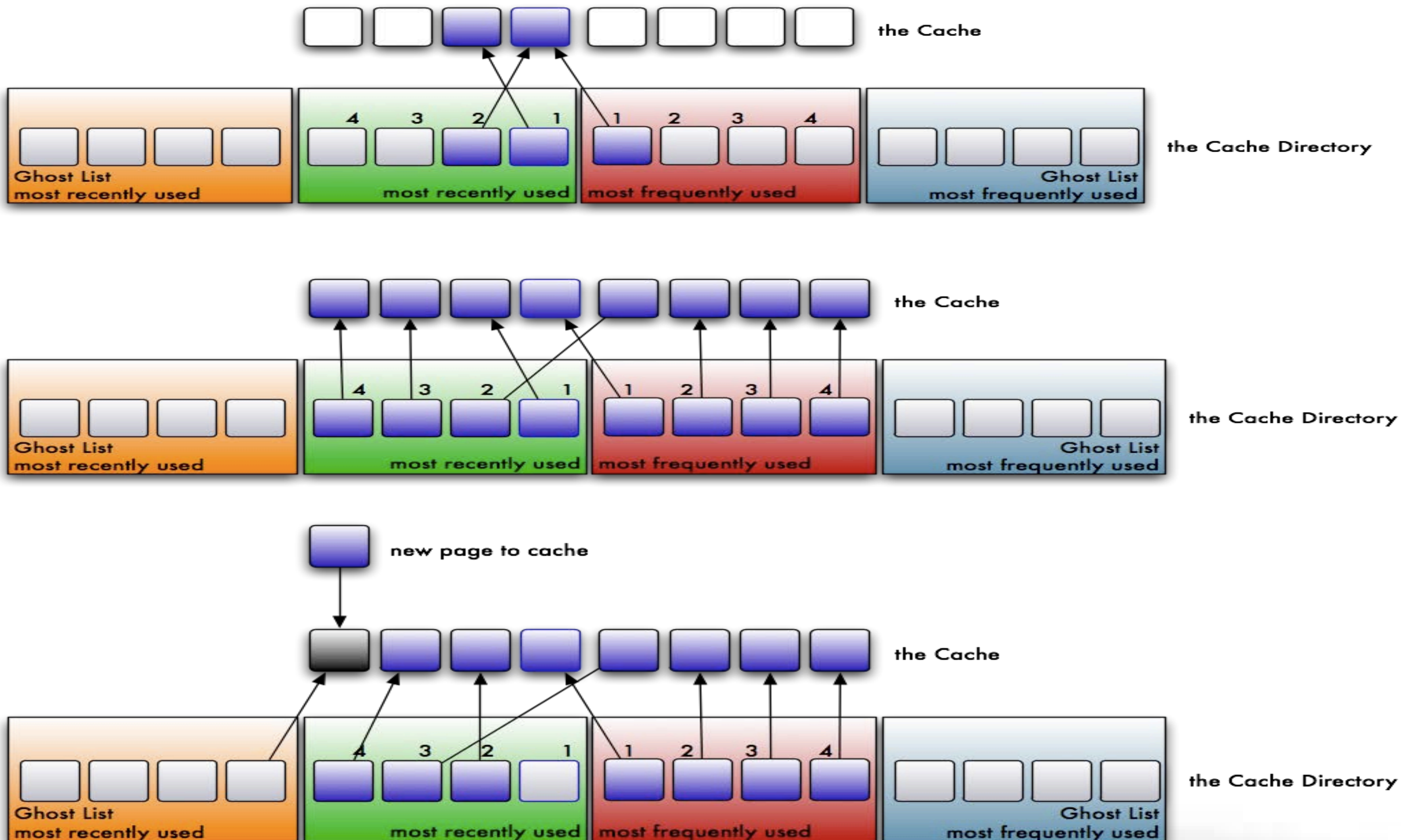
- Two Separate Lists – MRU, MFU
- Two Ghosts Lists - Ghost MRU, Ghost MFU
- Track recently evicted pages, has no data
- Top Level Separation for Metadata and Data
- Referenced Pages are tracked in ANON list
- MRU + MFU = GMRU + GMFU = ARC size
- List size dynamically adaptive based on hits on the respective ghost lists.

ARC

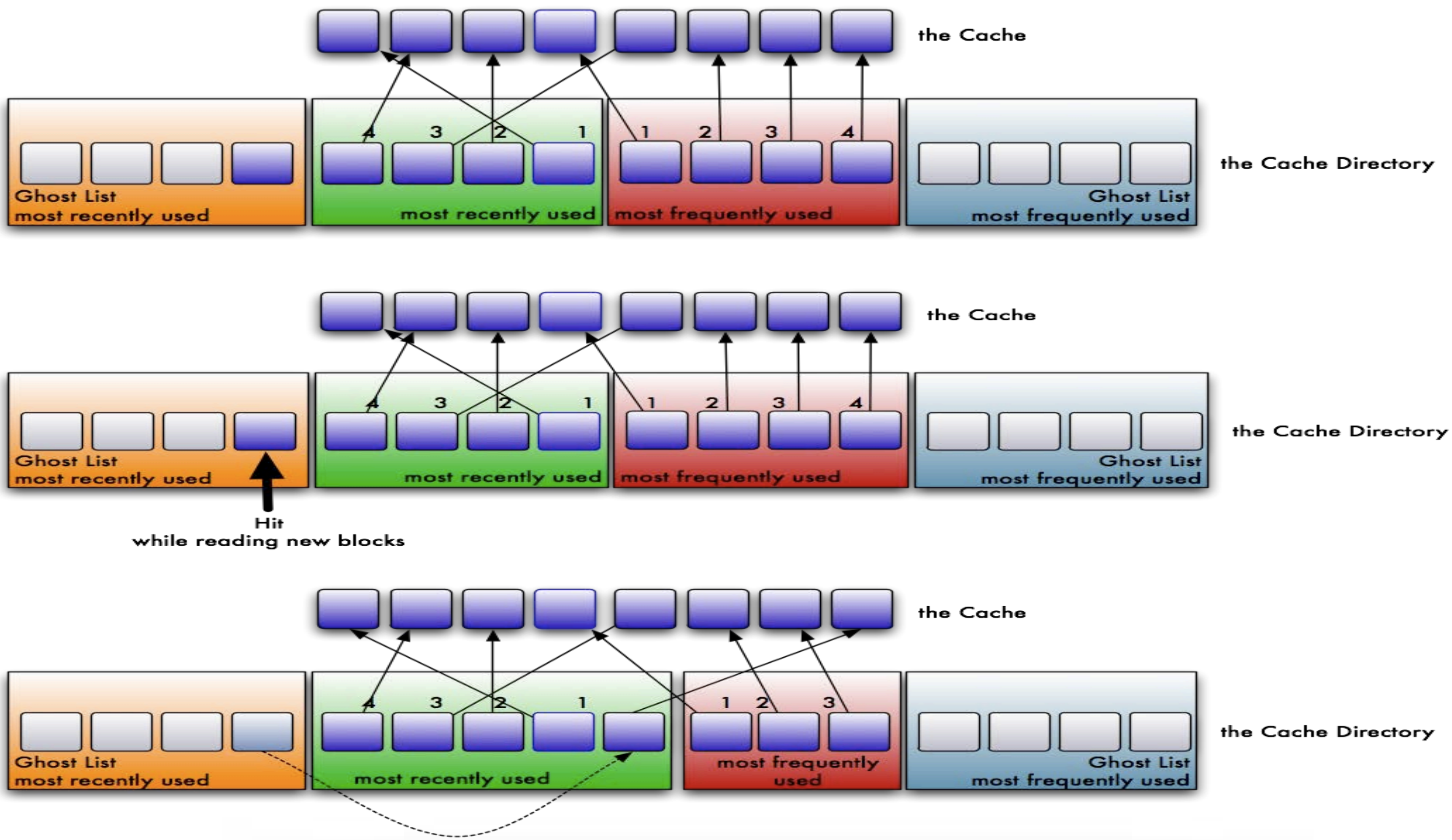


Images sourced from the blog at : http://www.c0t0d0s0.org/archives/5329-Some-insight-into-the-read-cache-of-ZFS-or-The-ARC.html

ARC Mechanism



ARC Mechanism Continued



ARC Performance – Scale View

Lockstat Output on 4K 100 % random read – from 8 Windows client running IOMETER/FIO

Count	indv	cuml	rcnt	nsec	Lock	Caller
…						
120476	**4%**	**27%**	**0.00**	**181191**	**ARC_mru+0x58**	**remove_reference+0x63**
…						
61496	**2%**	**31%**	**0.00**	**3662**	**ARC_mru+0x58**	**arc_evict+0x54d**
50349	**2%**	**36%**	**0.00**	**214993**	**ARC_mru+0x58**	**add_reference+0x7e**

Lockstat	Tool to capture locking contention on Solaris/OS platform.
Test Platform	Test was run on Tegile's Entry Level All flash product with 96GB RAM & Intel(r) Xeon(r) CPU E5-2450 v2 @ 2.50GHz
Test Steps	Create 100 GB LUN from each client (total 800GB data)
	100 % random prep read for 30 minutes
	Measure Results - 10 minutes 100% random read run

Scaling Issue of ARC

- LRU Lists - global Lock per list
- Each Access / Drop – Linked List movement
- Storage Service times < 75 us(SSD) and going down(NVMe/3D).
- Increasing CPU Cores & Threads
- Allocation increasingly higher priority than Caching value
- Impacts both Read and Write IOs

Million IOPS per box goal

- Read service time - end-to-end <=62.5 us
- SSD service times similar order
- Pipeline processing & SSD servicing
- Processing <= SSD service times all the time
- Constant overhead – interrupt (at least 2, one context switch, and, copy out) ~10us

Locking & Alloc/Eviction – Single digit overhead

Ideal Page Cache / Design Goals

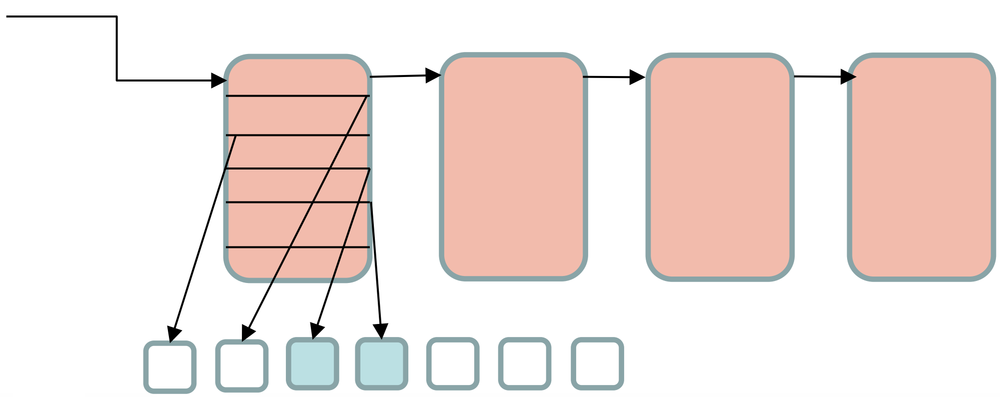
1. Minimizes big lock usage
2. Minimize movements (within or across)
3. Short term - Long term value differentiation
4. Scan resistant

ARC – good in 3 and 4

Items 1 and 2 are scaling bottleneck

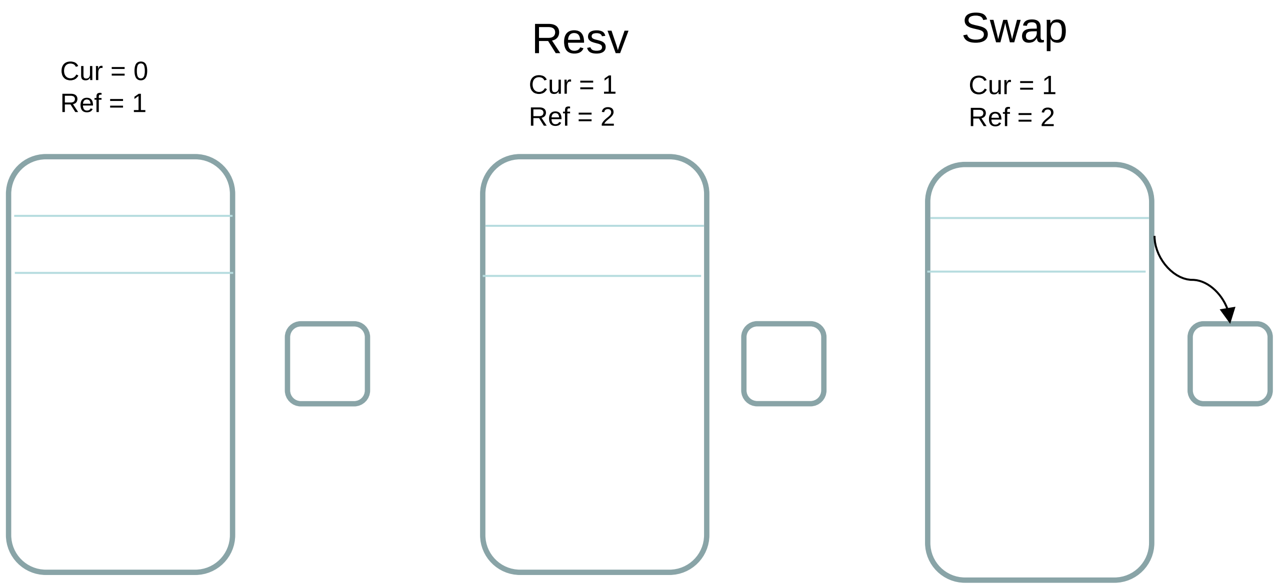
Scalable MC-ARC – Lockless Page Cache

- Linked lists are replaced with page arrays
- Configurable Page size
- Page array points to cache buffers
- Pages are linked in FIFO order
- Pages are dynamically linked for flexibility



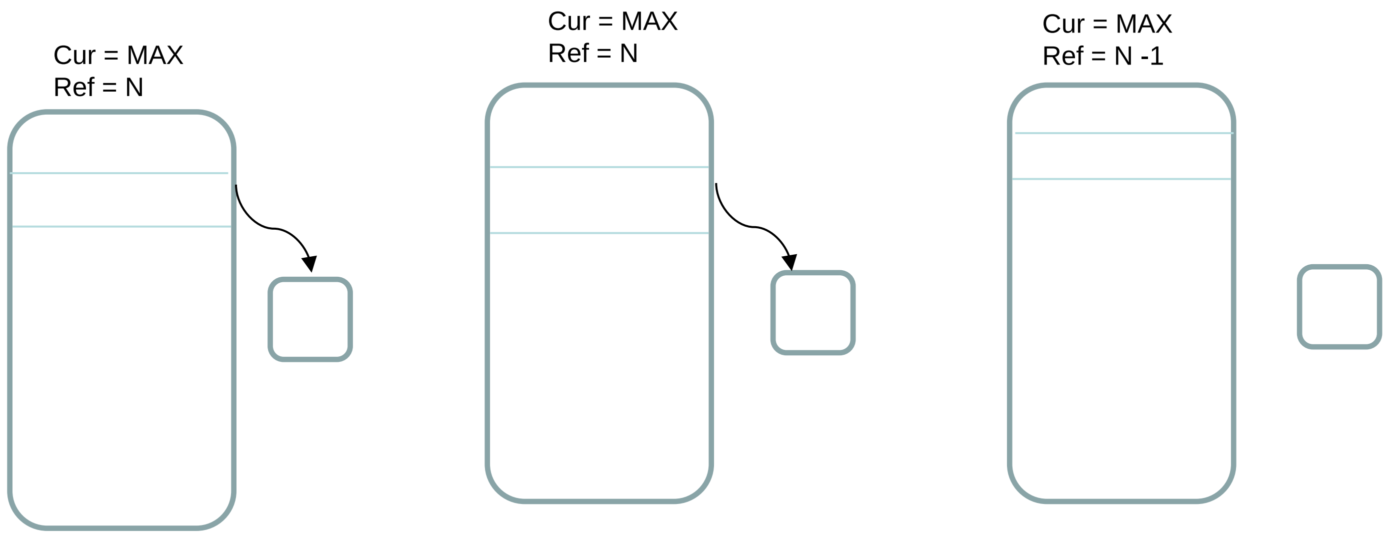
MC-ARC - Insertion

- Reserve Swap – Atomic Add
- Swap Pointer – Atomic Swap



MC-ARC – Remove

- Swap NULL – Atomic Swap
- Dec Ref Count – Atomic Dec



MC-ARC - Access

- Buf hash table lookup
- Remove from the old Page
- Re-insert in the page at the head
- Protected by buffer lock

Eviction

- Take the page out from tail
- Check if array pointer is valid
- Try buf hash lock – Locked and eligible, free buf
- Try lock fails, add page to a private list
- At the end of the eviction span, re-insert the private list at tail

Test Results

- Scaled IOPS from 275K to 1 Million IOPS
- Lockstat Contention – No ARC function in top 50
- Higher block sizes – 4x10GB ports are saturated at 100% Random Reads
- OS memory management – next level bottleneck
- CPU burns out at small block sizes(4k/8k)

References

- ARC: A Self-Tuning, Low Overhead Replacement Cache“ - by N. Megiddo & D. Modha
- http://www.cs.cmu.edu/~15-440/READINGS/megiddo-computer2004.pdf
- http://users.cis.fiu.edu/~raju/WWW/publications/hotstorage2015/paper.pdf
- http://www.tegile.com/products/all-flash-array/
- https://pthree.org/2012/12/07/zfs-administration-part-iv-the-adjustable-replacement-cache
- http://www.c0t0d0s0.org/archives/5329-Some-insight-into-the-read-cache-of-ZFS-or-The-ARC.html

SDC 16