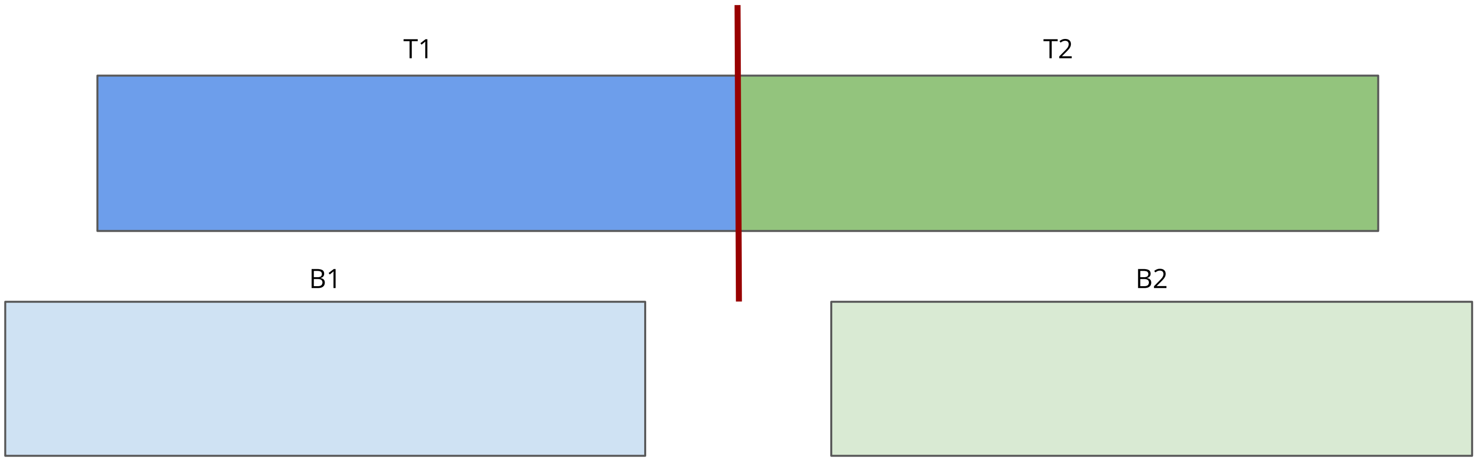


ZFS Adaptive Replacement Cache

iXsystems

Caching Overview

- System Memory (RAM) is much MUCH faster than hard disks, but much MUCH smaller
- Idea of a cache: predict what the user will need before they request it, *cache* that data in system memory to improve access speed
- Prediction accuracy called “cache hit rate”, i.e., how often is the data we’re requesting in the cache?
- Even if cache hit rate is 10%, overall performance improvement still will be significant
- Example of a simple prediction algorithm - Least Recently Used (LRU):
  - When user accesses data, it will be put in the cache in anticipation of future re-use
  - When cache fills up, evict the oldest item (or the “least recently used”)
  - Very simple algorithm, but performance is just okay
- Adaptive Replacement Cache (ARC) improves on this by adding a second list to track frequently-used data

iXsystems

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec

iXsystems

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec

iXsystems

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec
Intel Optane SSD	6 to 15 hours	10 to 15 μSec

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec
Intel Optane SSD	6 to 15 hours	10 to 15 µSec
NVMe SSD	3 to 11 days	100 to 200 µSec

iXsystems™

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec
Intel Optane SSD	6 to 15 hours	10 to 15 μSec
NVMe SSD	3 to 11 days	100 to 200 μSec
SAS/SATA SSD	69 to 105 days	2 to 3 mSec

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec
Intel Optane SSD	6 to 15 hours	10 to 15 μSec
NVMe SSD	3 to 11 days	100 to 200 μSec
SAS/SATA SSD	69 to 105 days	2 to 3 mSec
15K RPM HDD	105 to 210 days	3 to 6 mSec

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec
Intel Optane SSD	6 to 15 hours	10 to 15 μSec
NVMe SSD	3 to 11 days	100 to 200 μSec
SAS/SATA SSD	69 to 105 days	2 to 3 mSec
15K RPM HDD	105 to 210 days	3 to 6 mSec
10K RPM HDD	243 to 315 days	8 to 9 mSec

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec
Intel Optane SSD	6 to 15 hours	10 to 15 µSec
NVMe SSD	3 to 11 days	100 to 200 µSec
SAS/SATA SSD	69 to 105 days	2 to 3 mSec
15K RPM HDD	105 to 210 days	3 to 6 mSec
10K RPM HDD	243 to 315 days	8 to 9 mSec
7.2K RPM HDD	315 to 525 days	10 to 15 mSec

iXsystems

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	2 to 4 minutes	50 to 70 nSec
Intel Optane SSD	6 to 15 hours	10 to 15 μSec
NVMe SSD	3 to 11 days	100 to 200 μSec
SAS/SATA SSD	69 to 105 days	2 to 3 mSec
15K RPM HDD	105 to 210 days	3 to 6 mSec
10K RPM HDD	243 to 315 days	8 to 9 mSec
7.2K RPM HDD	315 to 525 days	10 to 15 mSec
5.4K RPM HDD	525 to 700 days	15 to 20 mSec

Why Cache? How slow are disks REALLY?

- Modern CPUs usually run at ~3GHz, meaning they can perform 3,000,000,000 (3 billion) instructions every second.
- If we stretched out time so that our CPU did just one instruction every second, this is how long we would have to wait to access various storage medias (from time of request to time that response begins):

Storage Type	Slowed Time Scale	Real Time Scale
Single CPU Instruction (at 3 GHz)	1 second	0.3 nSec
Registers (storage for active instructions)	1 to 3 seconds	0.3 to 1 nSec
Level 1 Cache (on-CPU cache)	2 to 8 seconds	0.7 to 3 nSec
Level 2 Cache (off-CPU but still on chip)	5 to 12 seconds	2 to 4 nSec
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec
Intel Optane SSD	6 to 15 hours	10 to 15 µSec
NVMe SSD	3 to 11 days	100 to 200 µSec
SAS/SATA SSD	69 to 105 days	2 to 3 mSec
15K RPM HDD	105 to 210 days	3 to 6 mSec
10K RPM HDD	243 to 315 days	8 to 9 mSec
7.2K RPM HDD	315 to 525 days	10 to 15 mSec
5.4K RPM HDD	525 to 700 days	15 to 20 mSec
3.5" Floppy Disk	*23.75 years!!*	250 mSec

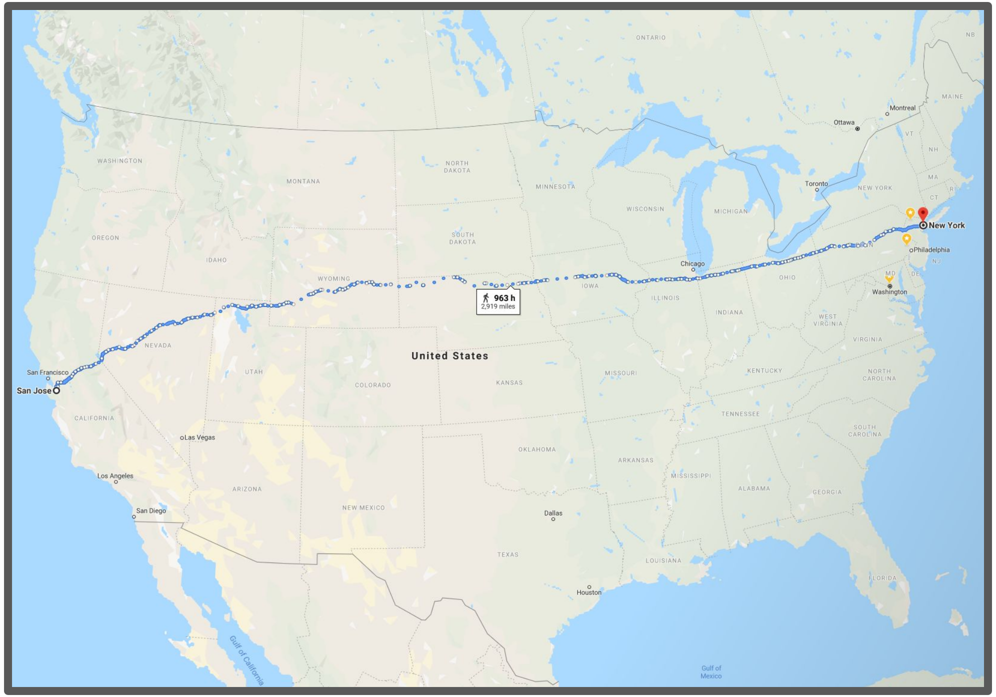
iXsystems™

Why Cache? How slow are disks REALLY?

Let's imagine we want a slice of pizza:

- Accessing the data from RAM would be like having the pizza in our fridge. We get up and microwave it, it’s ready in **about a minute**.
- Accessing data from the hard disks is like **walking from San Jose to New York**, buying a slice of pizza, then **walking all the way back to San Jose** before we eat it (round trip of ~1.3 years)!

Storage Type	Slowed Time Scale	Real Time Scale
RAM	30 to 60 seconds	10 to 20 nSec
7.2K RPM HDD	315 to 525 days	10 to 15 mSec

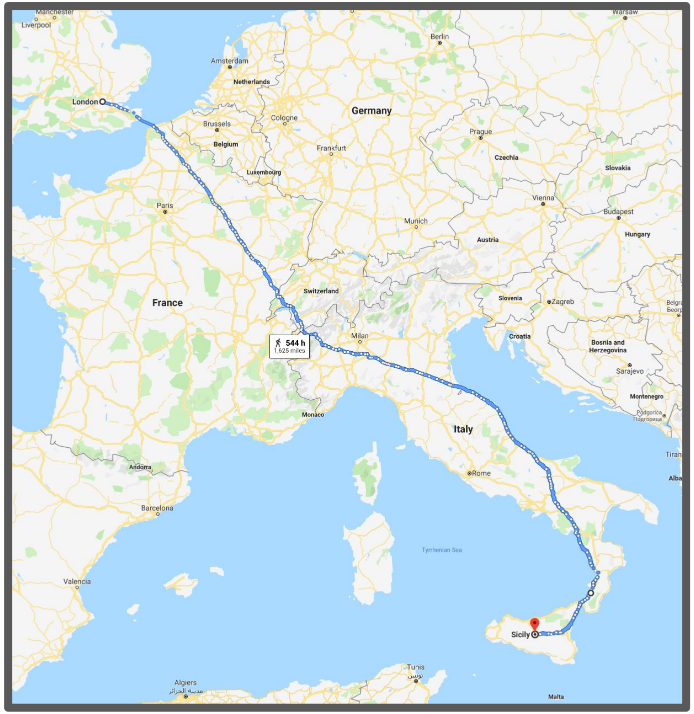


Why Cache? How slow are disks REALLY?

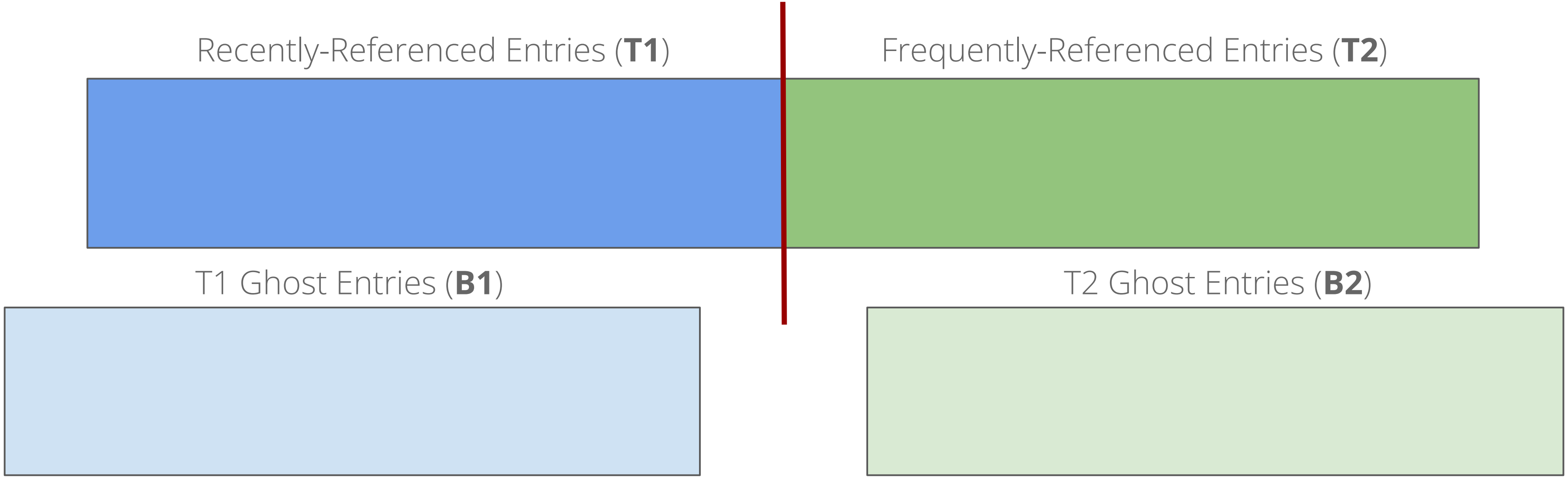
Let's imagine we want a slice of pizza (but in Europe now):

- Accessing the data from RAM would be like having the pizza in our fridge. We get up and microwave it, it's ready in **about a minute**.
- Accessing data from the hard disks is like **walking from London to Sicily**, buying a slice of pizza, then **walking all the way back to London** before we eat it (round trip of ~1 year at 10 miles per day)!

Storage Type	Slowed Time Scale	Real Time Scale
Main System Memory (RAM)	30 to 60 seconds	10 to 20 nSec
7.2K RPM HDD	315 to 525 days	10 to 15 mSec



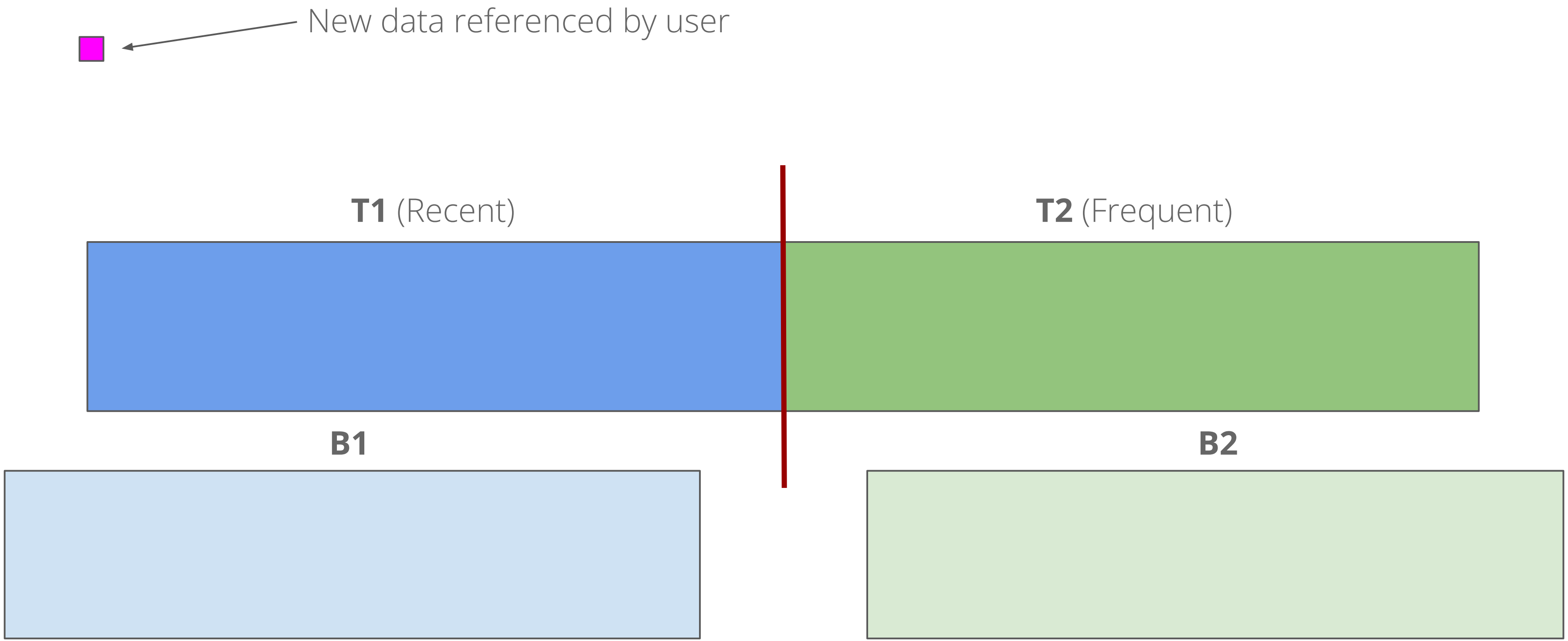
Adaptive Replacement Cache (ARC)



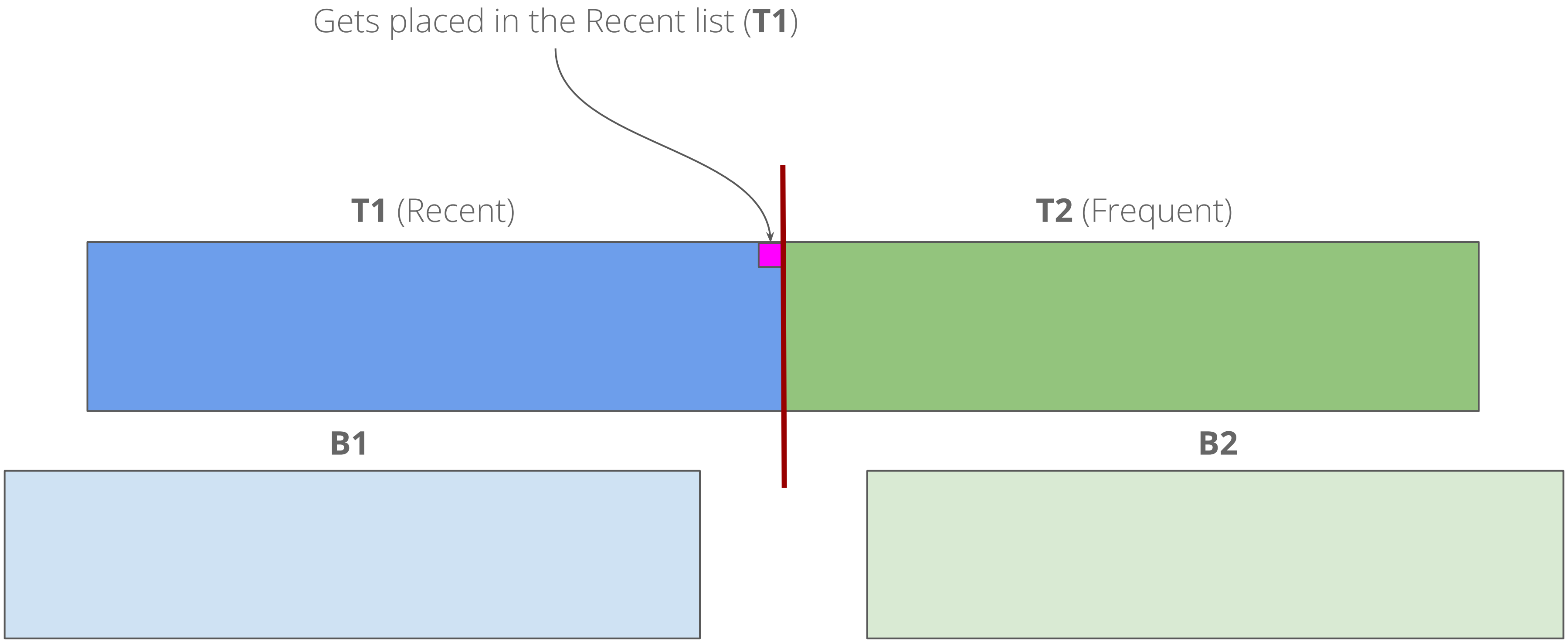
Cache split into two parts: **T1** for recently-used items and **T2** for frequently-used items (used at least two times)

Items evicted from **T1**, **T2** are tracked in ghost list **B1**, **B2**. These lists keeps track of what was evicted without storing the actual data was (just stores reference to evicted data, not the data itself)

Adaptive Replacement Cache (ARC)

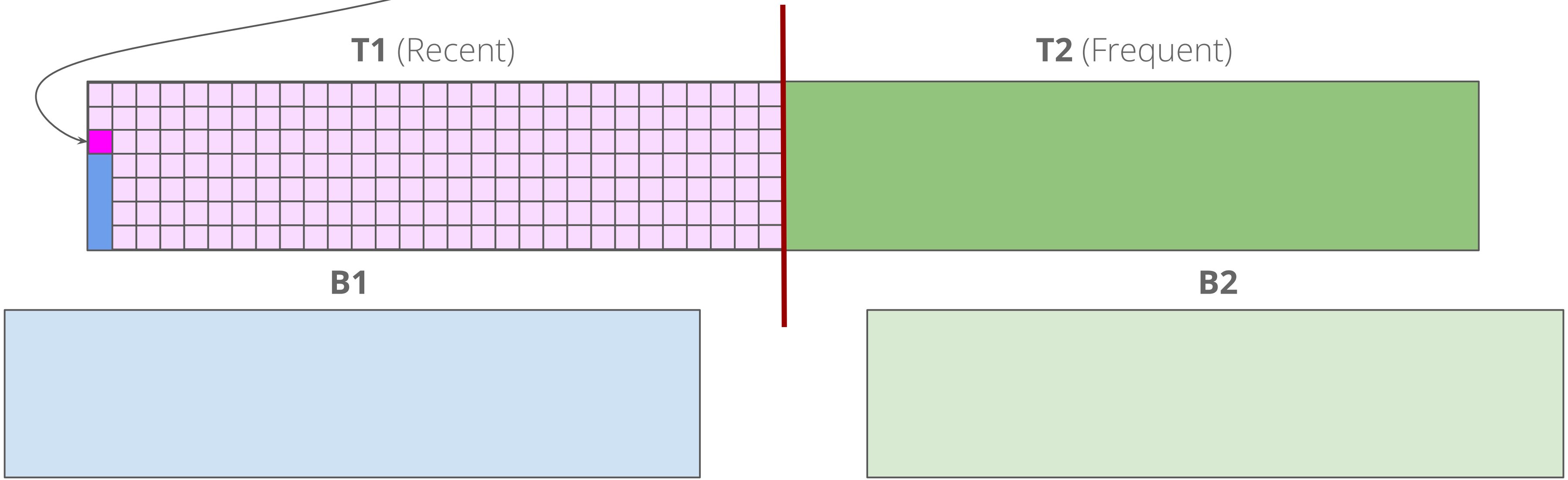


Adaptive Replacement Cache (ARC)



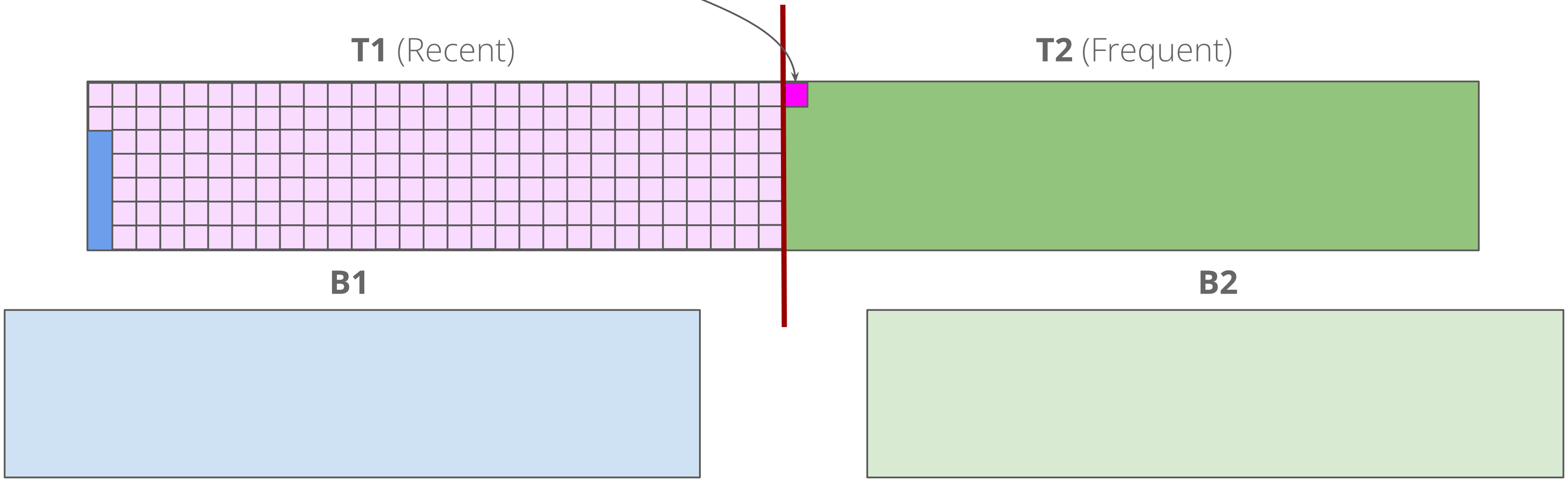
Adaptive Replacement Cache (ARC)

As more new data comes in, the block
gets pushed down the list



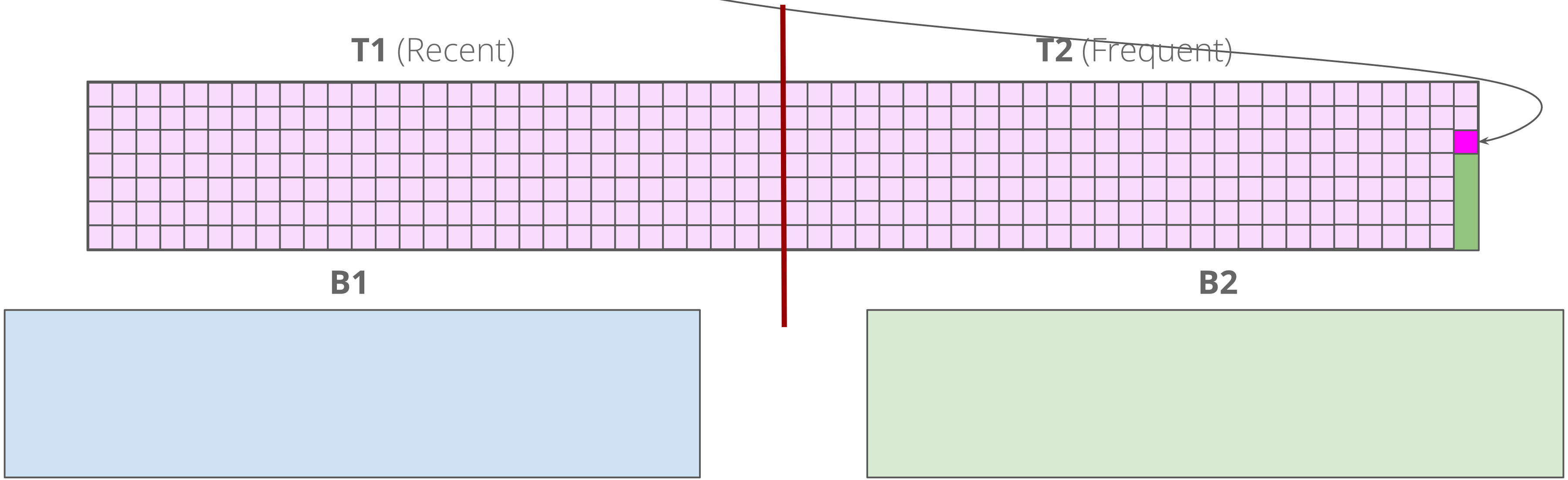
Adaptive Replacement Cache (ARC)

If it gets requested by the user a second time, it gets moved into **T2**



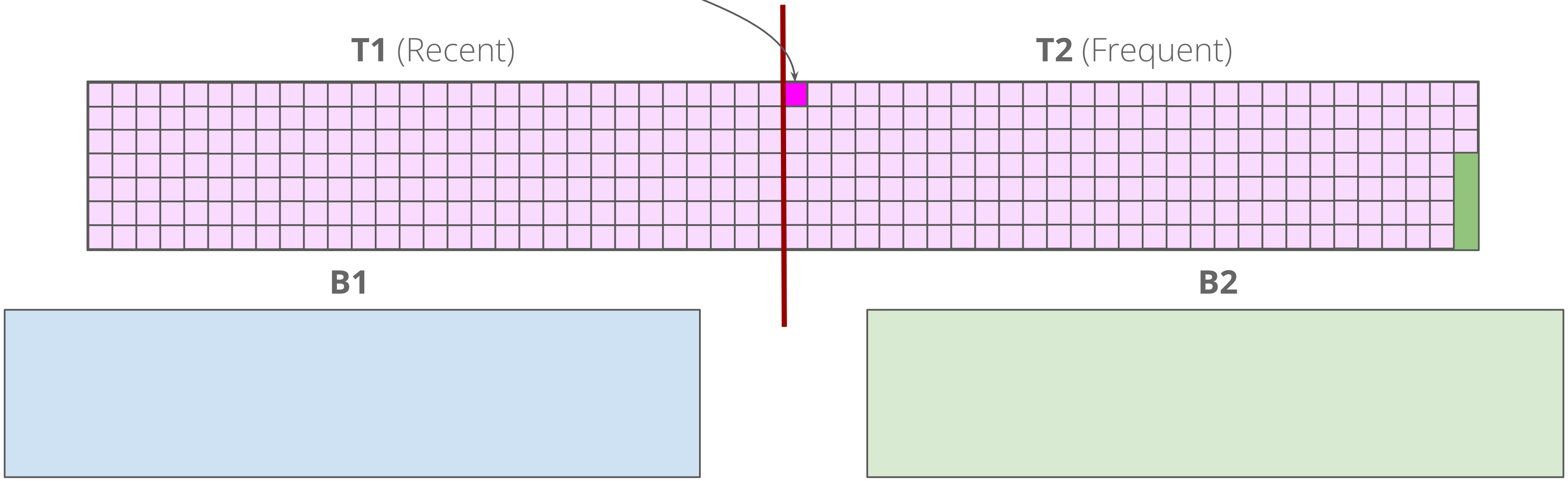
Adaptive Replacement Cache (ARC)

T2 fills up with frequently-used blocks,
our block gets pushed down the list
(**T1** usually fills up, too)

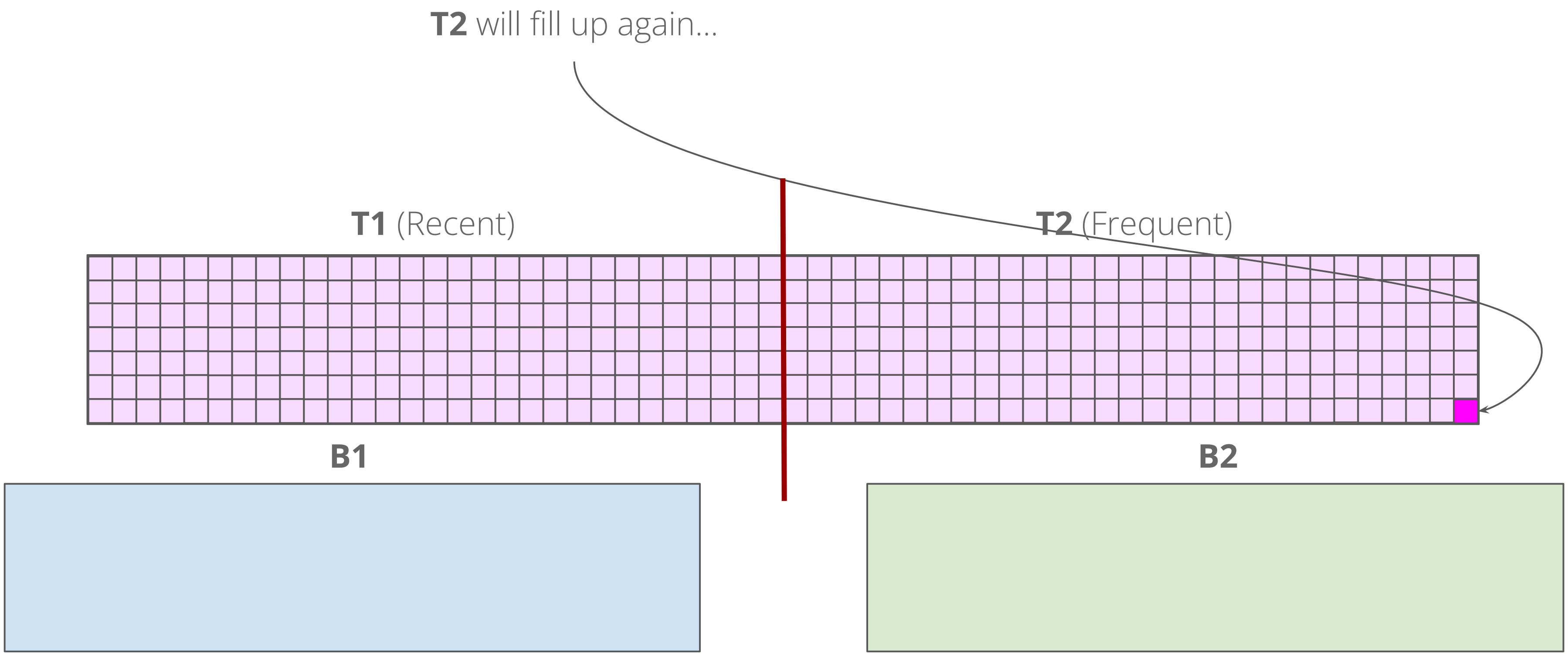


Adaptive Replacement Cache (ARC)

If our block gets accessed again, it jumps to front of **T2** list again

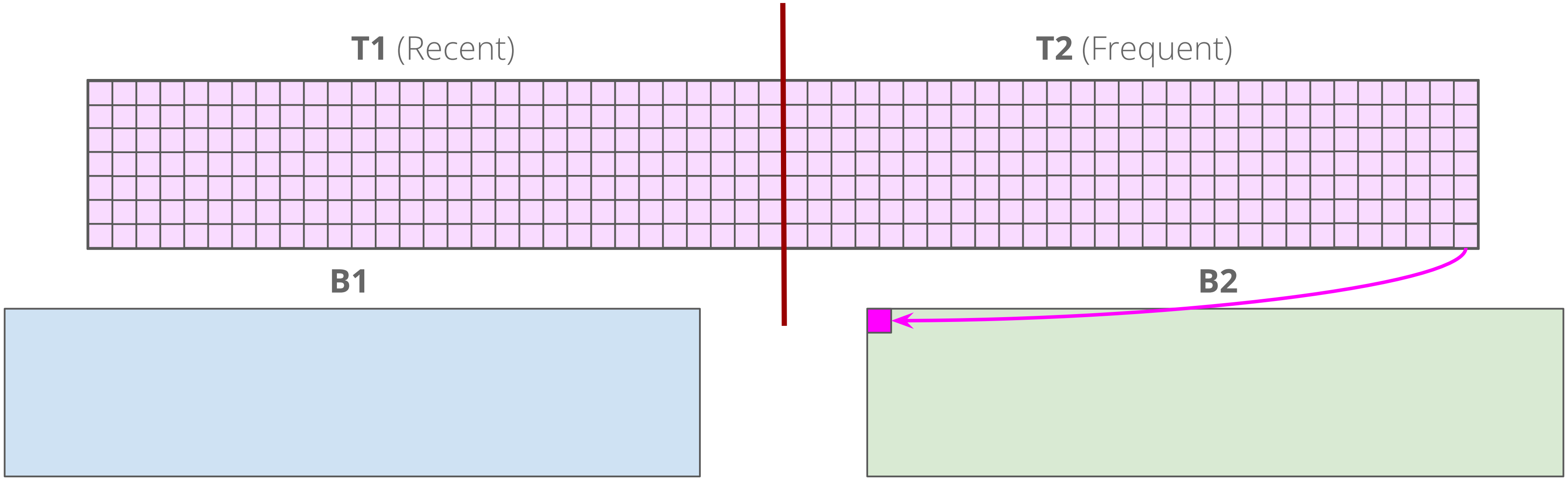


Adaptive Replacement Cache (ARC)



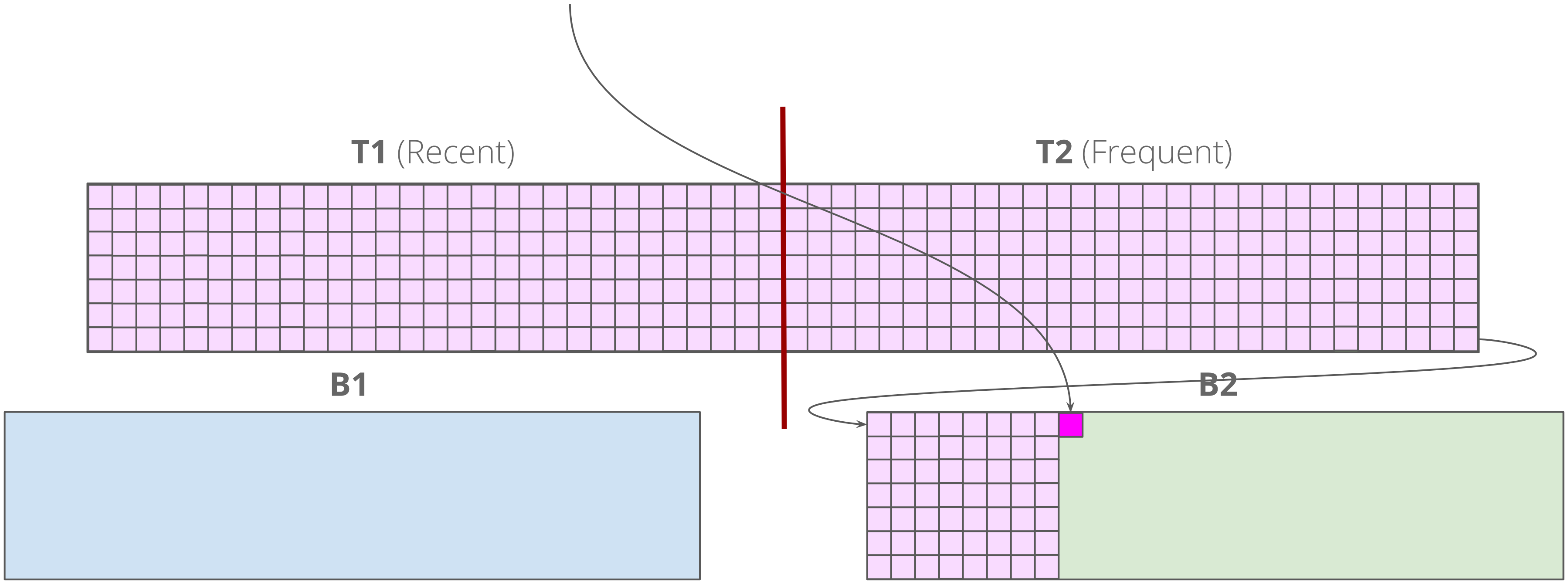
Adaptive Replacement Cache (ARC)

Our block will eventually be evicted from **T2**. It will be *REFERENCED* in **B2** but its data will not actually be stored there



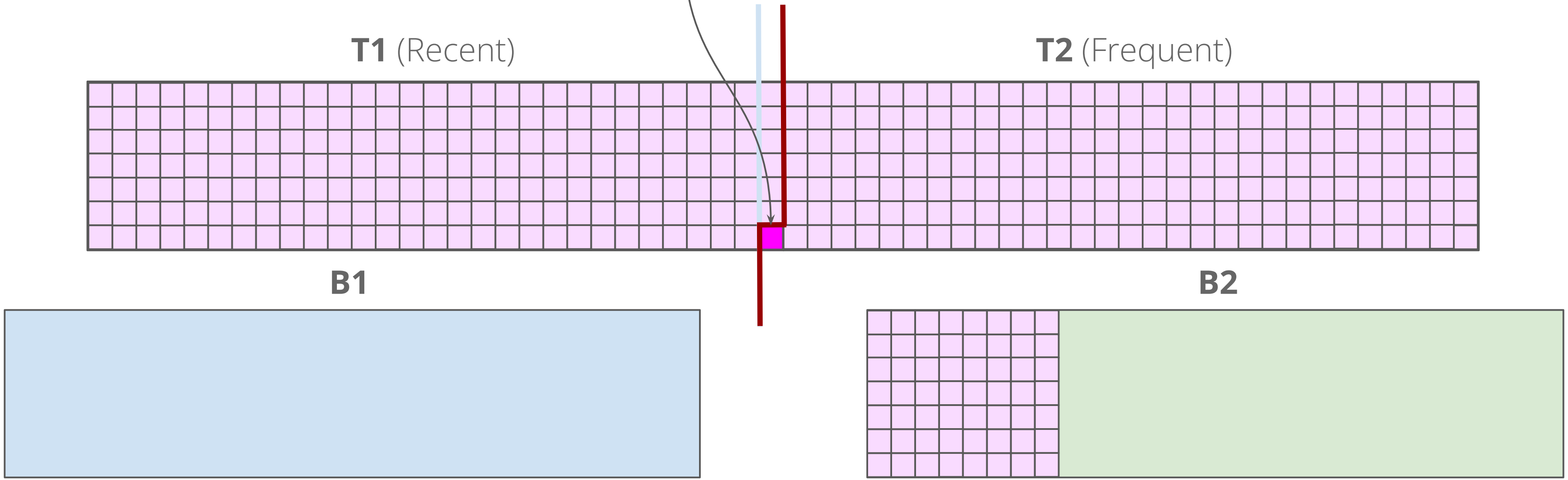
Adaptive Replacement Cache (ARC)

B2 will fill up as blocks are evicted from **T2**



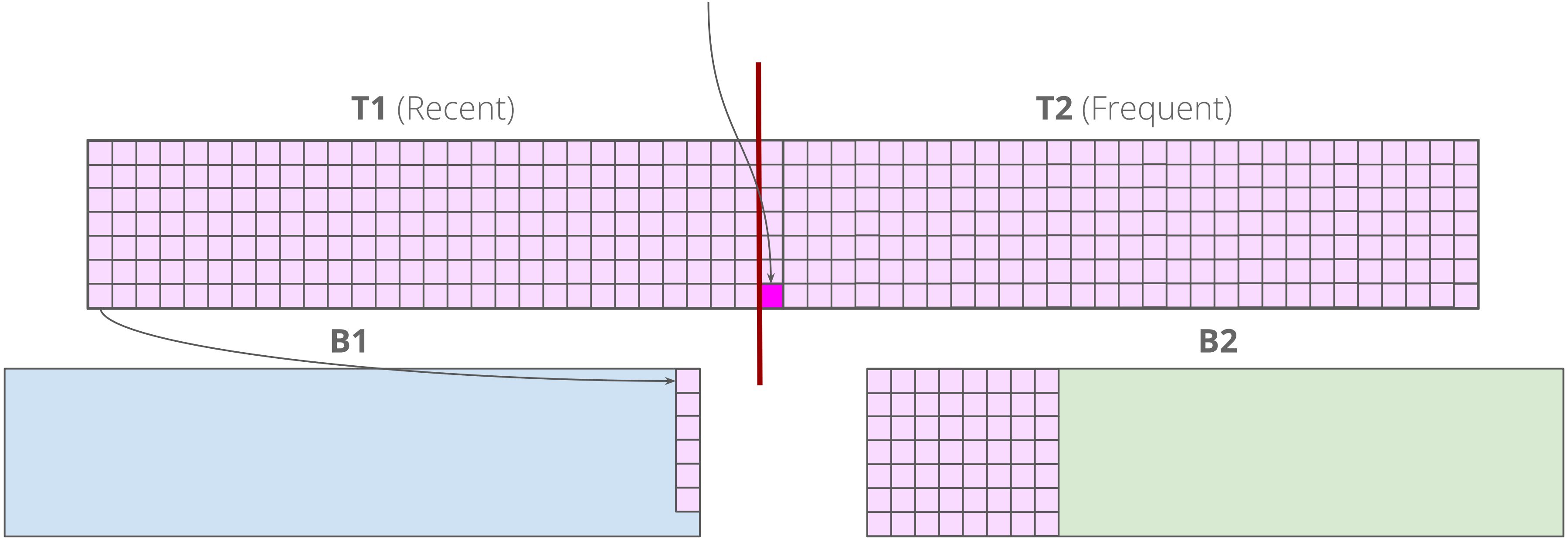
Adaptive Replacement Cache (ARC)

If our block is referenced again, it will get re-read from disk and loaded into **T2**. **T1/T2 division** will also shift, increasing **T2** size (decreasing **T1** size). The blue line is the target **T1/T2** sizes. **T1** will shrink as items are evicted, **T2** will grow as items are added.



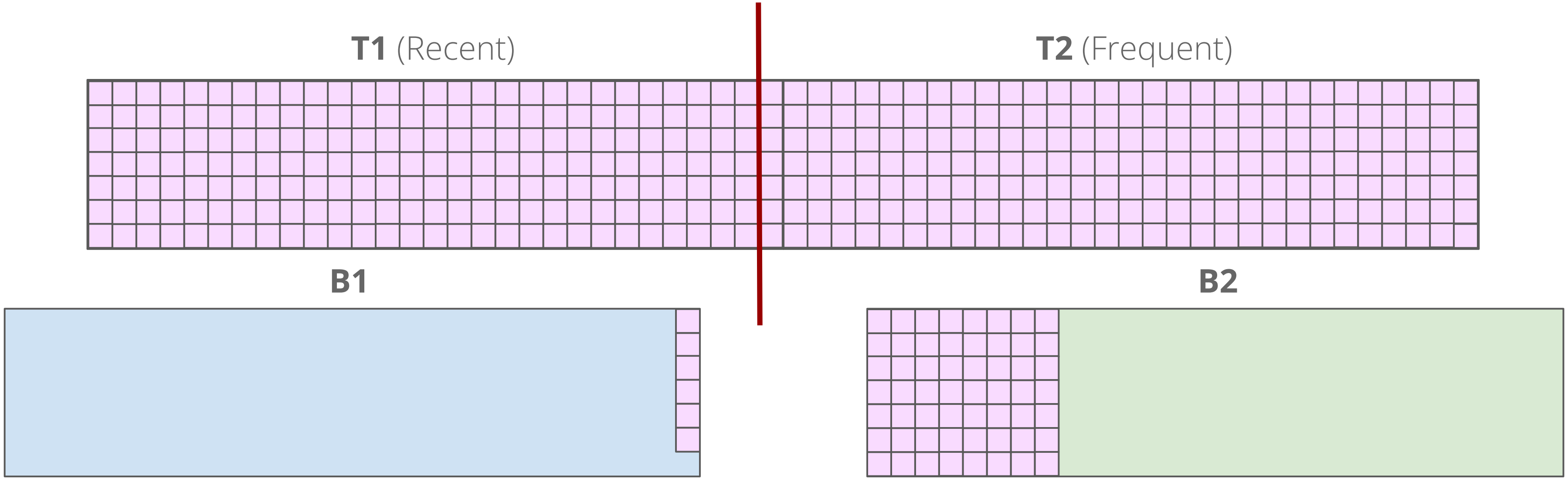
Adaptive Replacement Cache (ARC)

After 6 items are added to **T2**, 6 items will be evicted from **T1** (but still tracked in **B1**!), **T1** and **T2** will reach their target sizes. Note that our pink block didn't get pushed closer to end of the list. Instead, the beginning of the list just grew (i.e., it grew from head, not from tail).



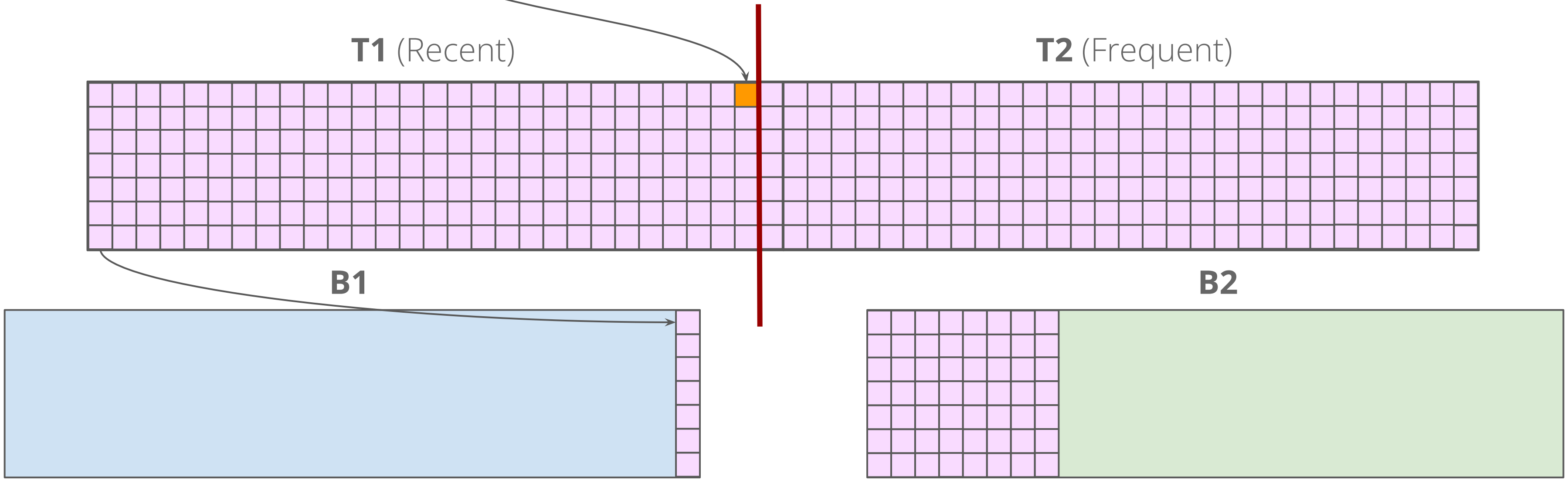
Adaptive Replacement Cache (ARC)

We'll forget about the pink block and add a new, orange block the the cache!



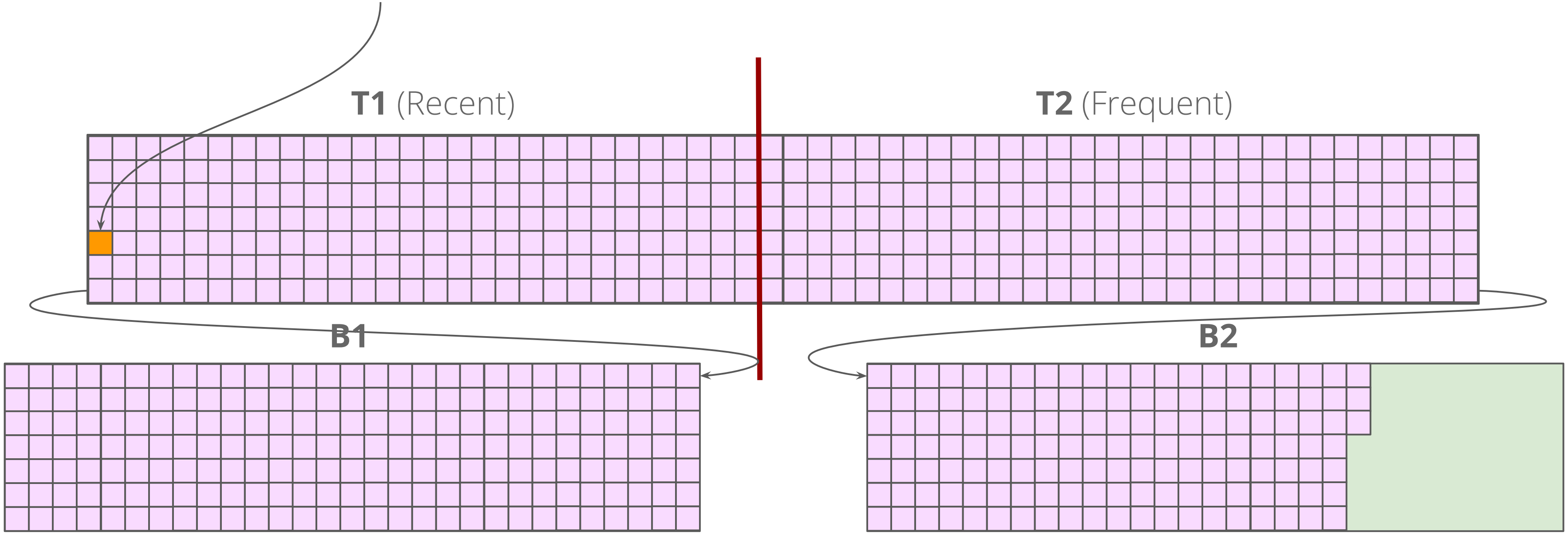
Adaptive Replacement Cache (ARC)

The orange block is added to top of **T1** which evicts bottom item in **T1** to **B1**



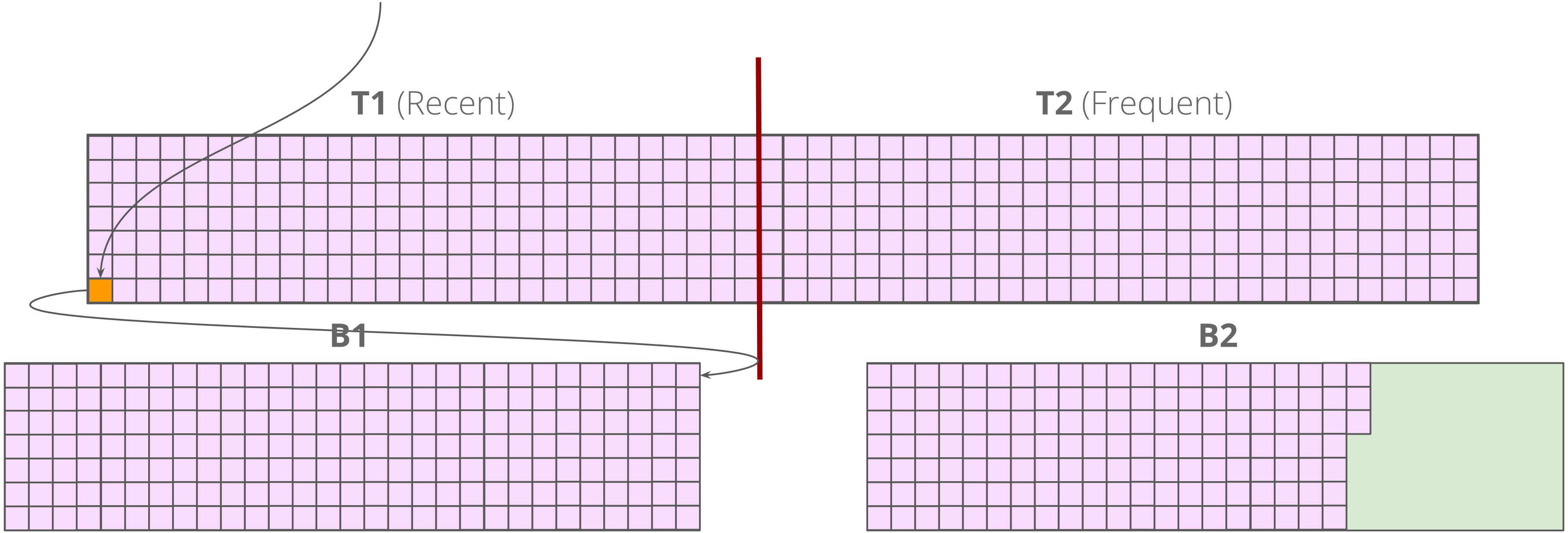
Adaptive Replacement Cache (ARC)

T1 will fill up with new data, pushing orange block down. Data referenced twice will move to **T2**, evicting blocks to **B2**



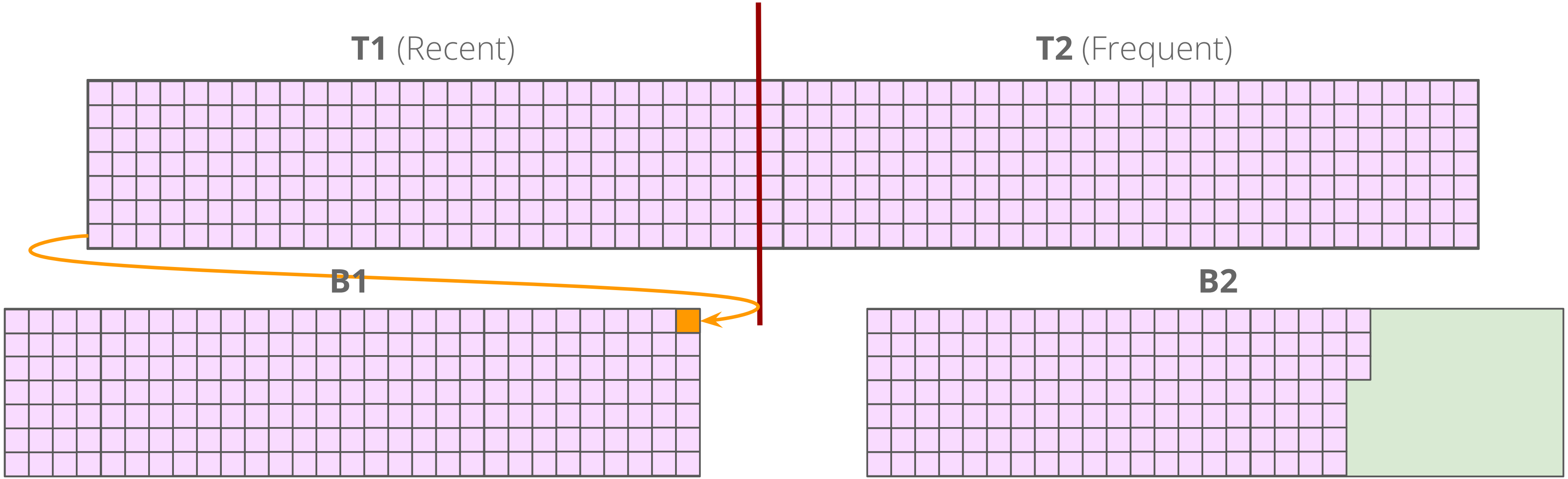
Adaptive Replacement Cache (ARC)

The orange block keeps getting pushed down, and eventually...



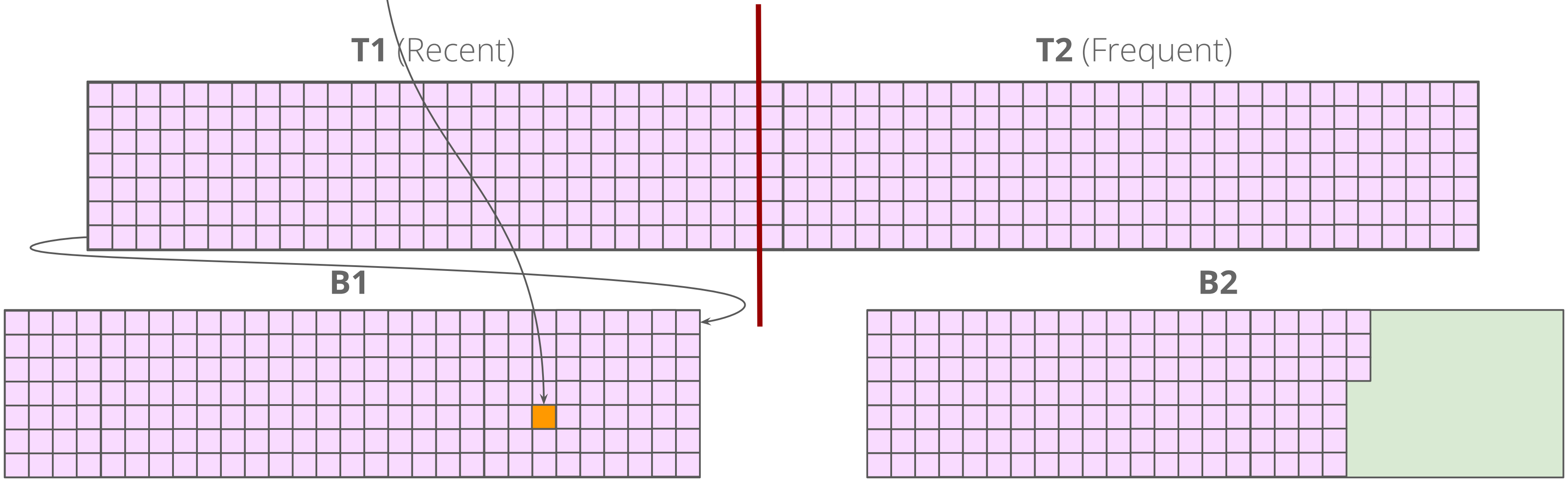
Adaptive Replacement Cache (ARC)

...gets evicted to **B1**. The orange block data is not stored in **B1**, just its reference.



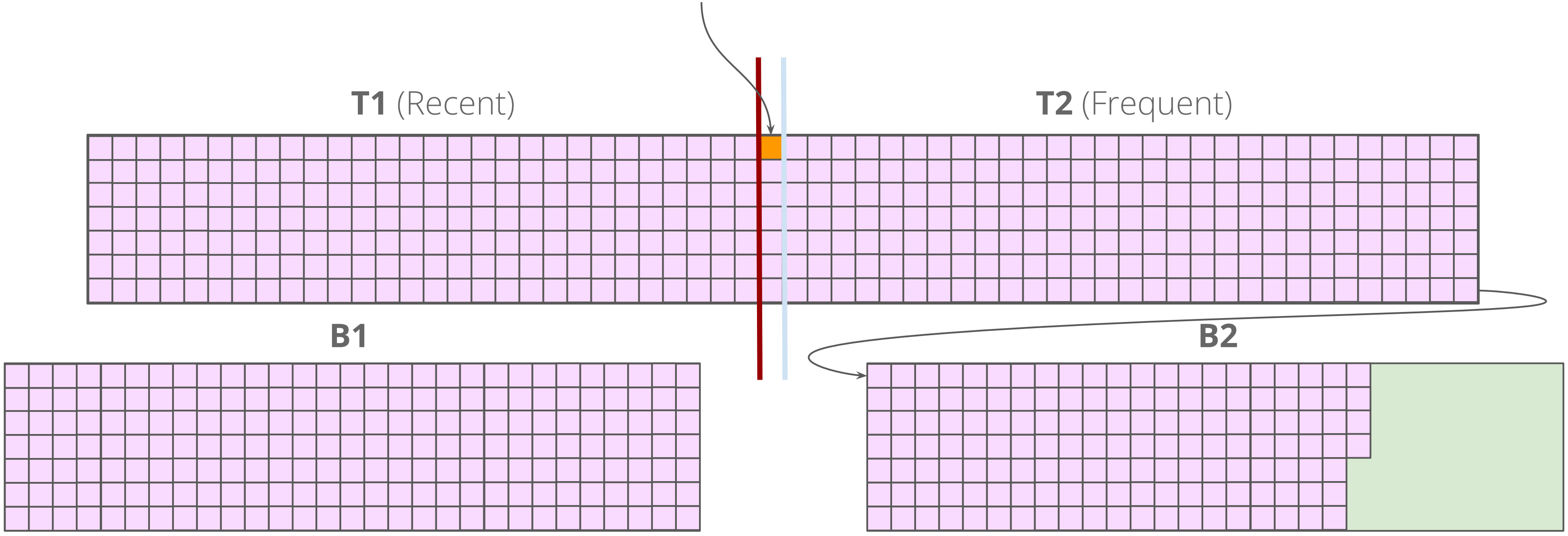
Adaptive Replacement Cache (ARC)

It will continue to get pushed down in **B1**



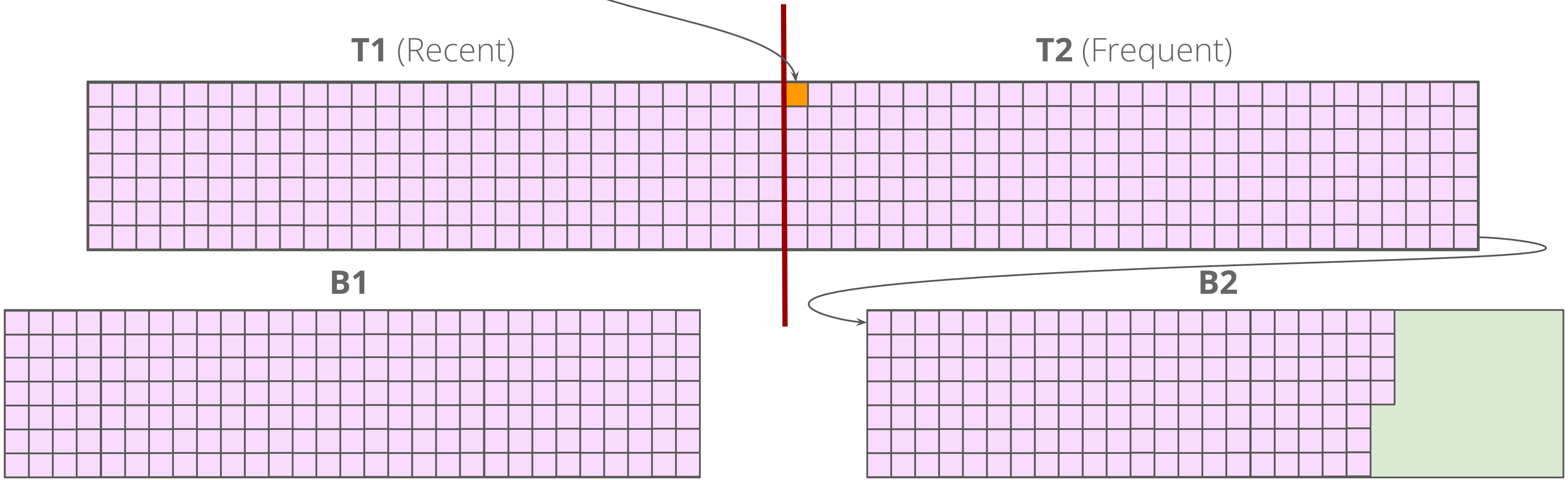
Adaptive Replacement Cache (ARC)

If orange block gets referenced again, it goes to top of **T2**. The **T1/T2 division** will shift, increasing **T1** size (decreasing **T2** size). The Blue line is target **T1/T2** sizes. **T1** will grow as items are added, **T2** will shrink as items are evicted.



Adaptive Replacement Cache (ARC)

After 6 items are added to **T1**, 6 items will be evicted from **T2** (but still tracked in **B2**!), **T1** and **T2** will reach their target sizing. This time, **T1** grows from its head rather than its tail.



Adaptive Replacement Cache (ARC)

- Typically, the size of T1 + T2 (i.e., total ARC size) remains constant and the division between them just shifts back and forth, but ZFS can also increase total ARC size if it's needed.
- **L2ARC** works the same way, but its entries are just items that have been evicted from T1 or T2. Items in L2ARC's T1 or T2 could also be in L1ARC's B1 or B2.
- ZFS puts both **Reads & Writes** into the ARC!
- B1 & B2 will eventually fill up, too. Items that fall out of B1 or B2 will be treated as brand new data when it's referenced again!
- We showed the target T1/T2 division shifting by 7 blocks at a time; this was an arbitrary choice for the sake of the diagram. I'm not sure how much it actually shifts by in practice, but it's probably controllable by the admin via a ZFS tunable.
- Workloads that benefit from larger ARC are workloads that re-use the same data over and over! For example, virtualization reuses the OS data; very large office file shares re-use the same doc files; a video editor will work on the same set of video files, many of which might be cached.
- Workloads that don't benefit from larger ARC are workloads that don't re-use the same data. For example, with video surveillance, data comes in, is written, and is (typically) never accessed again