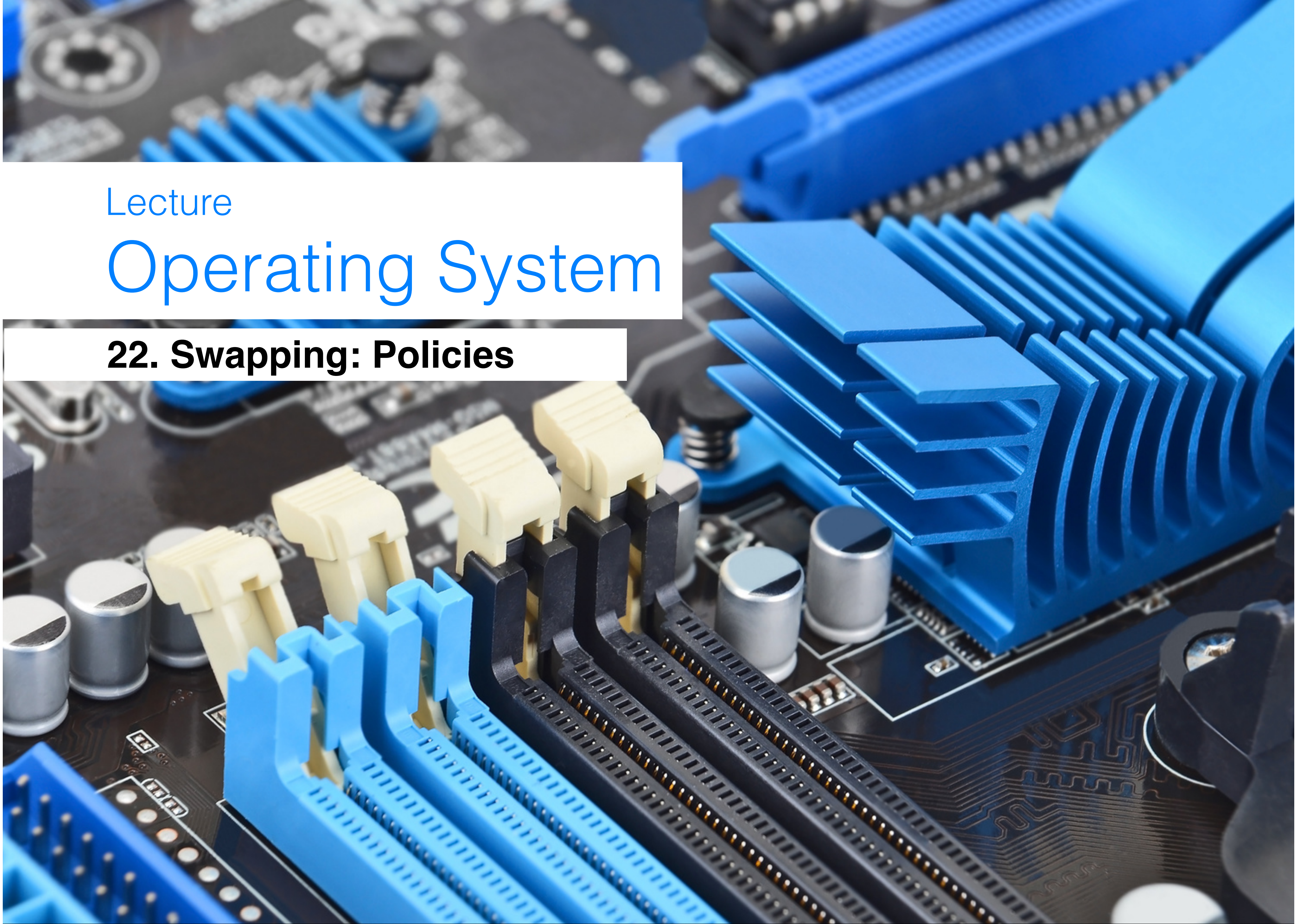


Lecture

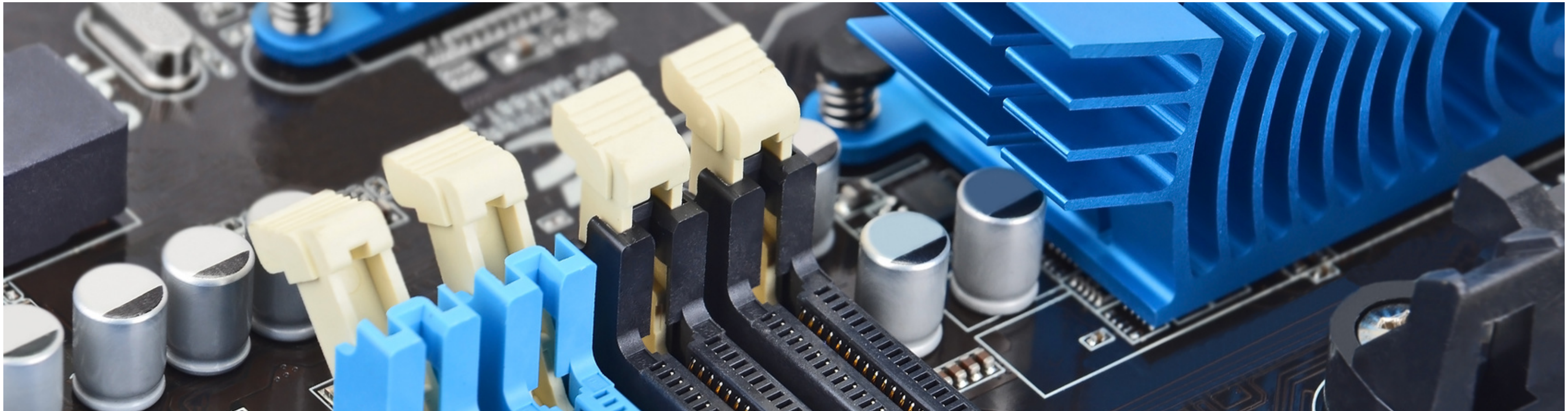
Operating System

22. Swapping: Policies

Professor Dr. Michael Mächtel

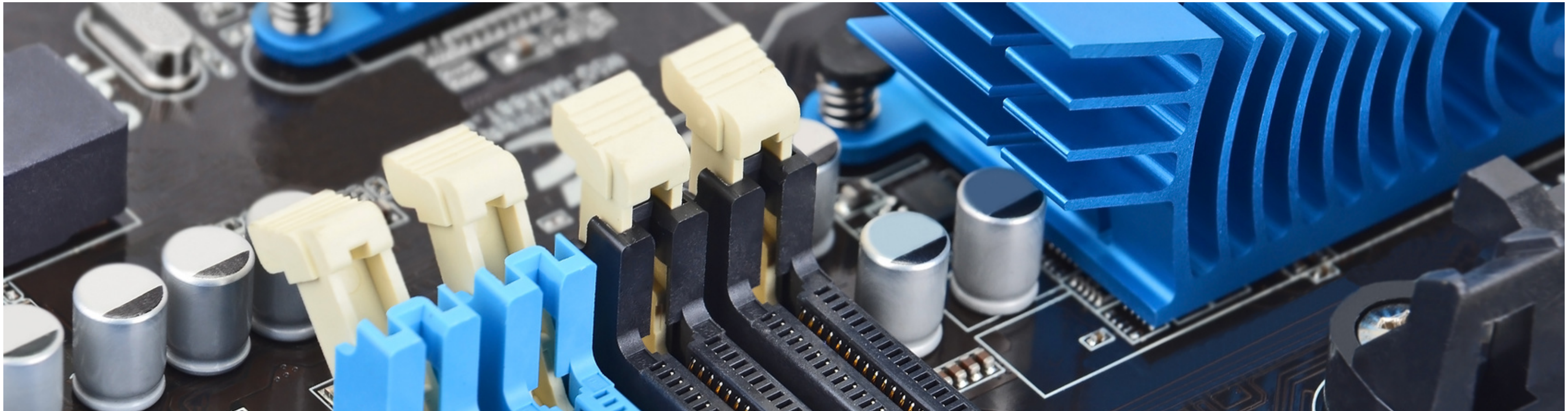
22. Swapping: Policies

1. **Goal of Replacement Policies**
2. **Replacement Policies**
3. **Page Selection Policies**



22. Swapping: Policies

1. **Goal of Replacement Policies**
2. Replacement Policies
3. Page Selection Policies



Beyond Physical Memory: Policies

- Memory *pressure* forces the OS to start **paging out** pages to make room for actively-used pages.
- Deciding which page to *evict* is encapsulated within the replacement policy of the OS.

Cache Management

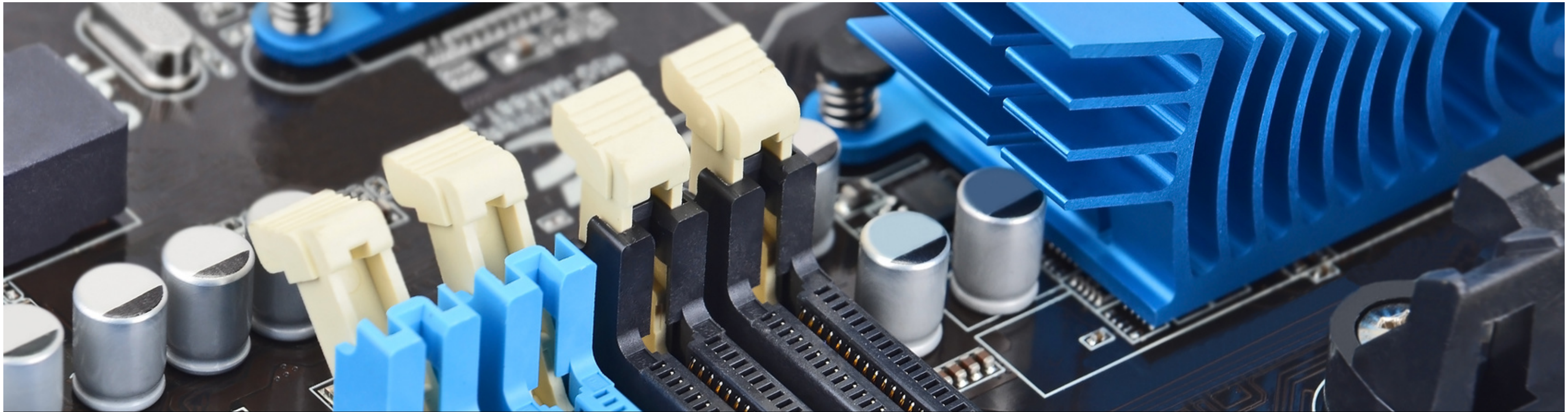
- **Goal** in picking a replacement policy for this cache is to minimize the number of cache misses.
- The number of cache hits and misses let us calculate the **average memory access time (AMAT)**.

$$AMAT = (P_{Hit} * T_M) + (P_{Miss} * T_D)$$

Arguement	Meaning
T_M	The cost of accessing memory
T_D	The cost of accessing disk
P_{Hit}	The probability of finding the data item in the cache(a hit)
P_{Miss}	The probability of not finding the data in the cache(a miss)

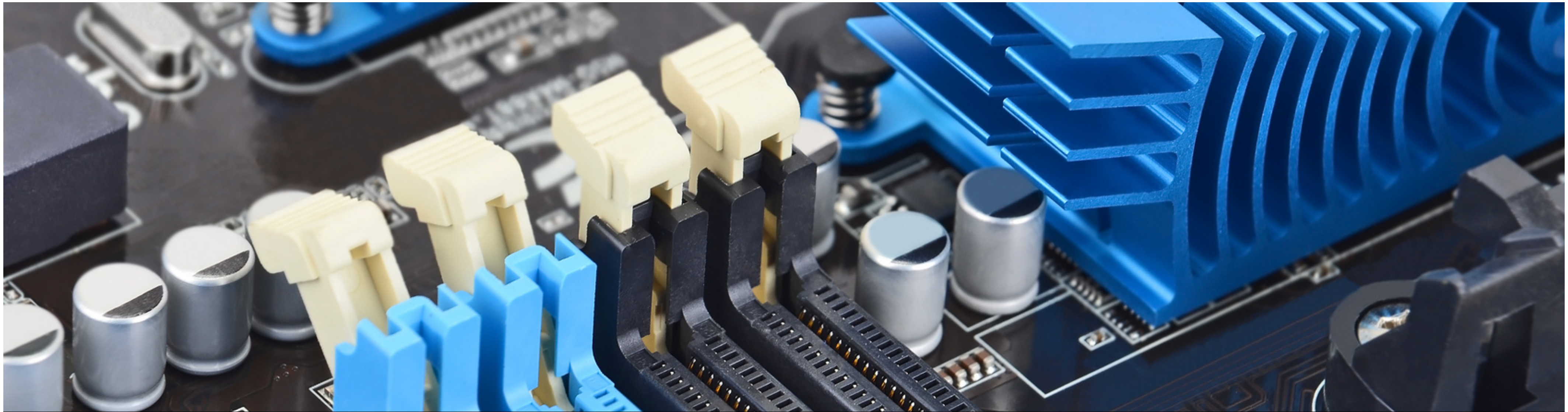
22. Swapping: Policies

1. Goal of Replacement Policies
2. **Replacement Policies**
3. Page Selection Policies



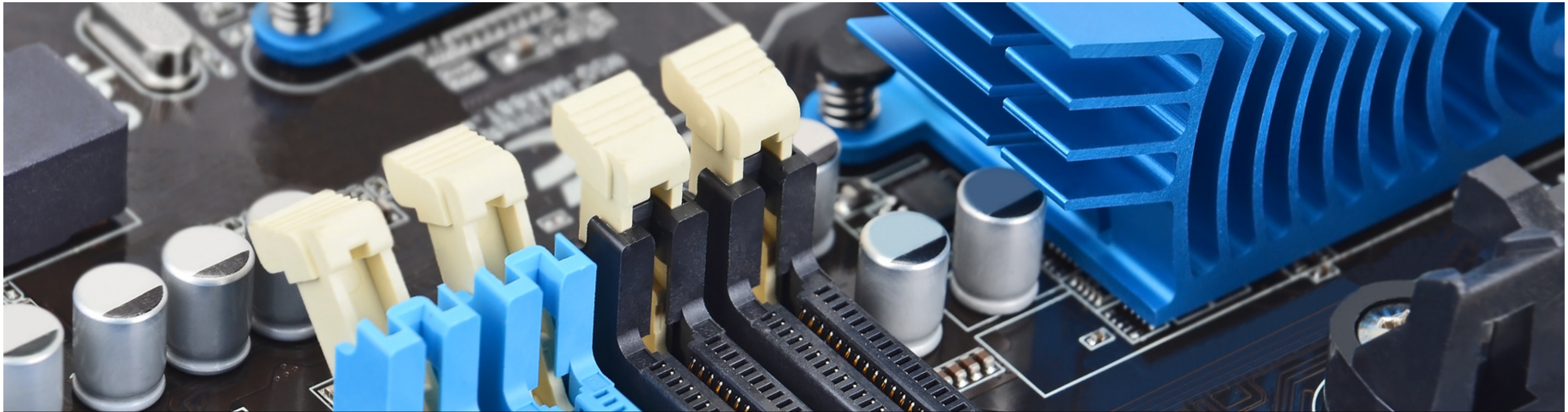
22. Swapping: Replacement Policies

1. **Policies**
2. **Workload**
3. **Implementation with HW Support**



22. Swapping: Replacement Policies

1. **Policies**
2. **Workload**
3. **Implementation with HW support**



The Optimal Replacement Policy

- Leads to the fewest number of misses overall
  - Replaces the page that will be accessed **furthest in the future**
  - Resulting in the **fewest-possible** cache misses
- Serve only as a comparison point, to know how close we are to perfect

Tracing the Optimal Policy

Reference Row

0 1 2 0 1 3 0 3 1 2 1

cache fits **three** pages.

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		0,1,2
1	Hit		0,1,2
3	Miss	2	0,1,3
0	Hit		0,1,3
3	Hit		0,1,3
1	Hit		0,1,3
2	Miss	3	0,1,2
1	Hit		0,1,2

Hit rate is:

$$\frac{Hits}{Hits + Misses} = 54{,}6\%$$

Future is not known.

A Simple Policy: FIFO

- Pages were placed in a queue when they enter the system.
- When a replacement occurs, the page on the tail of the queue(the “**First-in**” pages) is evicted.
  - It is simple to implement, but can’t determine the importance of blocks.

Tracing the FIFIO Policy

Reference Row

0 1 2 0 1 3 0 3 1 2 1

cache fits **three** pages.

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		0
1	Miss		0,1
2	Miss		0,1,2
0	Hit		0,1,2
1	Hit		0,1,2
3	Miss	0	1,2,3
0	Miss	1	2,3,0
3	Hit		2,3,0
1	Miss	2	3,0,1
2	Miss	3	0,1,2
1	Hit		0,1,2

Hit rate is:

$$\frac{Hits}{Hits + Misses} = 36{,}4\%$$

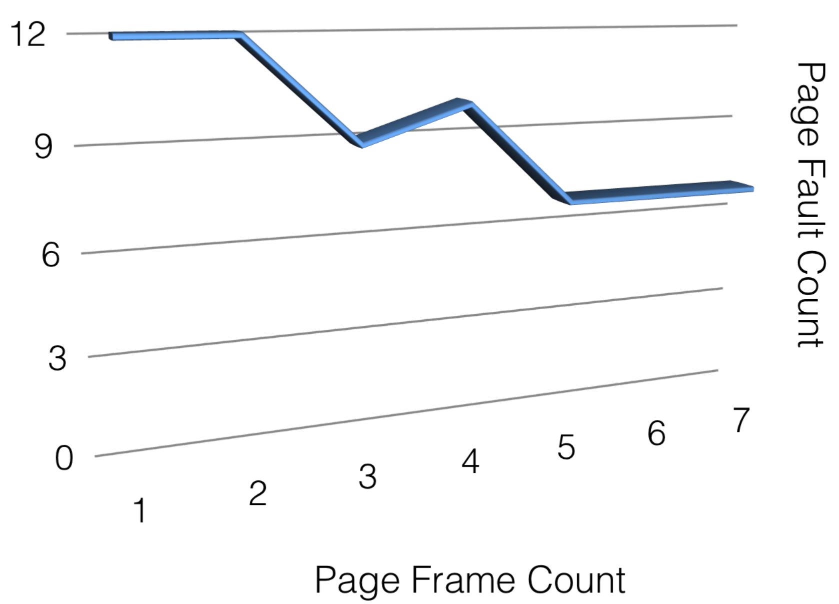
Even though page 0 had been accessed a number of times, <u>FIFO still kicks it out</u>.

BELADY'S ANOMALY

- We would expect the cache hit rate to **increase** when the cache gets **larger**. But in this case, with FIFO, it gets worse.

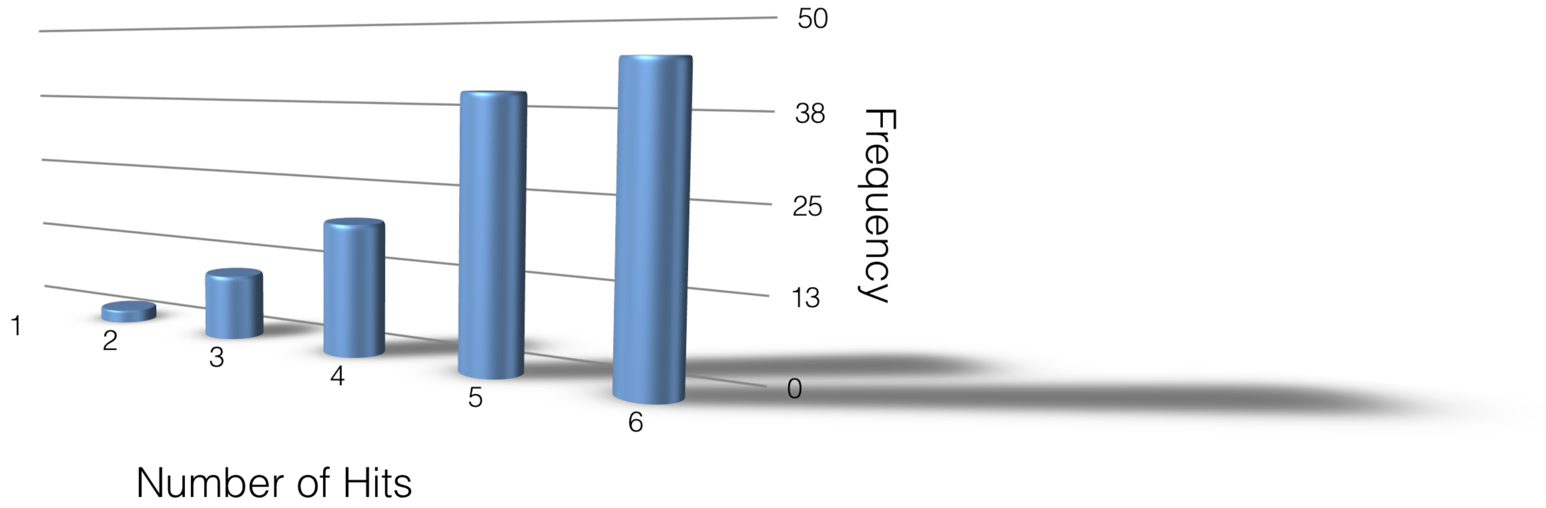
Reference Row

1 2 3 4 1 2 5 1 2 3 4



Another Simple Policy: Random

- Picks a random page to replace under memory pressure.
  - It doesn't really try to be too intelligent in picking which blocks to evict.
  - Random does depends entirely upon how lucky **Random** gets in its choice.
- Sometimes, **Random is as good as optimal**, achieving 6 hits on the example trace.



Using History in Policies

- Lean on the past and use **history**.
  - **Two type** of historical information.

Historical Information	Meaning	Algorithms
recency	The more recently a page has been accessed, the more likely it will be accessed again	**LRU**
frequency	If a page has been accessed many times, It should not be replcaed as it clearly has some value	**LFU**

Tracing the LRU Policy

Reference Row

0 1 2 0 1 3 0 3 1 2 1

- cache fits **three** pages.
- Replaces the least-recently-used page

Access	Hit/Miss?	Evict	Resulting Cache State
0	Miss		LRU → 0
1	Miss		LRU → 0,1
2	Miss		LRU → 0,1,2
0	Hit		LRU → 1,2,0
1	Hit		LRU → 2,0,1
3	Miss	2	LRU → 0,1,3
0	Hit		LRU → 1,3,0
3	Hit		LRU → 1,0,3
1	Hit		LRU → 0,3,1
2	Miss	0	LRU → 3,1,2
1	Hit		LRU → 3,2,1

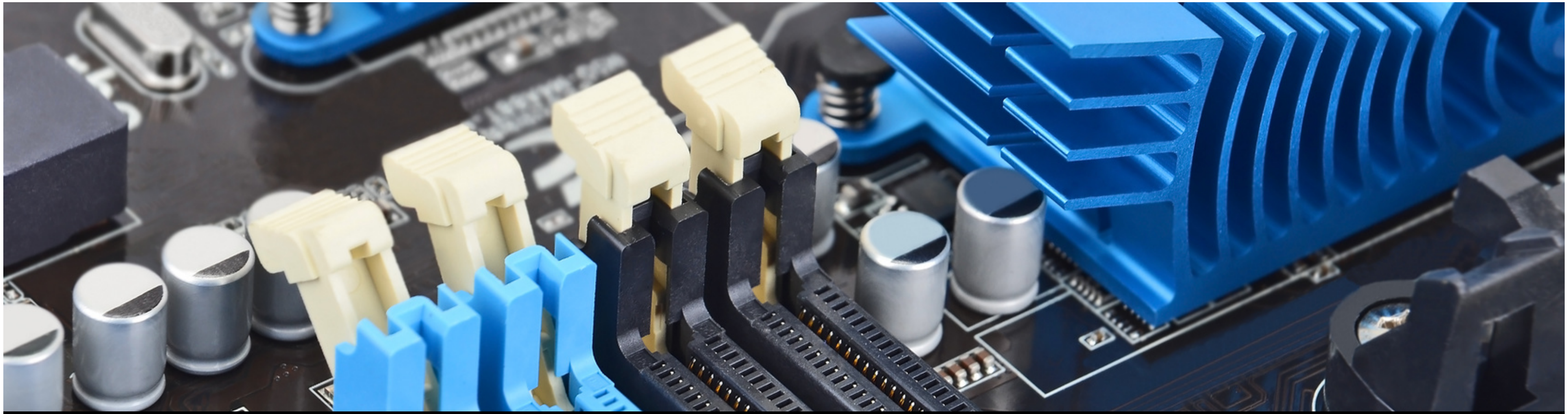
Hit rate is:

$$\frac{Hits}{Hits + Misses} = 54,6\%$$

In this example as good as optimal

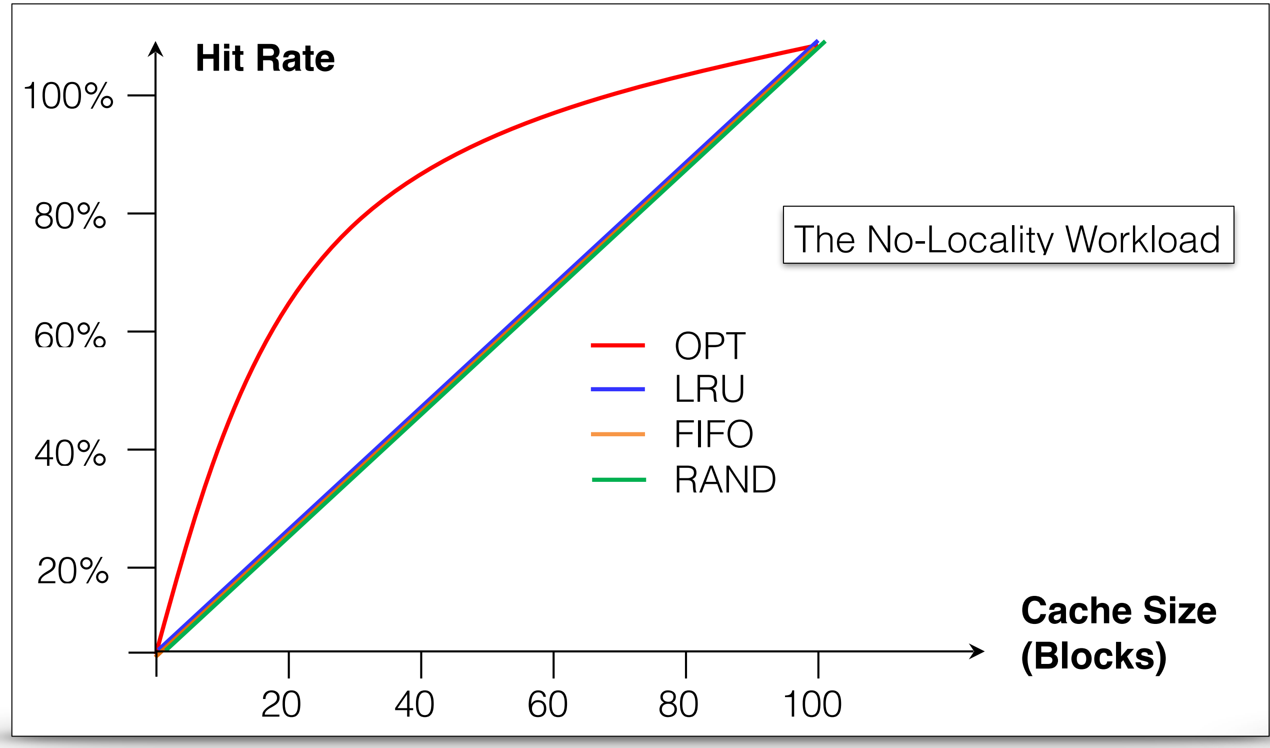
22. Swapping: Replacement Policies

1. Policies
2. **Workload**
3. Implementation with HW support



Example: The No-Locality Workload

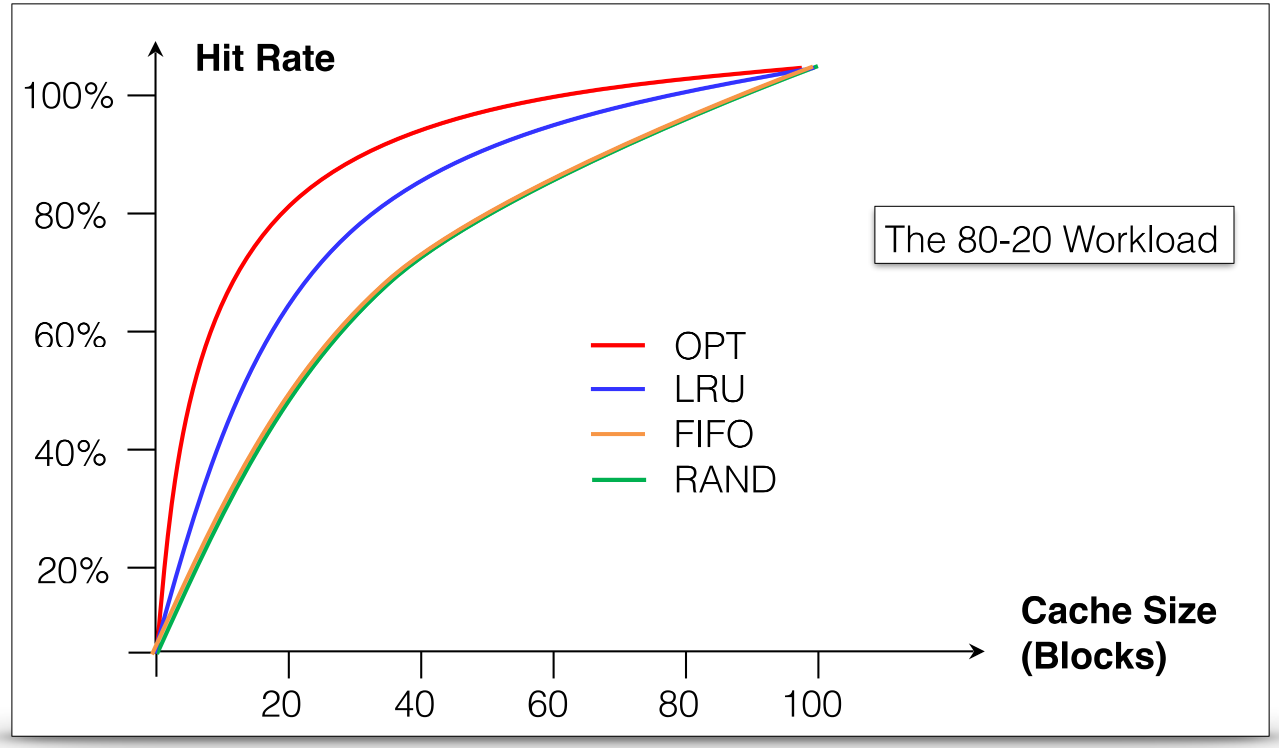
- Each reference is to a random page within the set of accessed pages.
  - Workload accesses 100 unique pages over time.
  - Choosing the next page to refer to at random



When the cache is large enough to fit the entire workload, it also **doesn't matter** which policy you use.

Example: The 80-20 Workload

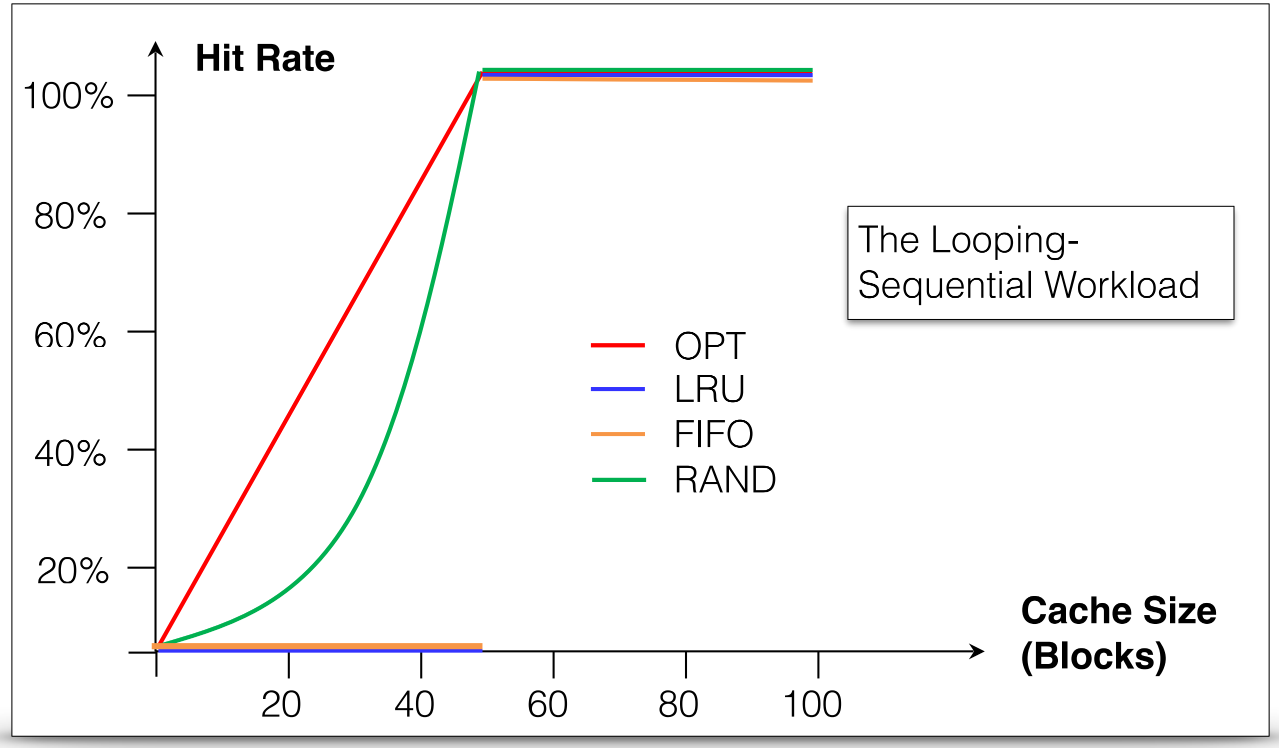
- Exhibits locality: 80% of the **reference** are made to 20% of the page
- The remaining 20% of the **reference** are made to the remaining 80% of the pages.



LRU is more likely to hold onto the **hot pages**.

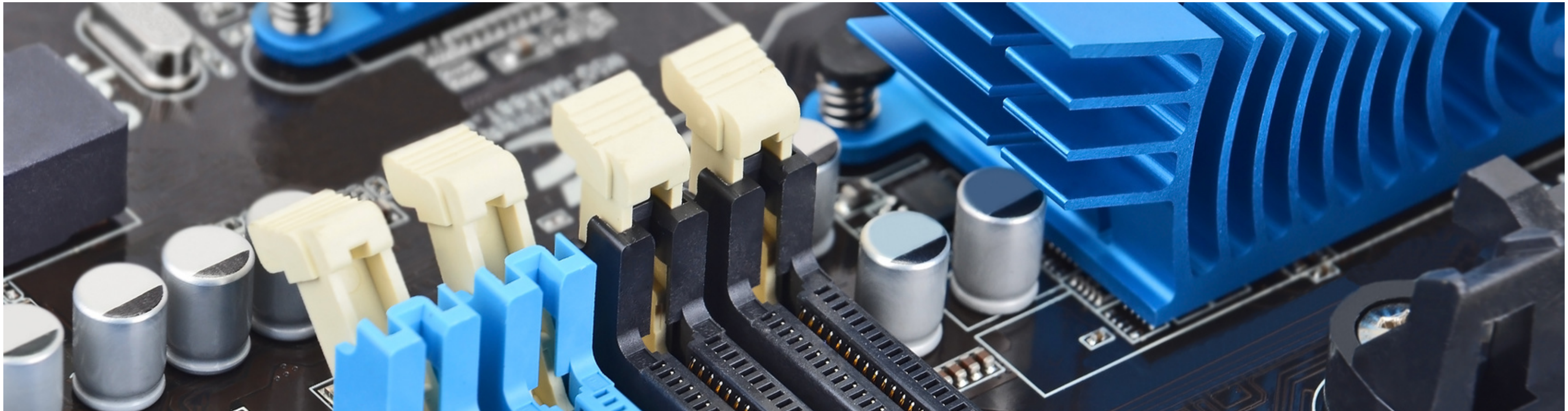
Example: The Looping Sequential

- Refer to 50 pages in sequence.
    - Starting at 0, then 1, … up to page 49, and then we Loop, repeating those accesses, for total of 10,000 accesses to 50 unique pages.



22. Swapping: Replacement Policies

1. Policies
2. Workload
3. **Implementation with HW support**



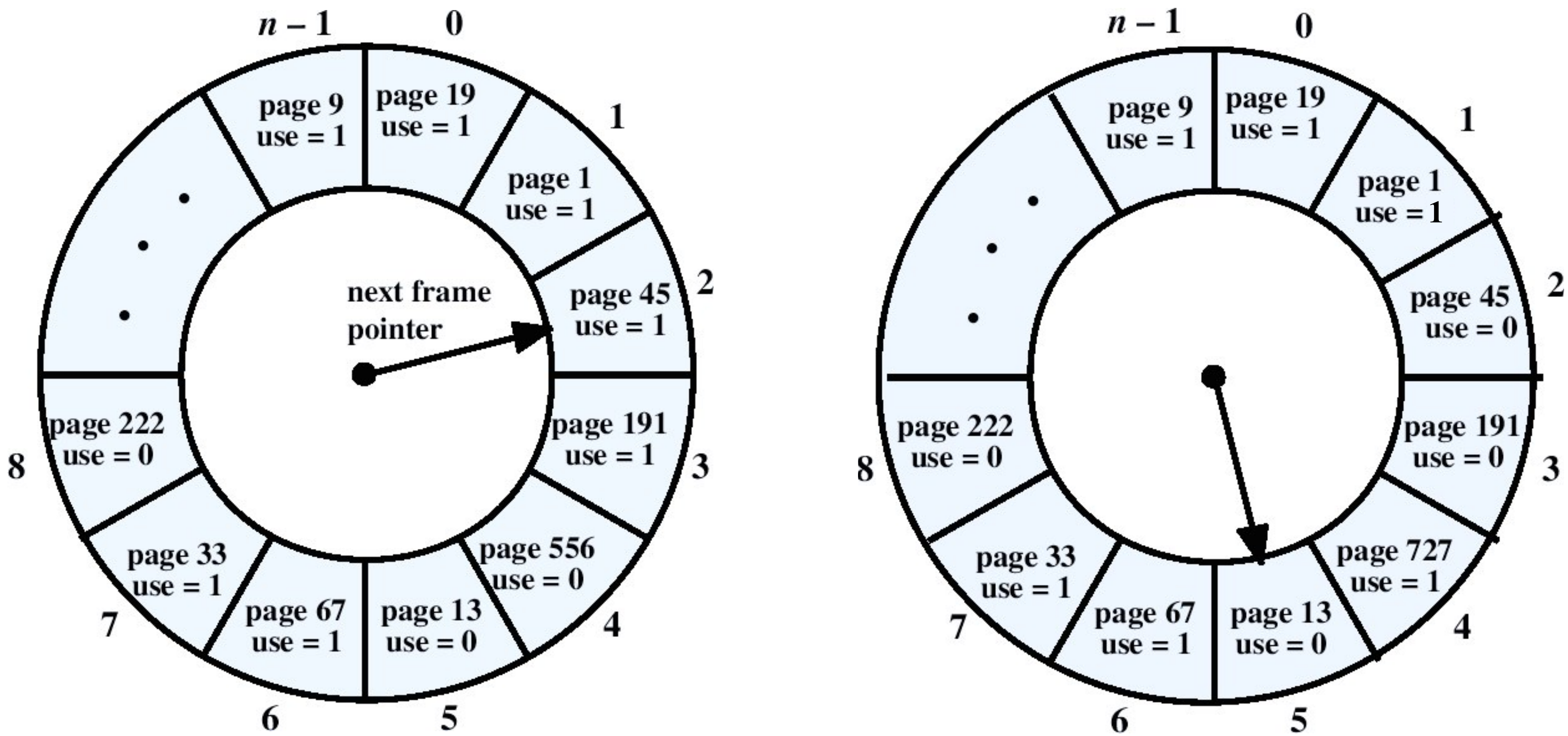
Implementing Historical Algorithms

- To keep track of which pages have been least-and-recently used, the system has to do some accounting work on **every memory reference**.
  - **Add a little bit** of hardware support.
- Require some hardware support, in the form of a **use bit**
  - Whenever a **page is referenced**, the **use bit** is set by hardware to **1**.
  - Hardware **never** clears the bit, though; that is the responsibility of the OS
- **Clock Algorithm**
  - All pages of the system arranges in a circular list.
  - A clock hand points to some particular page to begin with.

Clock Algorithm

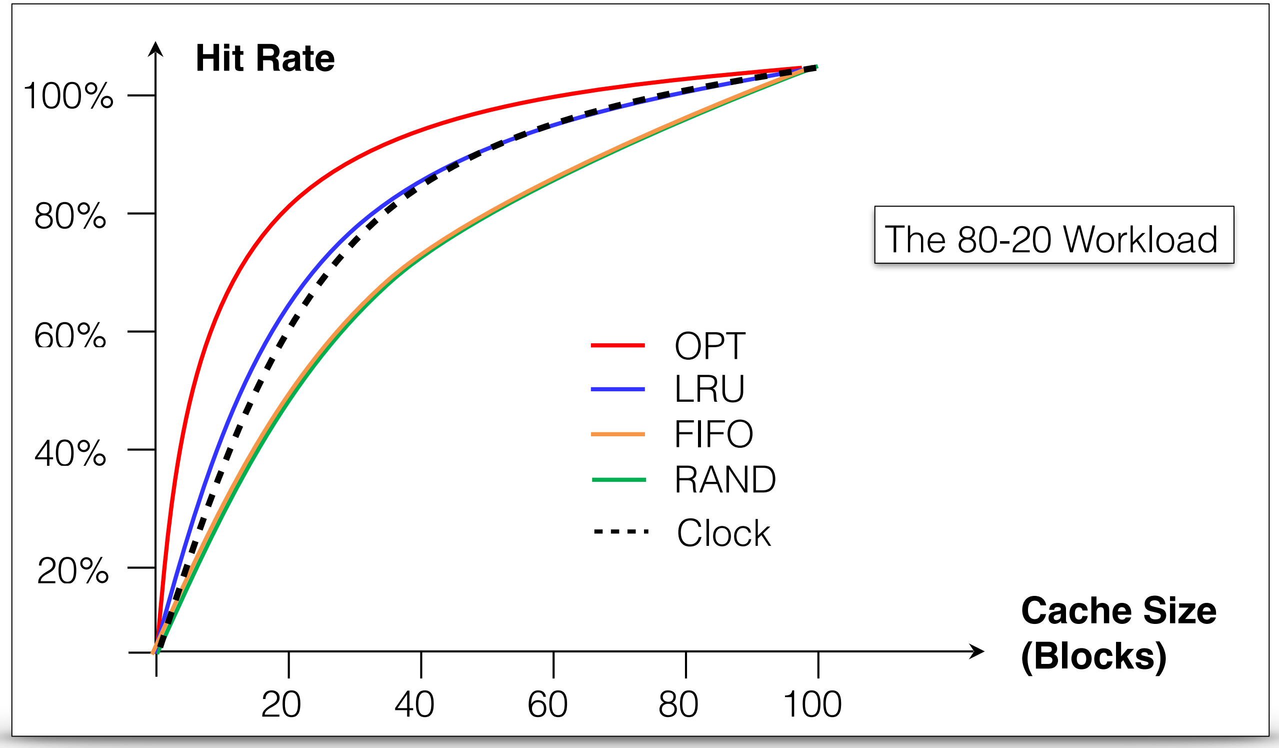
- **Hardware**
  - Keep use (or reference) bit for each page frame
  - When page is referenced: set use bit
- **Operating System**
  - Page replacement: Look for page with use bit cleared (has not been referenced for awhile)
  - Implementation:
    - Keep pointer to last examined page frame
    - Traverse pages in circular buffer
    - Clear use bits as search
    - Stop when find page with already cleared use bit, replace this page

Clock Algorithm: Example



Workload with Clock Algorithm

- Clock algorithm doesn't do as well as perfect LRU, it does better then approach that don't consider history at all.

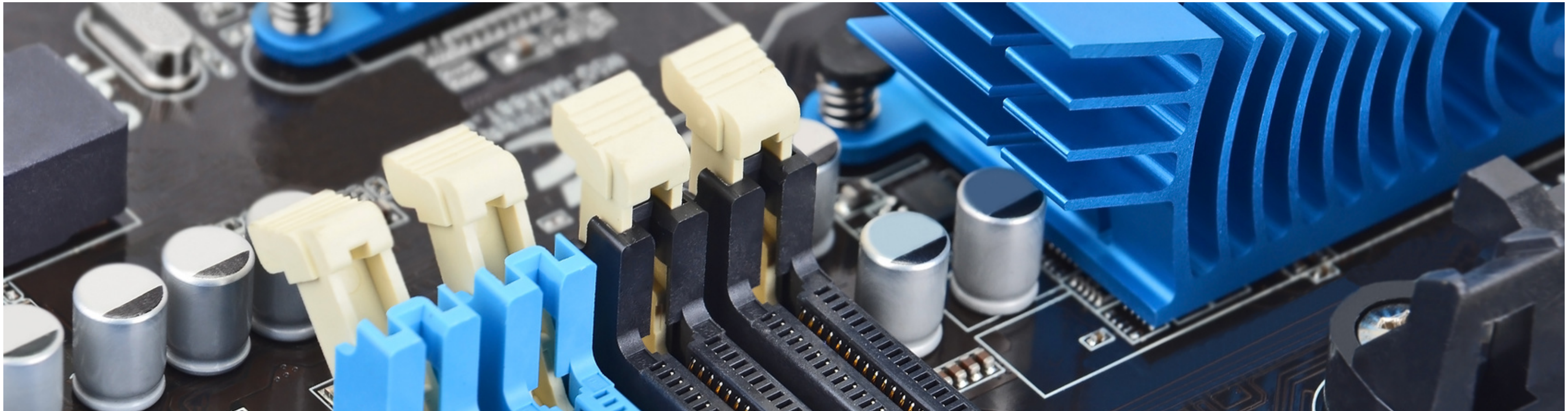


Considering Additional HW Bits

- The hardware include a **modified** bit (a.k.a dirty bit)
  - Page has been modified and is thus dirty, it **must be written back** to disk to evict it.
  - Page has not been modified, the eviction is free.
- **NRU: Not Recently Used**
  - Use bit and dirty bit are used to build classes of pages
    - **class 0**: not been referenced and not been modified
    - **class 1**: not been referenced but modified
    - **class 2**: referenced but not been modified
    - **class 3**: referenced and modified
  - **Evict** one **random** page from **lowest** not empty class

22. Swapping: Policies

1. Goal of Replacement Policies
2. Replacement Policies
3. **Page Selection Policies**

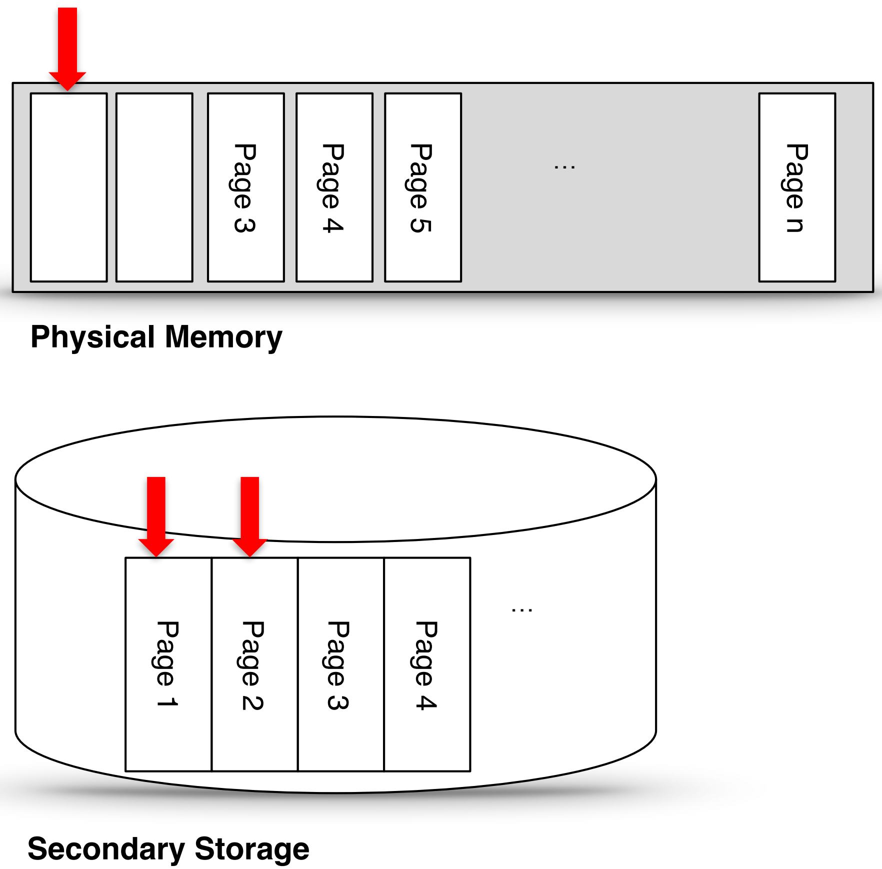


Page Selection Policy

- The OS has to decide **when** to bring a page **into** memory.
  - Presents the OS with some **different options**.
    - Demand Paging: Load the page which is accessed
    - Prefetching: Bring pages in ahead of time
- Another policy determines **how** the OS writes pages **out** to disk
- **Working Set**:
  - the pages that a process uses actively

Prefetching

- The OS **guess** that a page is about to be used, and thus bring it in **ahead** of time.

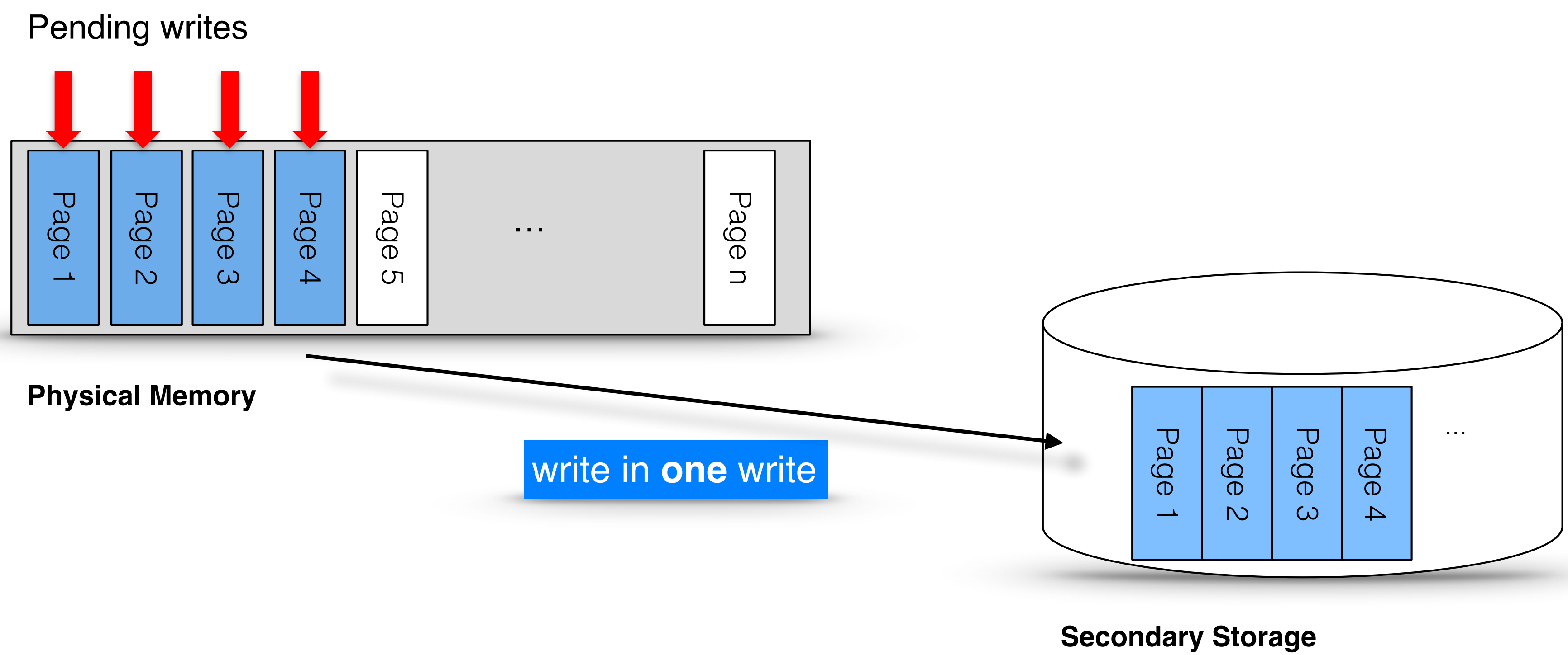


Page 1 is brought into memory

Page 2 likely soon be accessed and thus should be brought into memory too

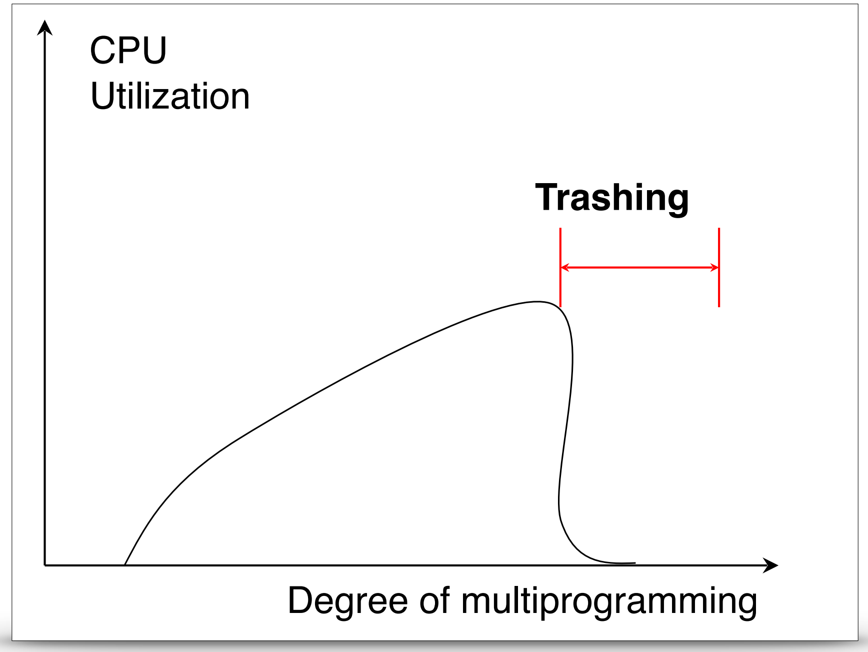
Clustering, Grouping

- Collect a number of **pending writes** together in memory and write them to disk in **one write**.
  - Perform a **single large write** more efficiently than **many small ones**.



Trashing

- Memory is **oversubscribed** and the memory demands of the set of running processes **exceeds** the available physical memory.
  - Decide not to run a subset of processes.
  - Reduced set of processes working sets fit in memory.



Thanks

Questions?