DATAKINETICS

DATA PERFORMANCE & OPTIMIZATION

Making the case for developers to use In-Memory tables

Larry Strickland

Chief Product Officer

Making the case for developers to use In-Memory tables

Presenter:

Larry Strickland

Chief Products Officer

Memory Costs are Decreasing

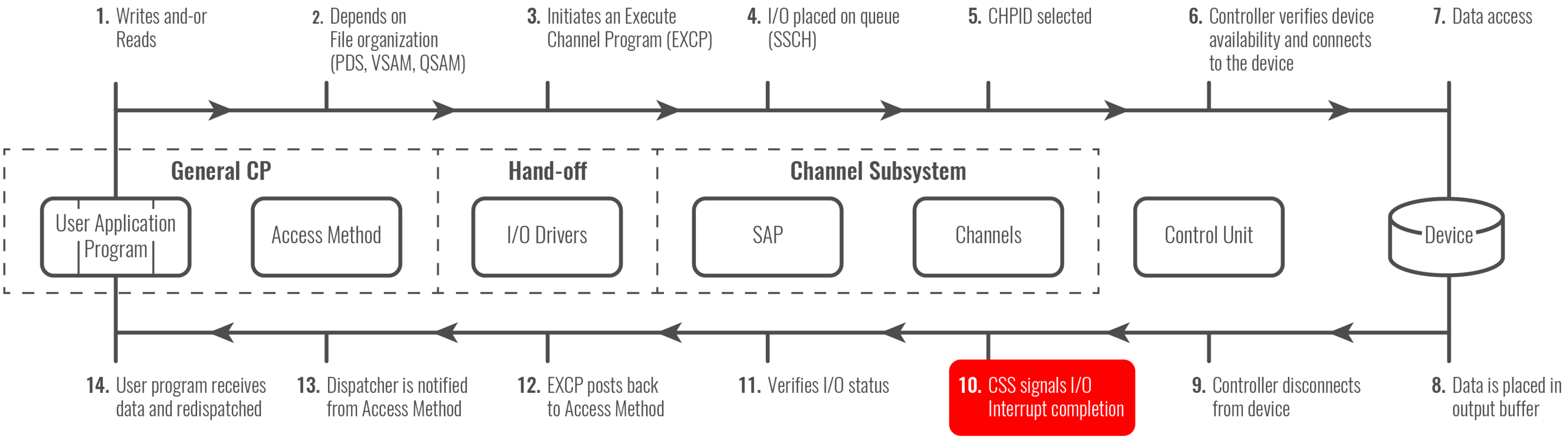
- Although the concept of in-memory processing has been around for a long time, the falling price of RAM and growing use cases have led to a new focus on in-memory techniques and processing

Disk Access is Much Slower Than Memory Access

DATAKINETICS
DATA PERFORMANCE & OPTIMIZATION

- It is orders-of-magnitude more efficient to access data from memory than it is to read it from disk
- Disk I/O is an expensive operation
- Memory access is usually measured in microseconds, whereas disk access is measured in milliseconds
  - 1 millisecond equals 1000 microseconds
- Avoiding I/O improves performance because there is a LOT going on “behind the scenes” when you request an I/O

What is Involved in an I/O Operation?



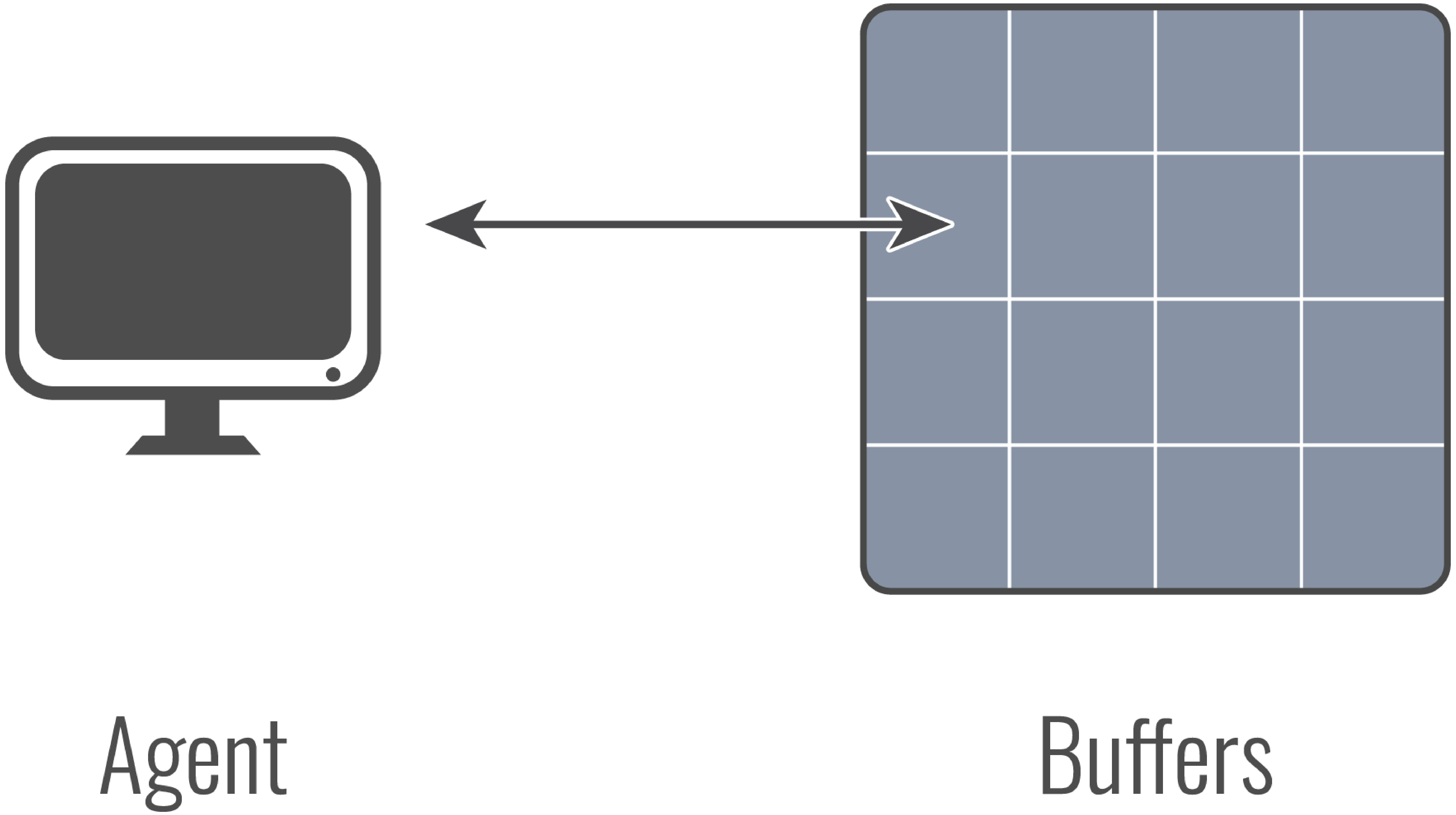
Source: *An I/O White Paper*,
http://idcp.marist.edu/pdfs/ztidbitz/An_IO_WhitePaperForZ.pdf

In-Memory use cases

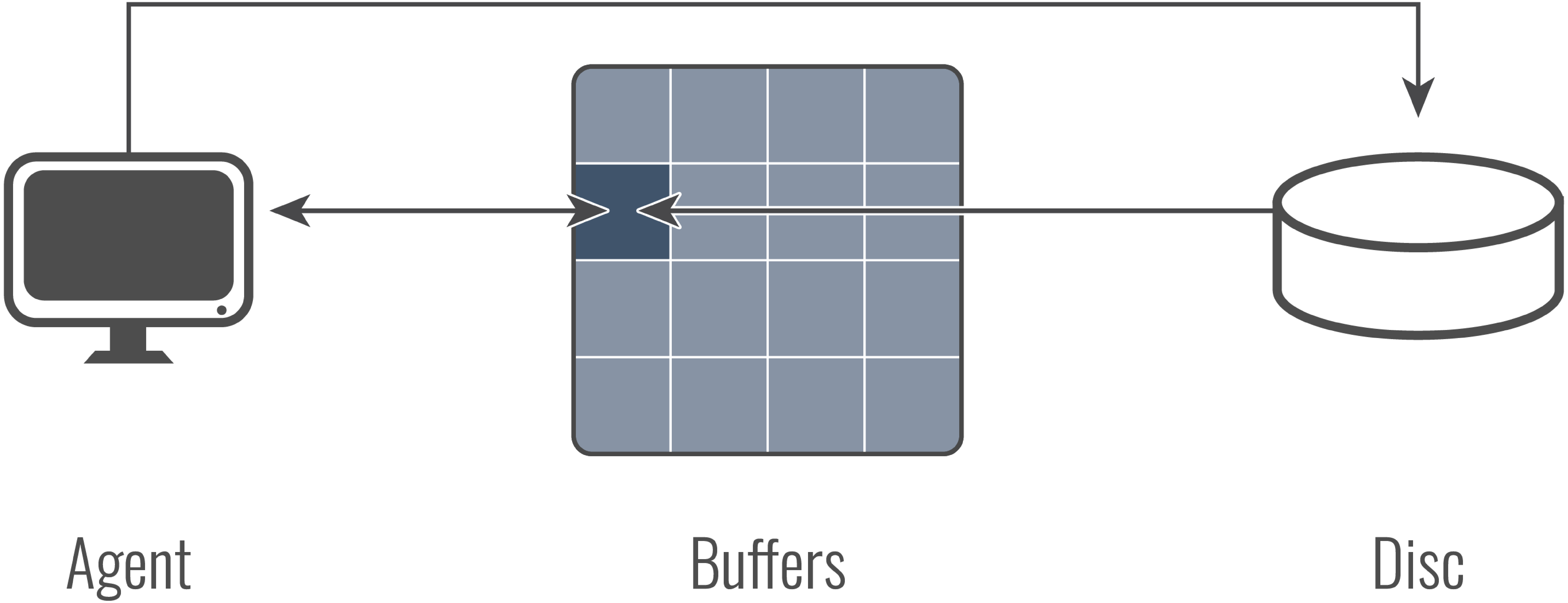
- **In memory use cases depend on workload!**

- **In this presentation will look at some examples**
- **Caches / Buffer Pools**
- **In-Memory Tables**
    - Shared In-Memory Tables
    - In-Memory Table Indexes
- **Fast Insert**
- **In-memory sorting**
- **Temporary Tables**
- **Large or small tables?**
- **Shared tables**

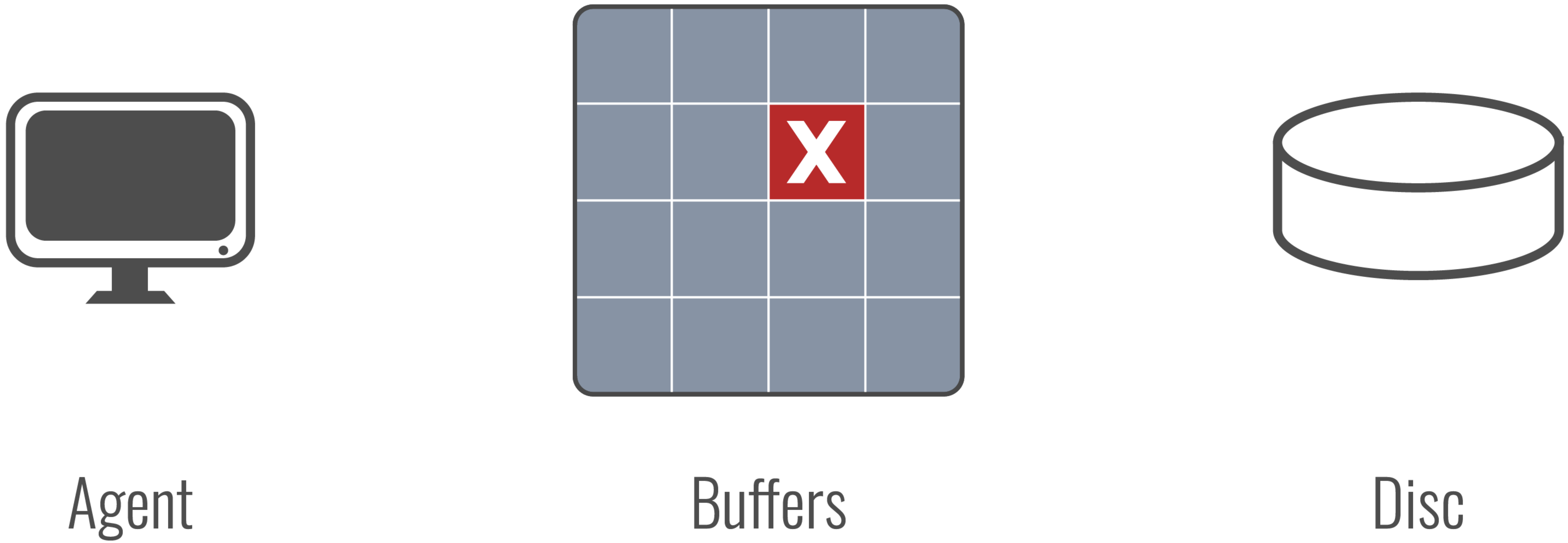
Technique: Buffer pools



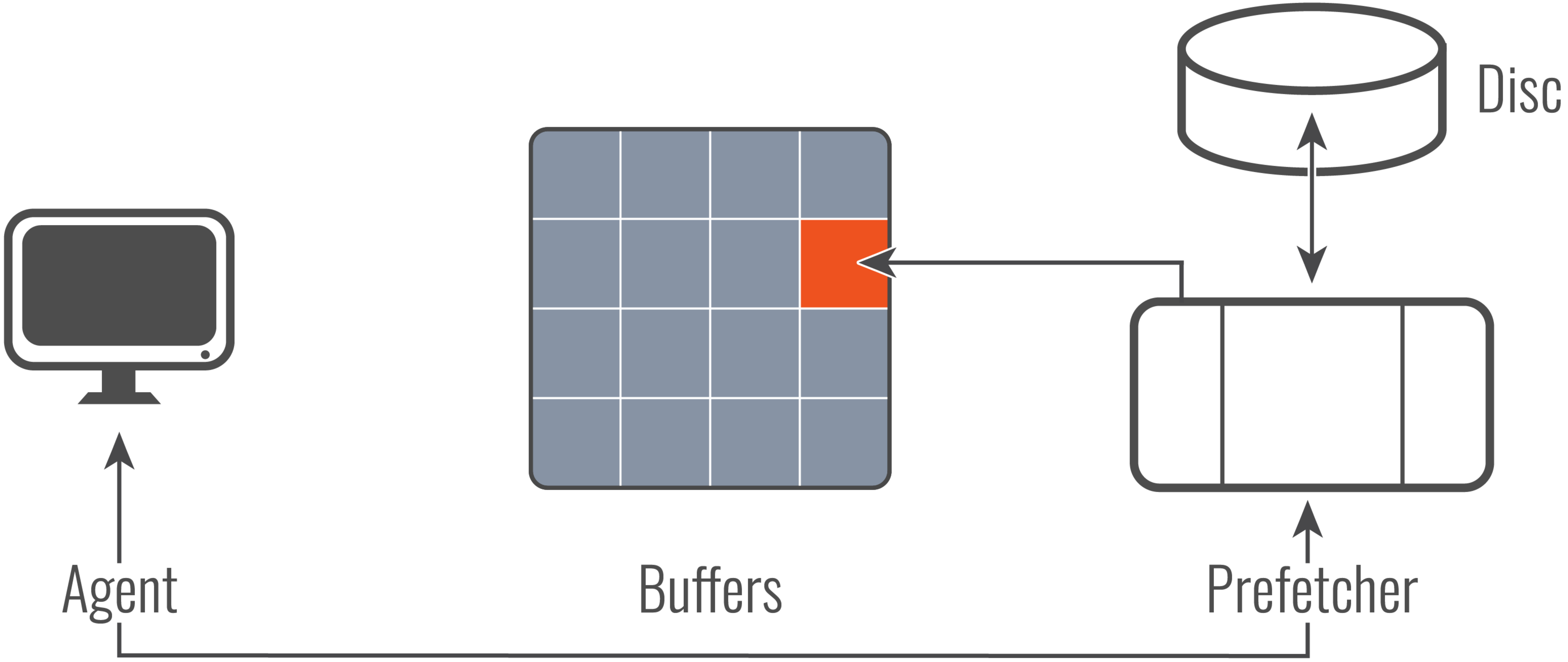
Buffer pools



Buffer pools invalidate



Buffer pools – Pre fetch



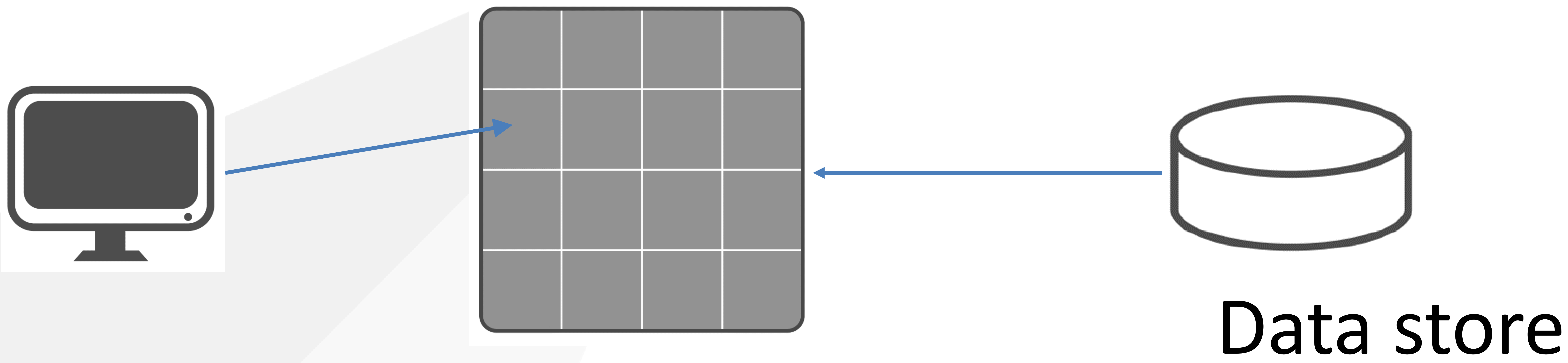
Same technique

DATAKINETICS
DATA PERFORMANCE & OPTIMIZATION

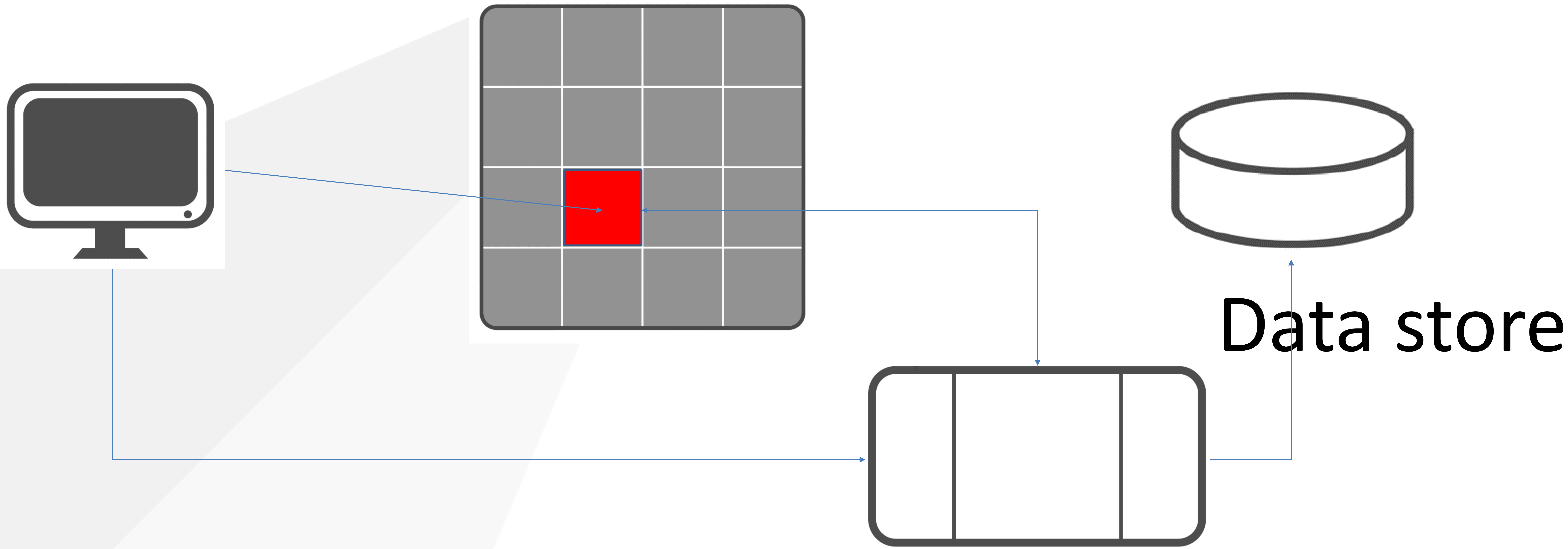
- DB2 Buffer Pools
- Package Cache
- Instruction Pipelines for CPUs
- Data Pipelines for CPU
- VSAM Buffers

Primary goal of this technique is to reduce I/O wait time (not CPU)

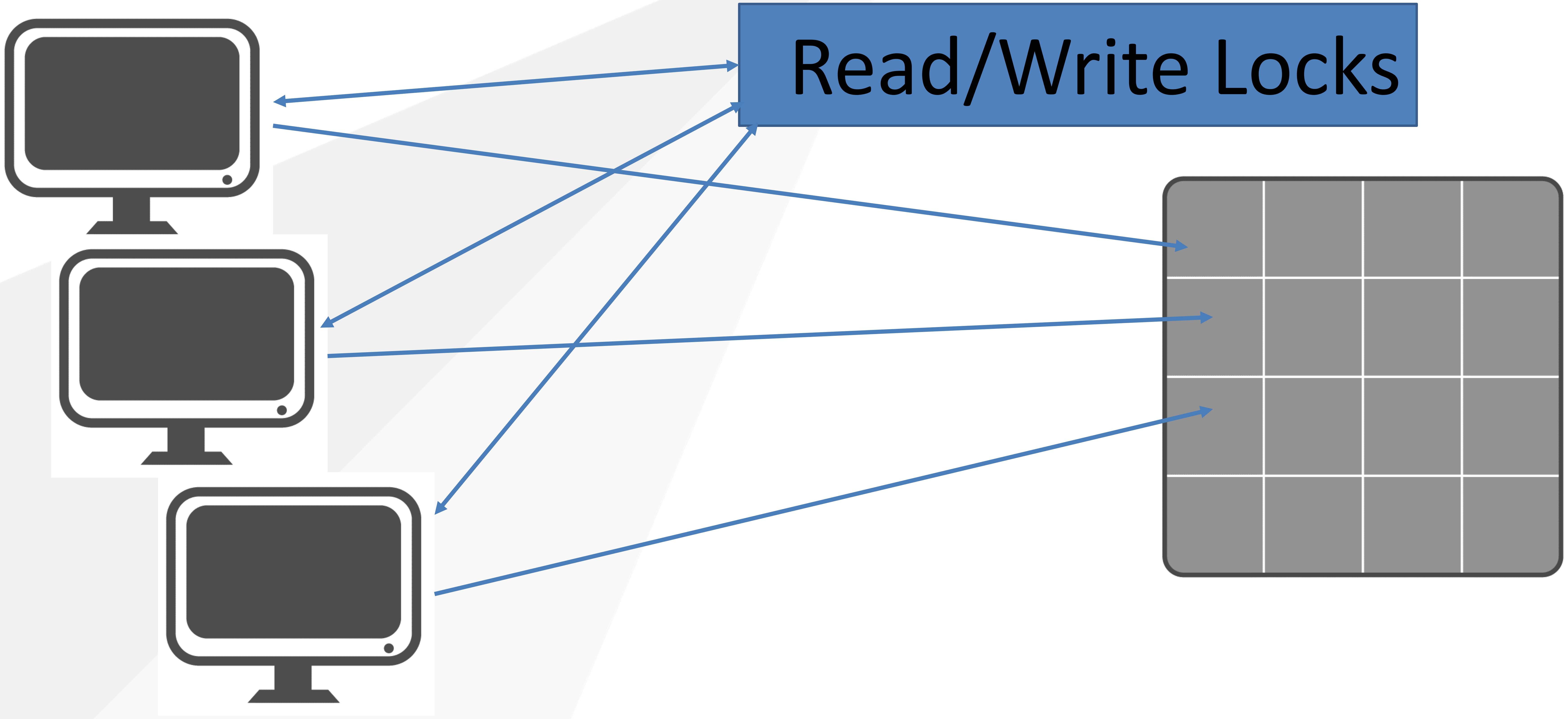
In Memory Tables



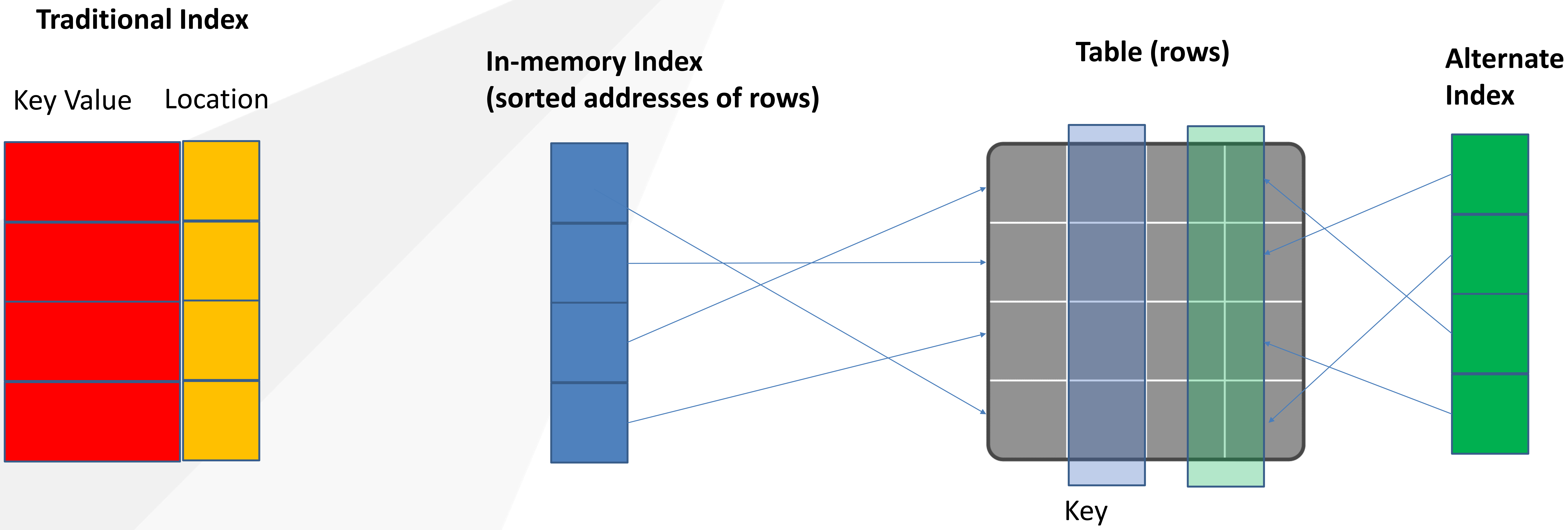
In Memory Tables



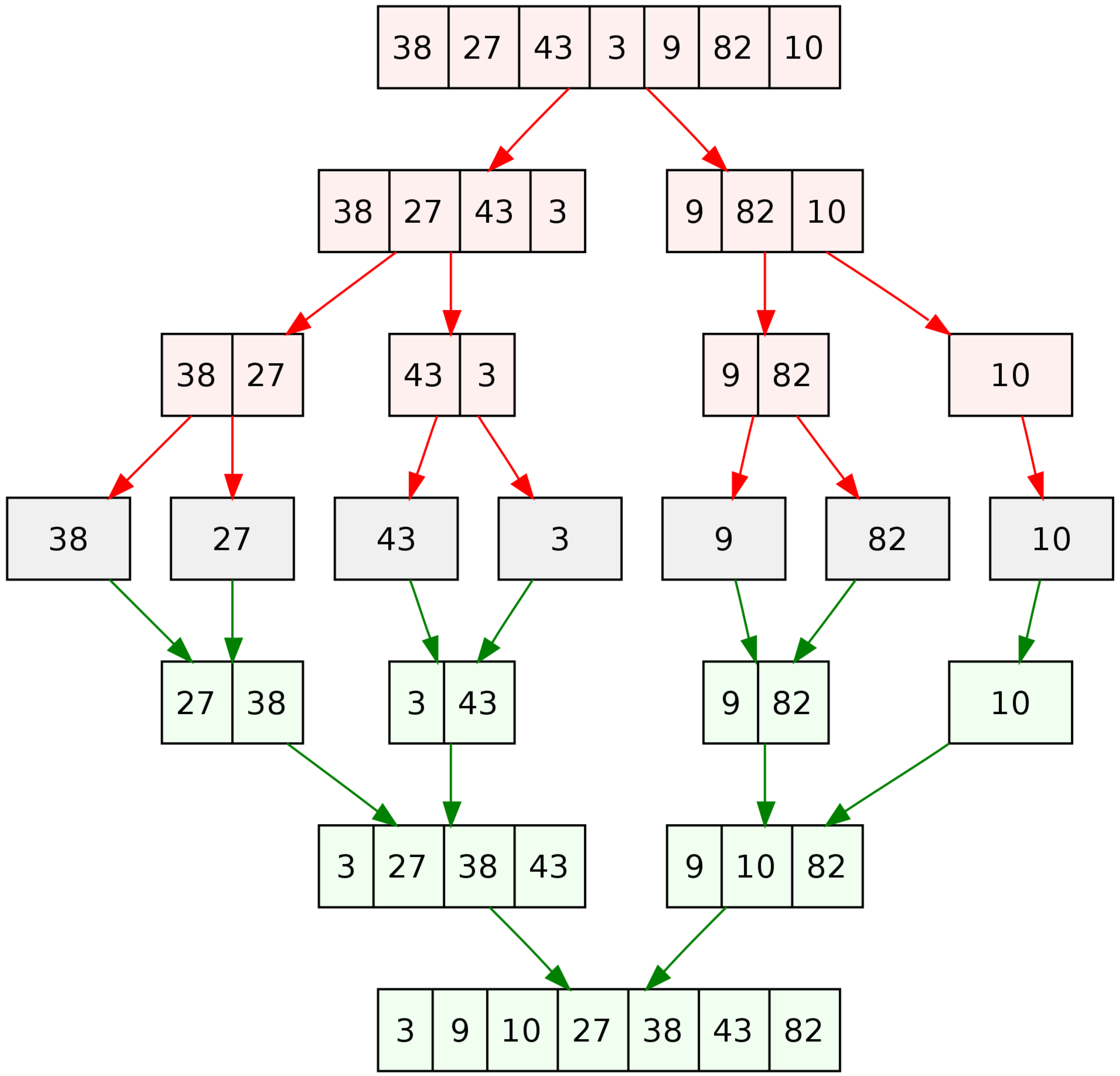
Shared In Memory Tables



In-Memory Table Indexes



In-Memory Sort



In-Memory Sort :DB2

- **Db2 v12 improved its RDS sort processing using more memory:**
- **Expanded the maximum number of nodes in a sort tree, from 32,000 to 512,000 for non-parallel sorts or 128,000 for parallel sorts under child tasks.**
- **These enhancements might require more memory to be allocated to the thread for sort activities, but can result in a significant CPU reduction.**

- **Requires the use of more memory – but ….**

Sort performance measurements (DB2 v12)

- **In-memory sorts that previously required work files for sort and merge processing**
  - 75% reduction in CPU time
- **Increased sort pool size**
  - 50% reduction in elapsed time and CPU time

Sort performance measurements (cont'd)

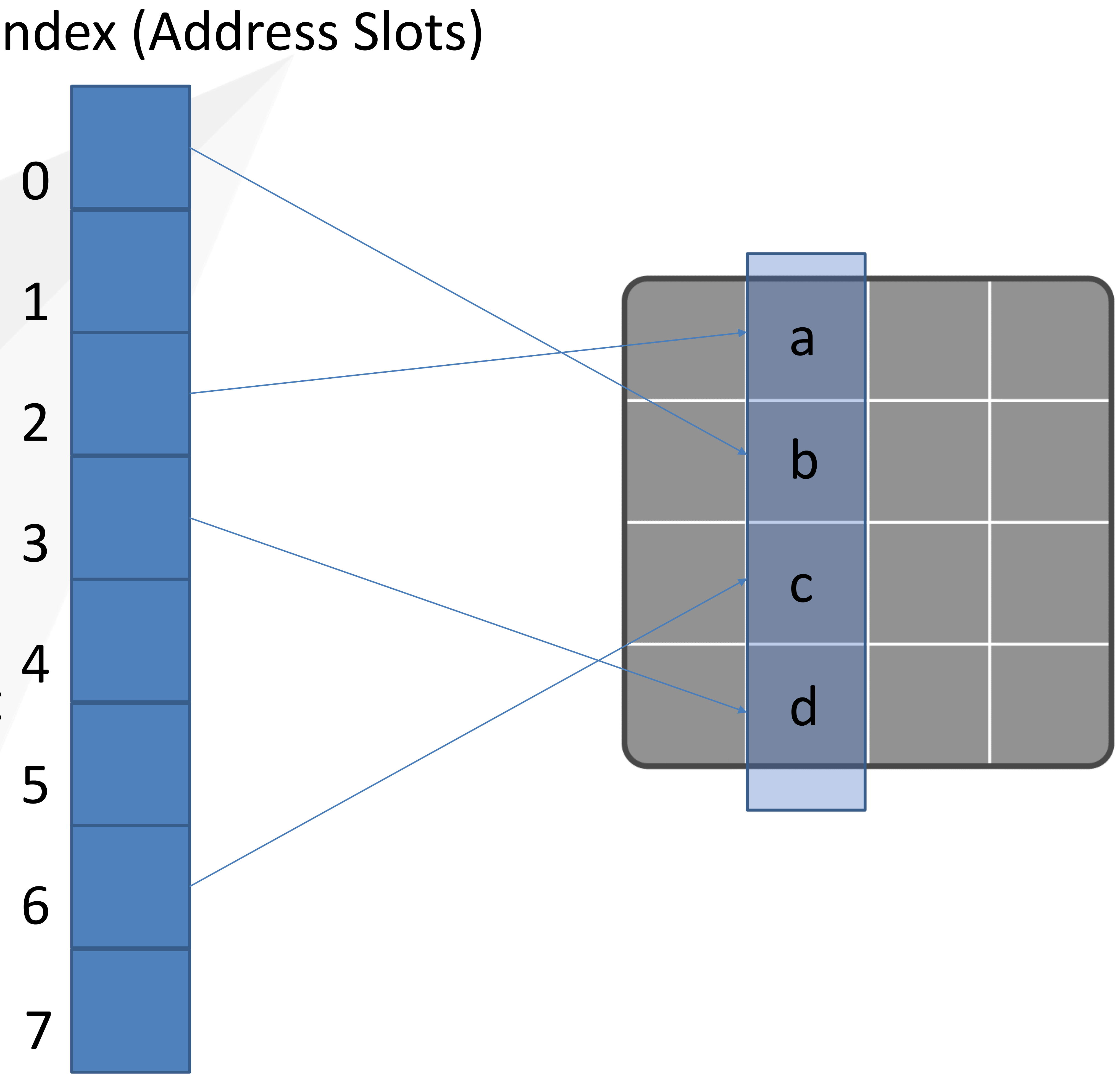
- **SAP workloads**
- **SAP CDS Fiori: 5% CPU time reduction for several queries**
  (1% CPU time reduction across the entire workload)
- **SAP CDS FINA: 1.8% reduction in CPU time for the entire workload**
  (12% reduction in the total number of GETPAGEs)

- **IBM Retail Data Warehouse**
- **Two queries: 14% and 6% CPU time reduction**

Slot calculated using Fn(Key)
Where Fn returns position 0..n

e.g. Fn(a) = 2

Empirically for In-Memory Indexes:
- Rows <10 – serial search fastest
- 10 < Rows <100 – binary search fastest
- Rows > 100 – Hash search

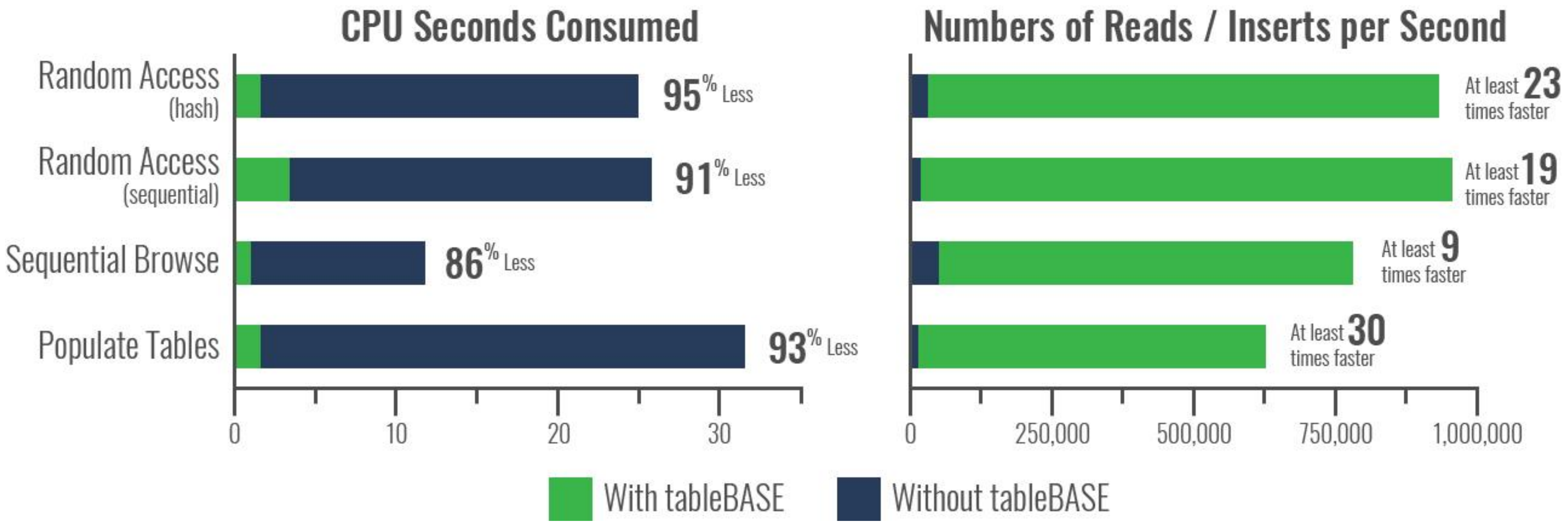


In-memory Table examples

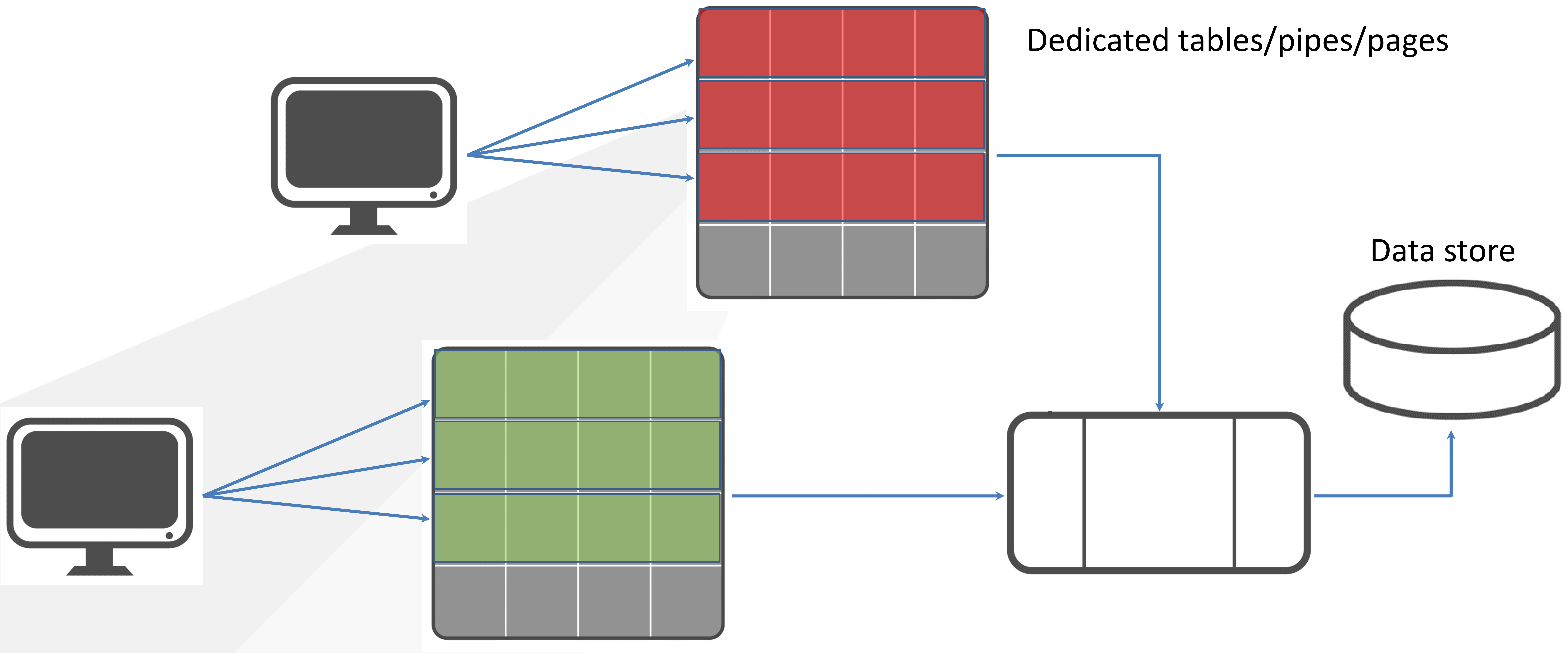
- **DB2 – In-memory table**
- **DB2 – Table fixed in buffer pools**
  - Structures still support on disc
- **Pure In-Memory Tables**
  - IBM IZTA
  - DKL tableBASE
- **Cobol Internal Tables (other languages too!)**
  - Limited to a primary index
  - Not Shareable
- **Home Grown In-Memory Accelerators**
  - Often from when people built their own everything

IBM Benchmark Results for Db2

- Two systems tested – one accessing data using Db2 with buffers, one accessing data using Db2 with tableBASE high-performance in-memory technology
- Improvements are made without changes to Db2 systems, and without changes to application logic



Fast Insert



Use cases for fast insert

- **High rate of concurrent INSERTs into a journal or audit table**
  - **Regulatory compliance**
  - **Access tracking**
  - **…..**
- **Challenges**
  - Indexing has to catch up – so immediate retrieval not possible
  - Keys must not conflict

DB2 v12 introduced a feature called Fast Insert 2

Temporary In-memory Tables

- **Leverage in-memory tables – no I/O**
- **Leverage fast insert – parallel write, no indexing**
- **Leverage in-memory indexes - fast to create**
- **Leverage in-memory sort – as part of building the indexes**

One customer's experience

Challenge

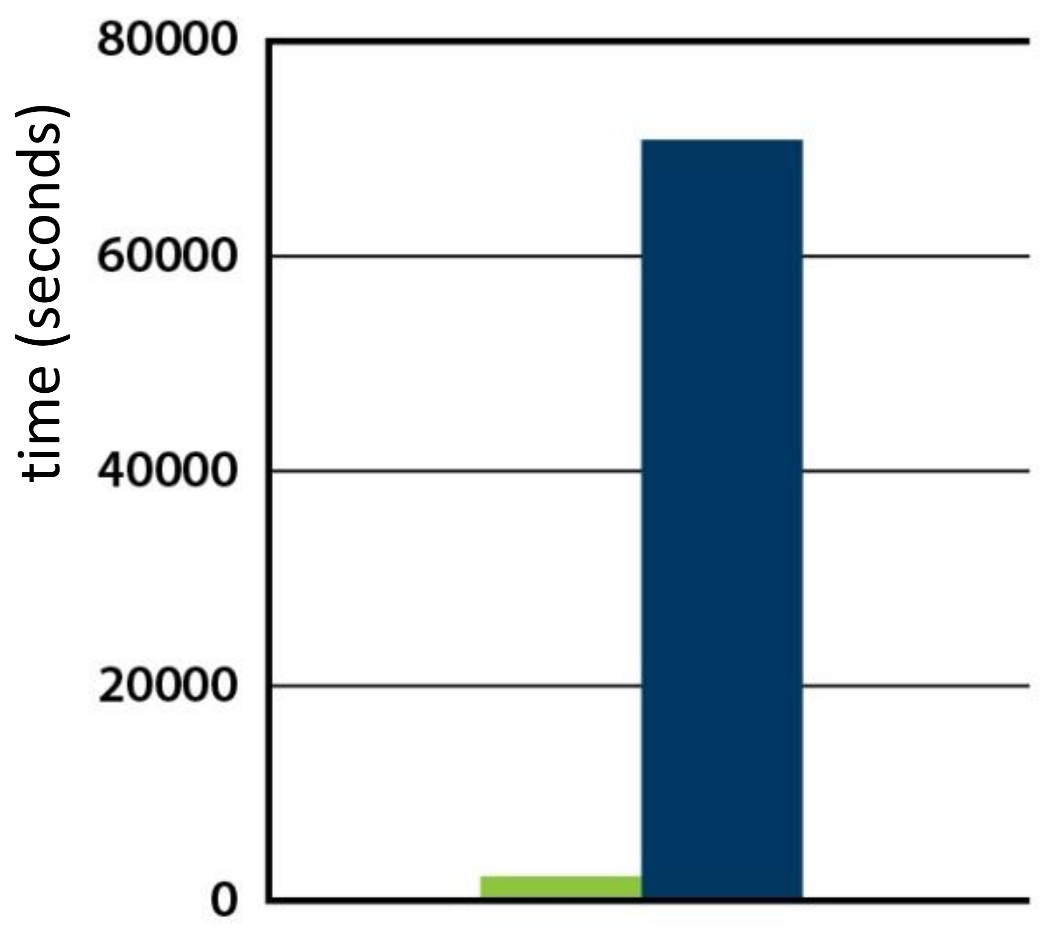
- A COBOL program was using an internal table and a binary search
- The search code was called 1.25 million times and had 4 searches in it
- Took over an hour of CPU to execute

Solution

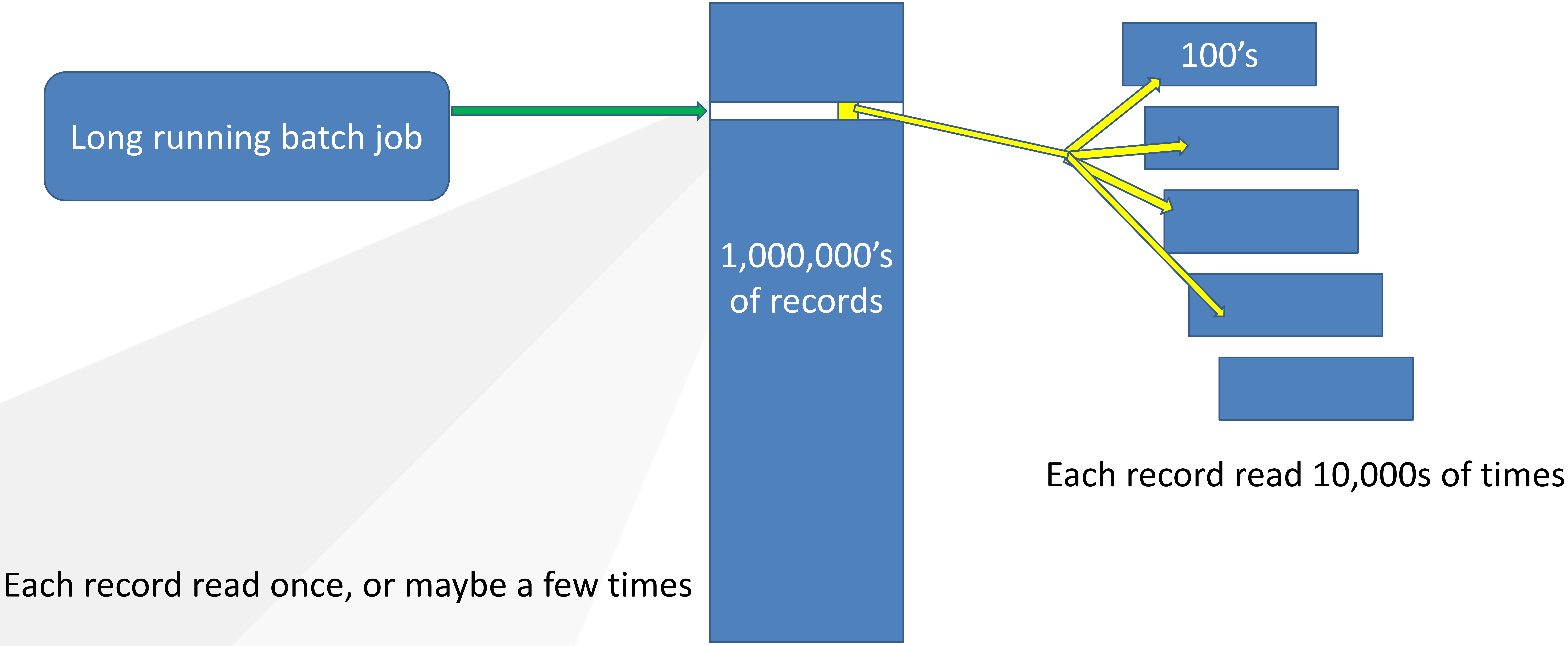
- Replace the 4 searches with calls to in-memory table with alternate indexes

Results

- 98% reduction in CPU required
- Now takes less than a minute to execute



Small or Large?



Results From Credit Card Processing

DATAKINETICS
DATA PERFORMANCE & OPTIMIZATION

Challenge

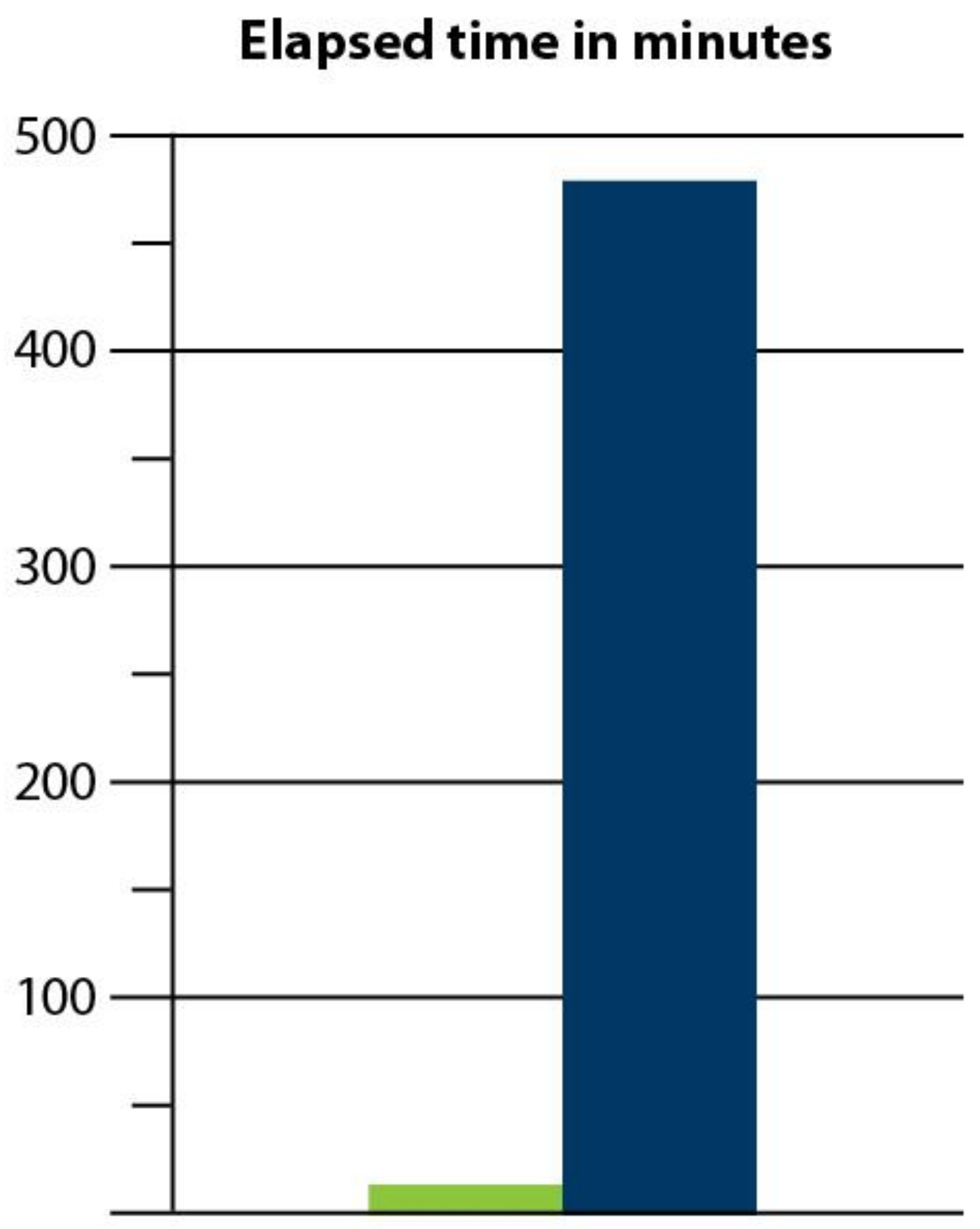
- **Reconciliation batch processing taking too long**

Solution

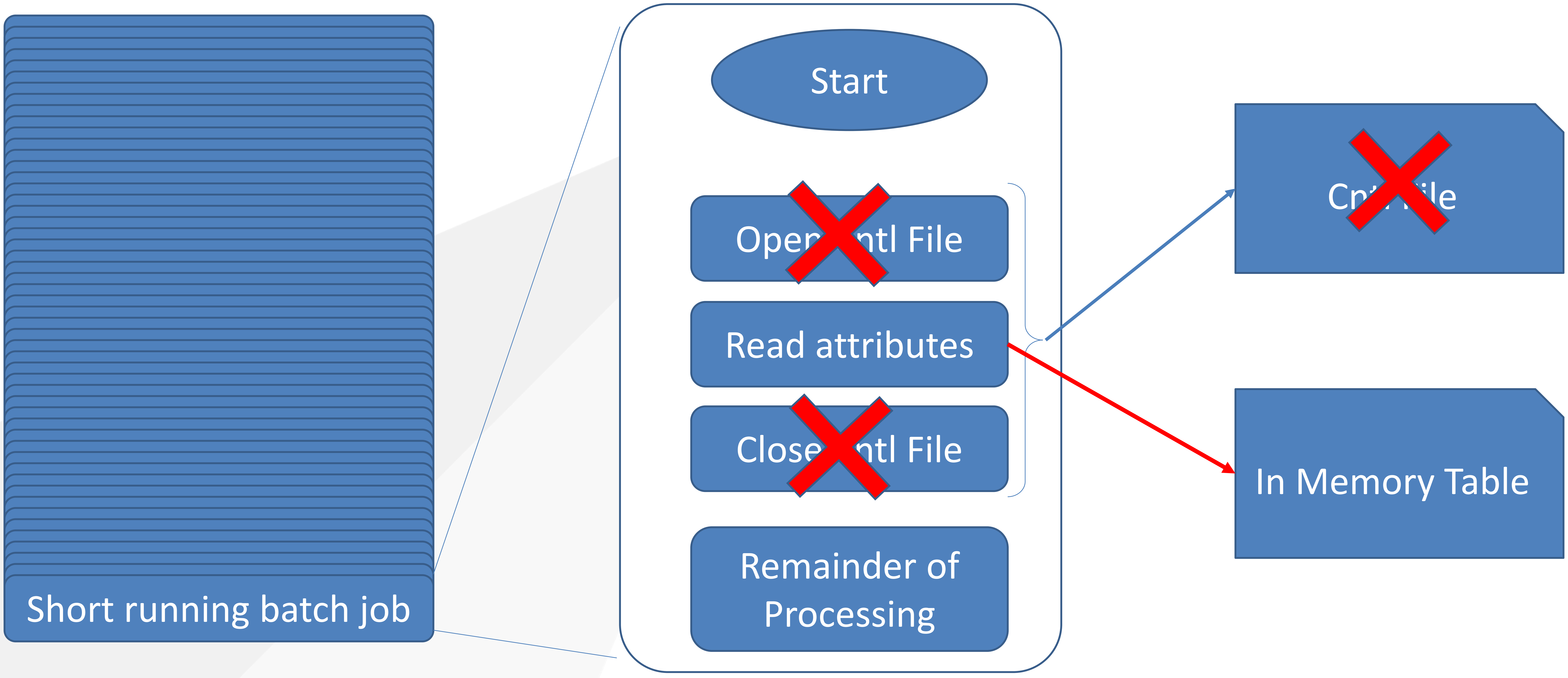
- **Move a table describing the credit card options into memory**
- **Each transaction required data from that table**

Results

- **97% reduction in elapsed time**
- **Batch job that took 8 hours to complete now takes 15 min**



Frequently opened VSAM



Customer Example

VSAM file opened/read/closed by very frequently running batch job

Moved file to sharable in-memory table

- 75% less CPU for reads
- 100% less CPU for open and close!

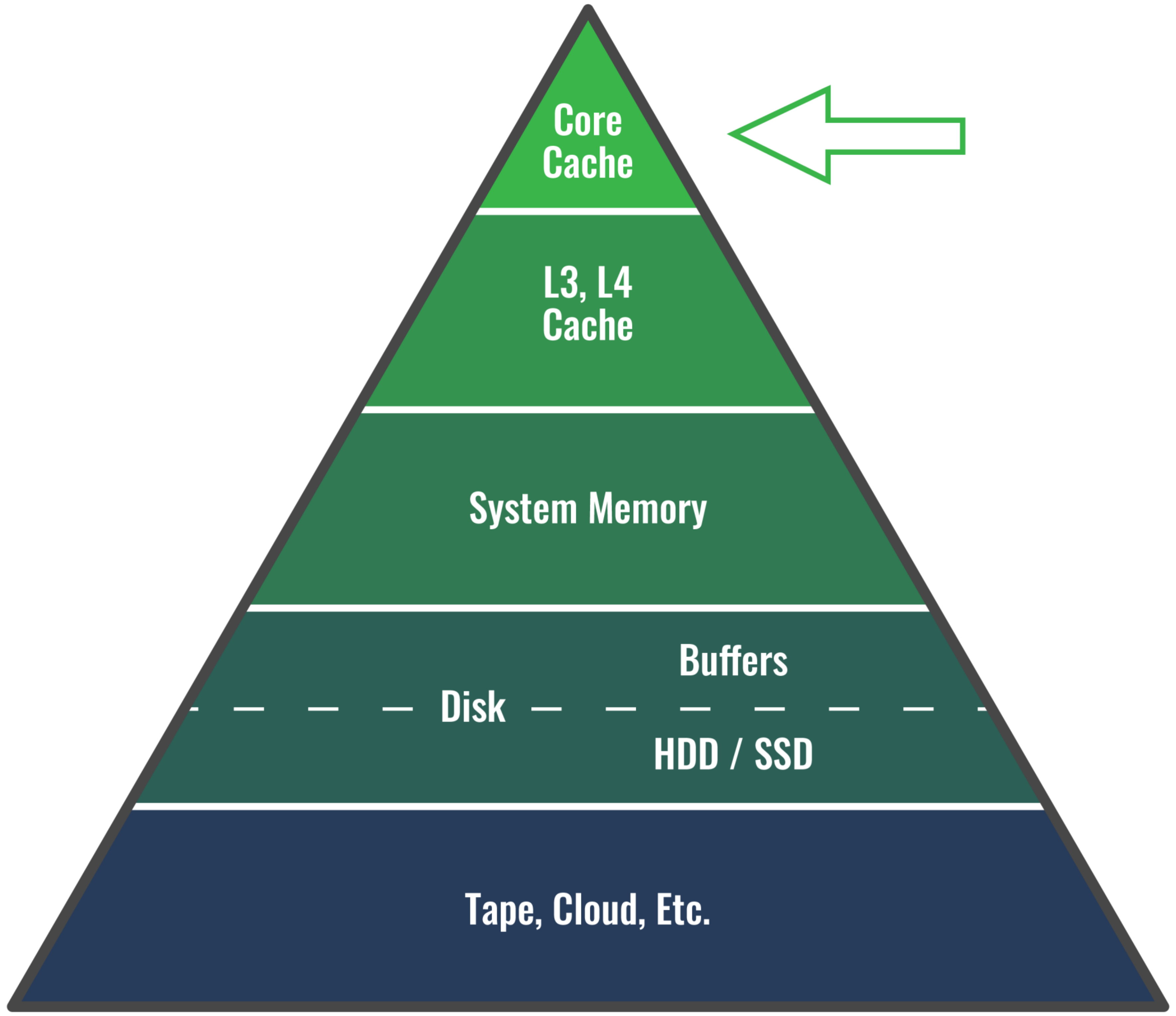
Results

- 98% reduction in elapsed time
- 93% reduction in CPU required
- More than 24 hours of CPU saved daily!

Memory usage on the mainframe

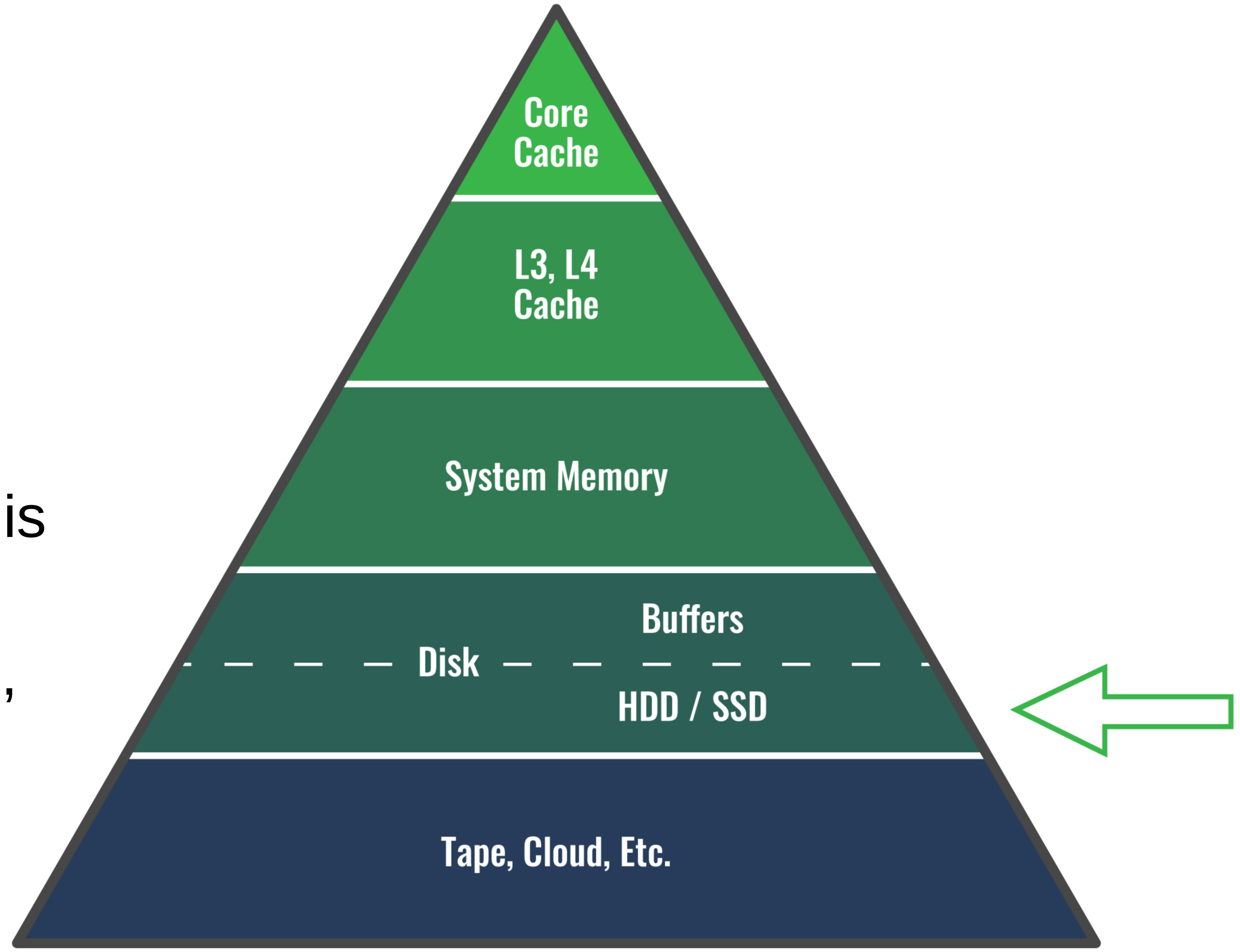
DATAKINETICS
DATA PERFORMANCE & OPTIMIZATION

- **The cache memory…**
  - The most powerful processor chips today (including the z16) have layers of on-chip and on-board cache in the form of eDRAM and SRAM.
  - Much of your data winds up here at some point, but it is all controlled by the system.
  - There's not much differentiation here other than the newer chips typically have more/faster cache.



Memory usage on the mainframe

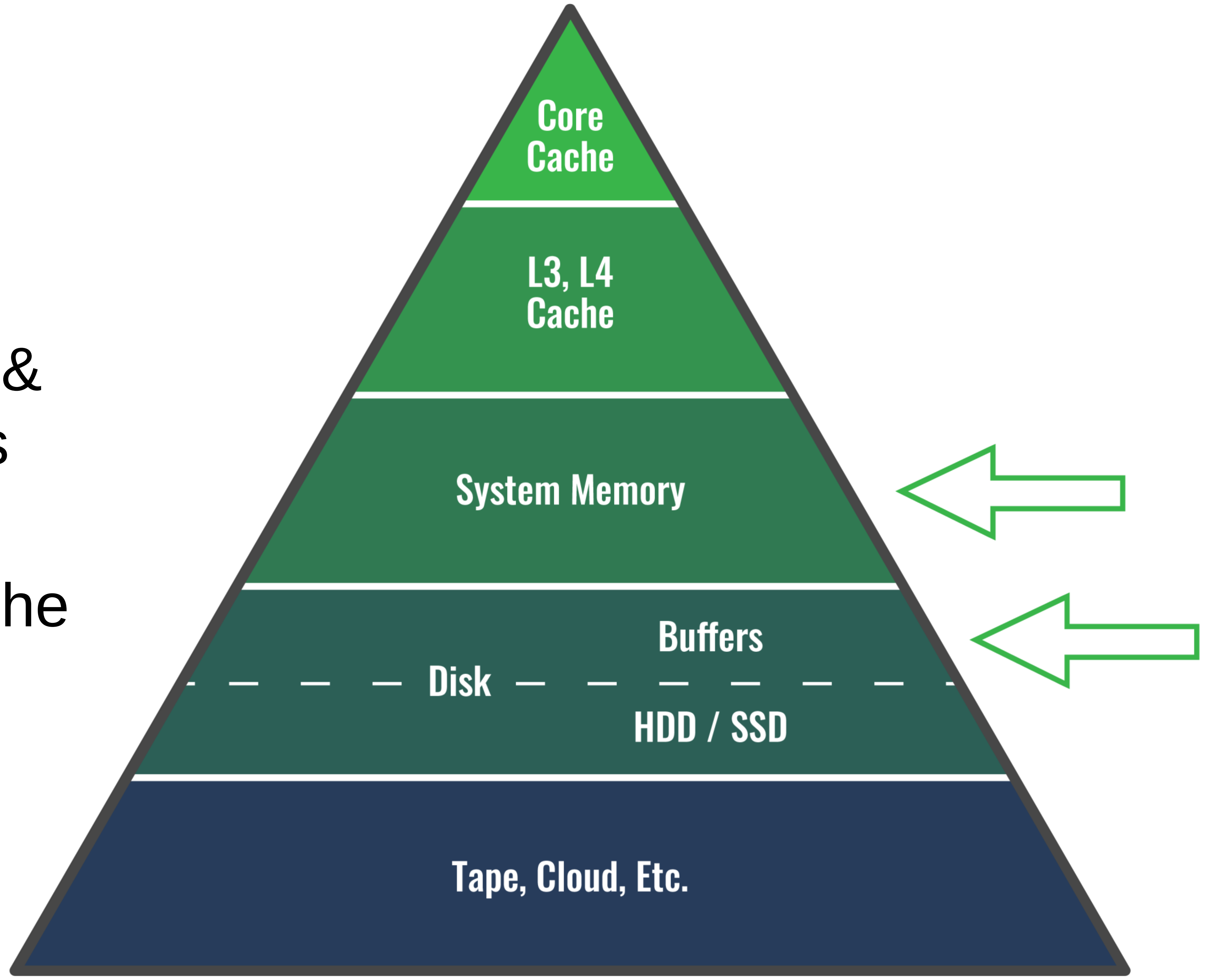
- **Your database**
  - Your database is where your enterprise data lives, and where your applications go to get data.
  - Typically, this activity requires lots of I/O disk access-this is the baseline for how fast you access data.
  - But since memory is about 1000x faster than disk access, we try to use that whenever possible.



Memory usage on the mainframe

DATAKINETICS
DATA PERFORMANCE & OPTIMIZATION

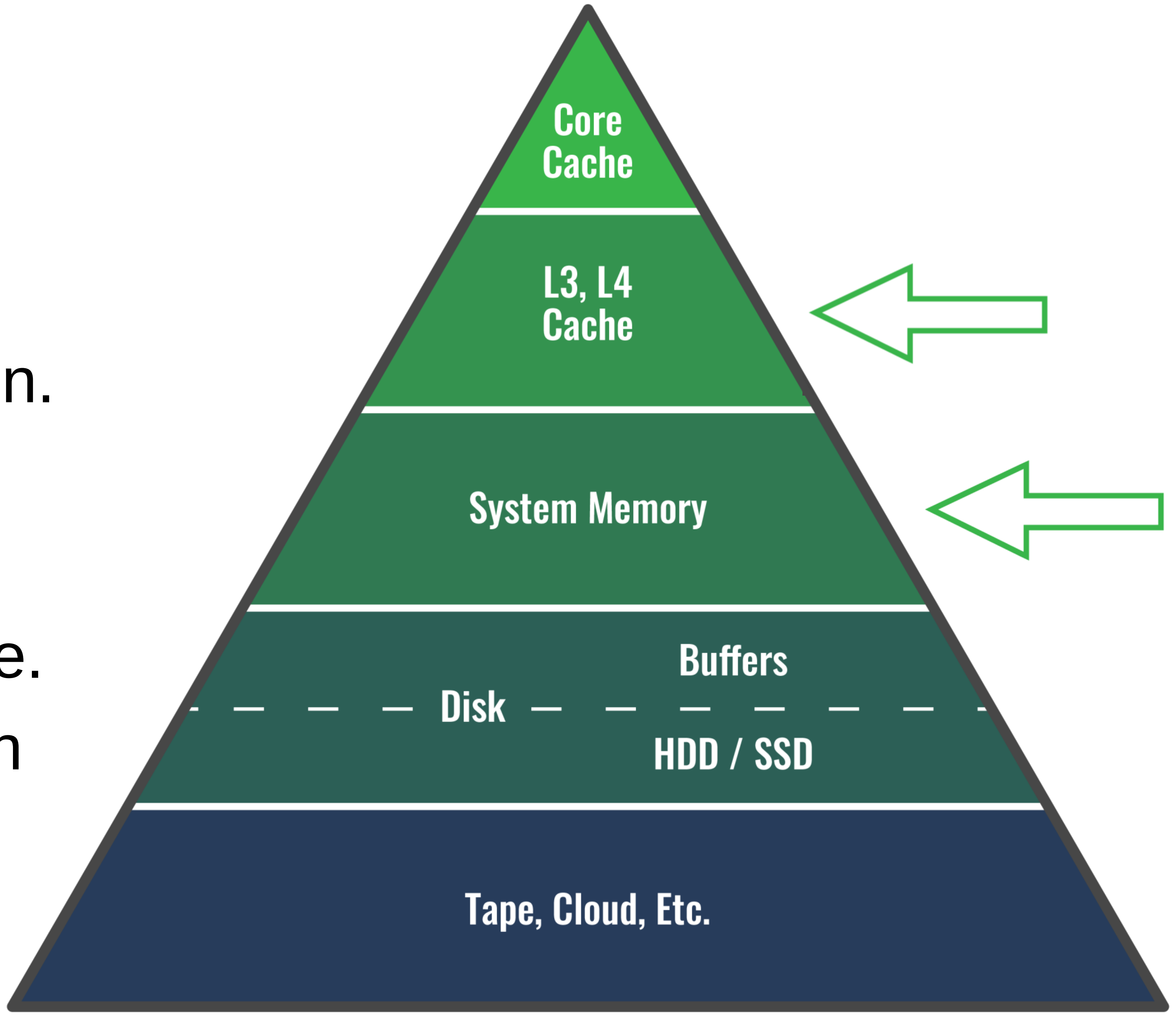
- **Buffering…**
  - DBMS buffers use main memory to cut out I/O for recent disk access.
  - They make a big difference in reducing data access time & processing time. (Which can translate to lower operations cost as well)
  - (There are even third-party buffer tools and database cache solutions that help improve buffer efficiency even more.)
  - Buffered DASD accesses data up to 10 times faster than non-buffered data.
  - Did you know that you can augment this performance?



Memory usage on the mainframe

DATAKINETICS
DATA PERFORMANCE & OPTIMIZATION

- **Mainframe high-performance in-memory technology...**
  - Shortens the code path for the data you access most often.
  - Augments your buffered database, using main memory.
  - Accesses data faster than buffer performance.
  - Requires no changes to application logic or your database.
  - If in-memory tables are small enough, and accessed often enough, they can make it into the L3-L4 cache for ultra-fast processing



What about IDAA?

- **IDAA is fantastic at reducing long running queries (by using parallelism)**
  - Queries are not run often
- **In-memory tables are best for reducing very short running queries**
  - Need many queries before the difference is noticeable.

Summary

In-Memory Table techniques that can improve performance of specific workloads:

- Buffers (or cache)
- In-Memory Tables
- In-Memory Indexes (In-memory Sorts, address only changes, Hash....)
- Fast Inserts
- Temporary Tables (leveraging multiple aspects)
- Small tables
- Shared tables

Q&A

Q&A

DATAKINETICS
DATA PERFORMANCE & OPTIMIZATION

Thank you for your time.

Larry Strickland

Chief Product Officer

613 523 5500 ext 256

lstrickland@dkl.com