IBM Research

Increasing Buffer-Locality for Multiple Index Based Scans through Intelligent Placement and Index Scan Speed Control

Christian A. Lang
Bishwaranjan Bhattacharjee
Tim Malkemus

Database Research Group
IBM T.J. Watson Research Center

Kwai Wong

IBM Toronto Lab

Goal

Improve query performance (throughput+latency)
for
ad-hoc index scan-heavy multi-query workloads
(e.g., DSS workloads)
with minimal architecture dependency/impact

Example DSS Queries

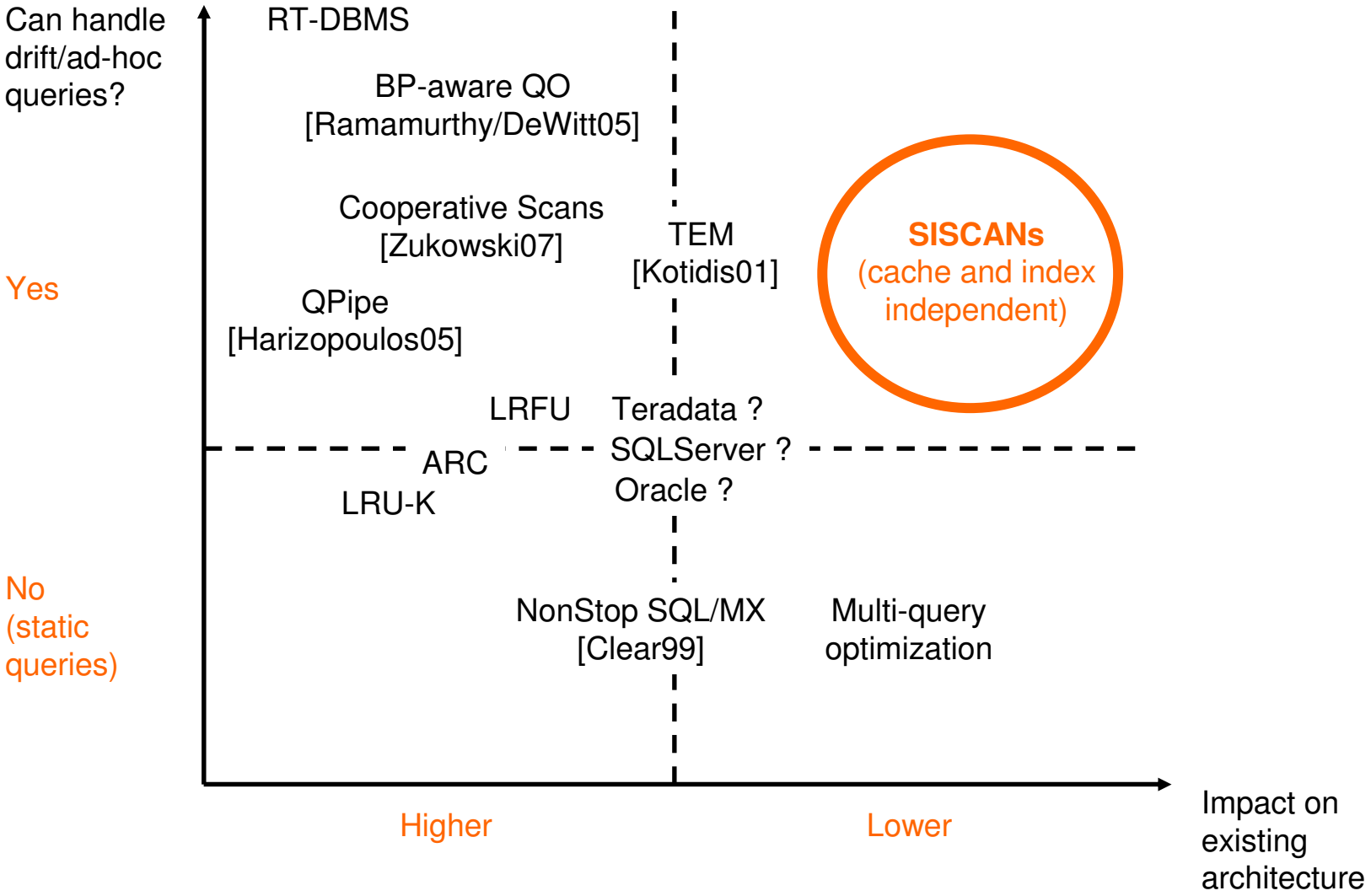
- select sum(l_extendedprice*l_discount) as revenue
  from lineitem
  where l_shipdate >= ’01/01/2006’ and
  l_shipdate < ’01/01/2006' + interval '1' year
  and l_quantity > 10;

- select sum(l_extendedprice*l_discount) as revenue,
  avg(l_extendedprice*l_discount) as avgSale
  from lineitem
  where l_shipdate >= ’10/01/2006’ and
  l_shipdate < ’10/01/2006’ + interval ‘3' month
  and l_quantity > 30;

Challenges in Multi-Query DSS Workloads

- DSS workloads include scanning of large amounts of data (e.g., aggregate calculation)
- Cannot optimize ahead of time (many ad-hoc queries, unknown start times)
- Trend: even more I/O bound queries (disk seek/access times not keeping up with capacity growth/CPU speed)
- Sub-optimal cache reuse (current RDBMS treat queries (mostly) in isolation)

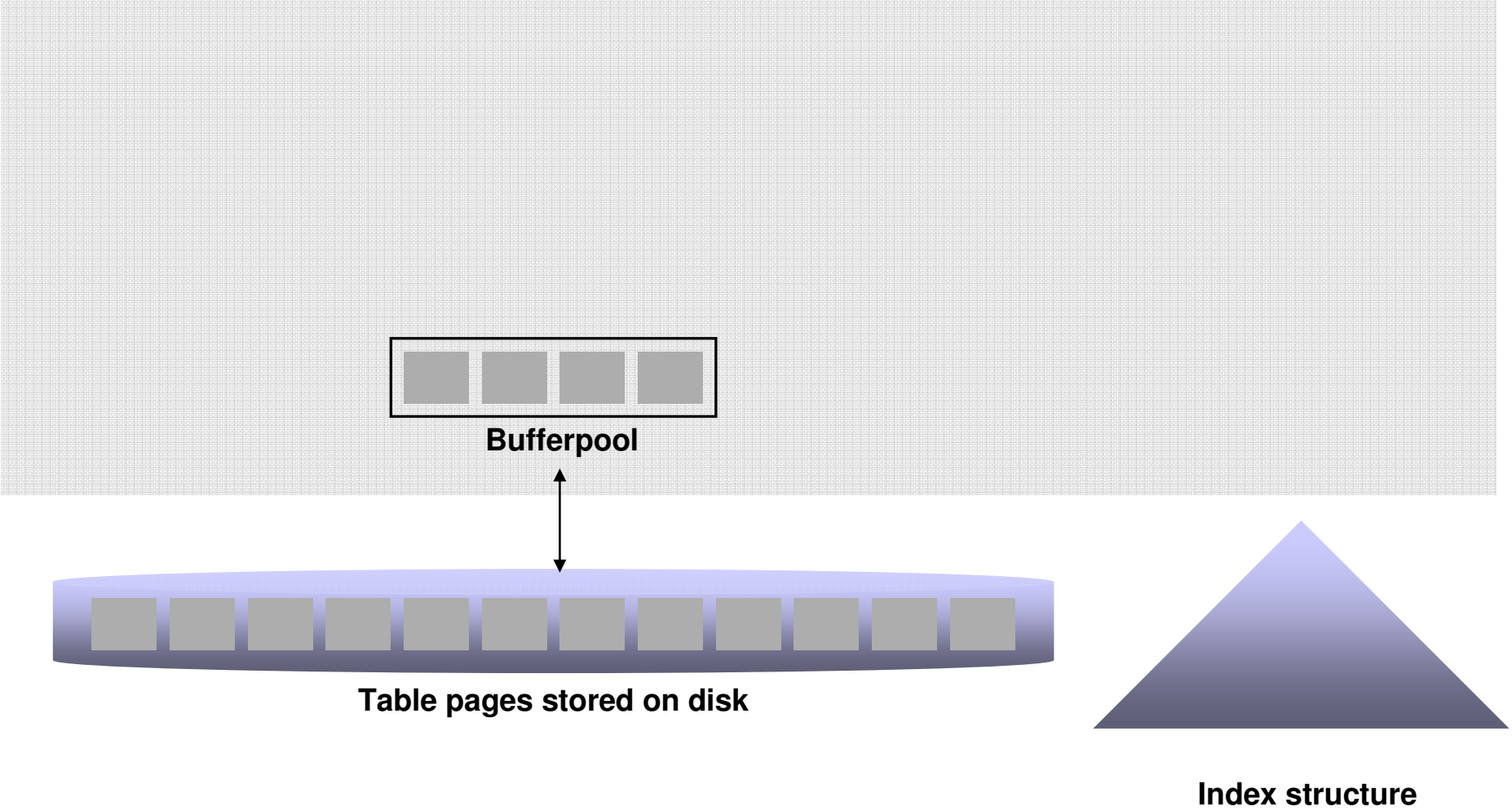
Known Solutions (not to scale)



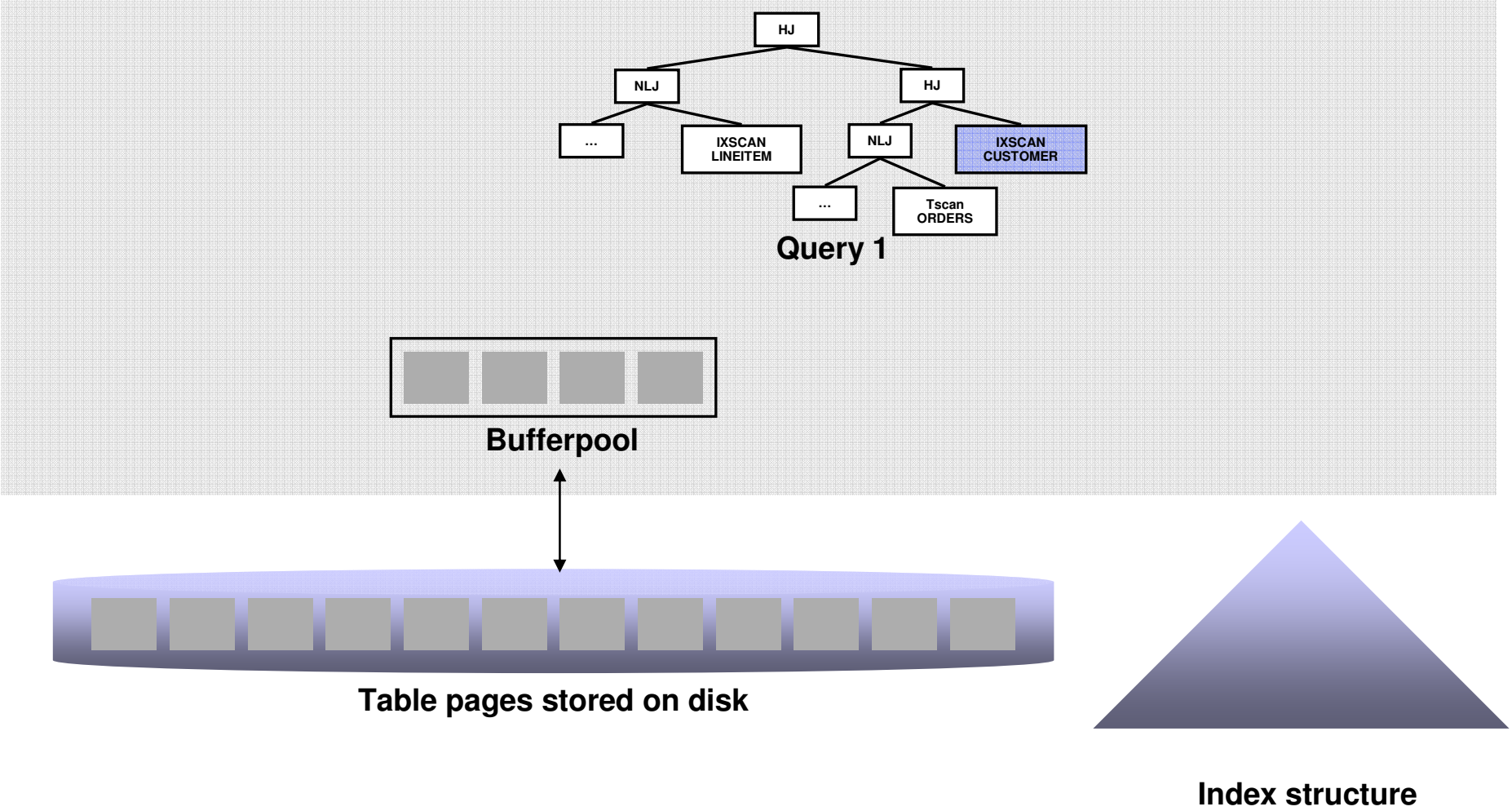
Outline

- **Current Index Scan Architecture**
- **SISCAN – "Circular" Index Scan**
  - **Placement**
  - **Speed Control**
- **Implementation Issues**
  - **Index-independent Relative SISCAN Location**
  - **"Bufferpool-independent" SISCAN-aware Caching**
- **Experimental Results**
- **Conclusions**

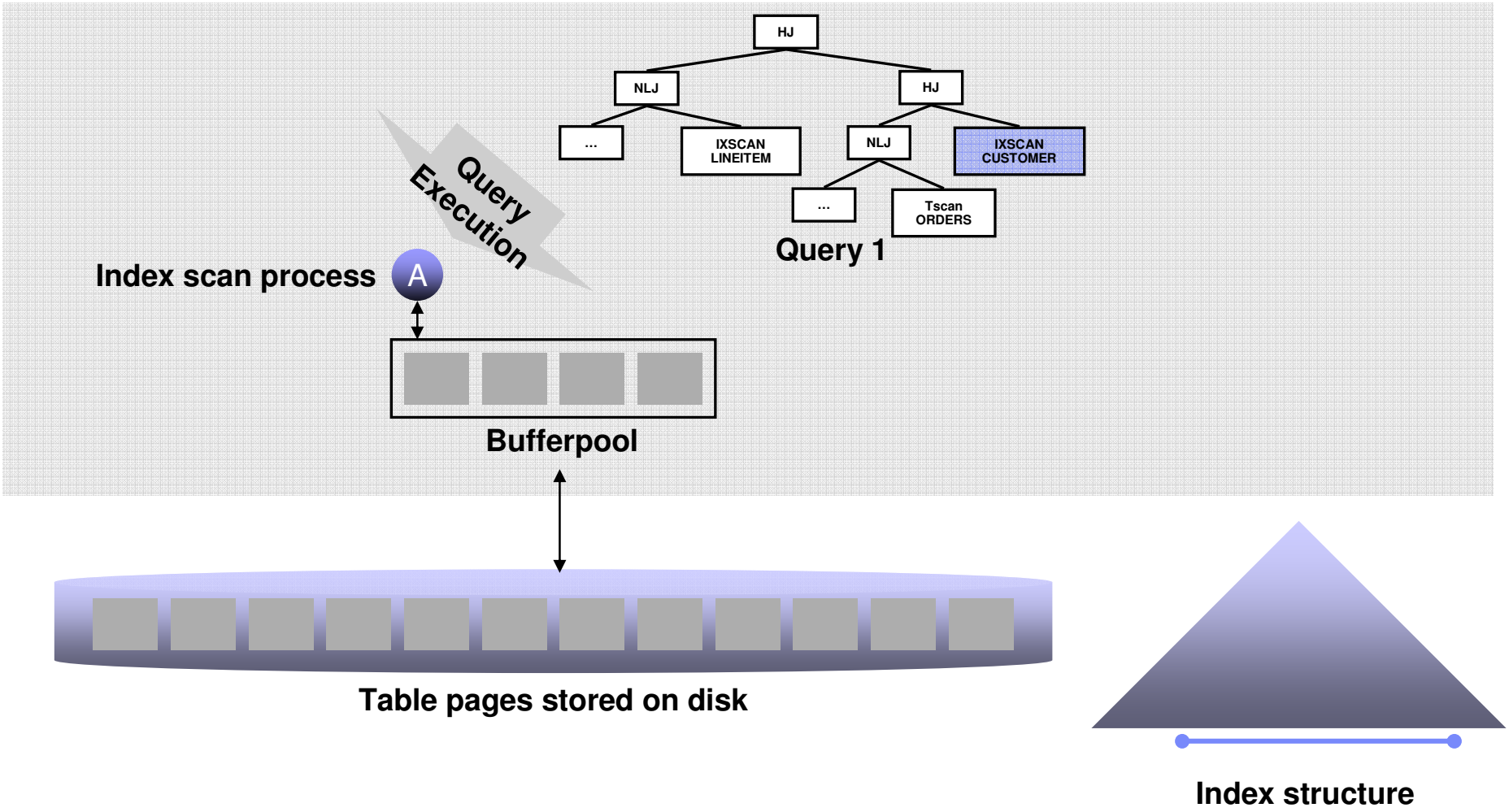
Current Index Scan Architecture



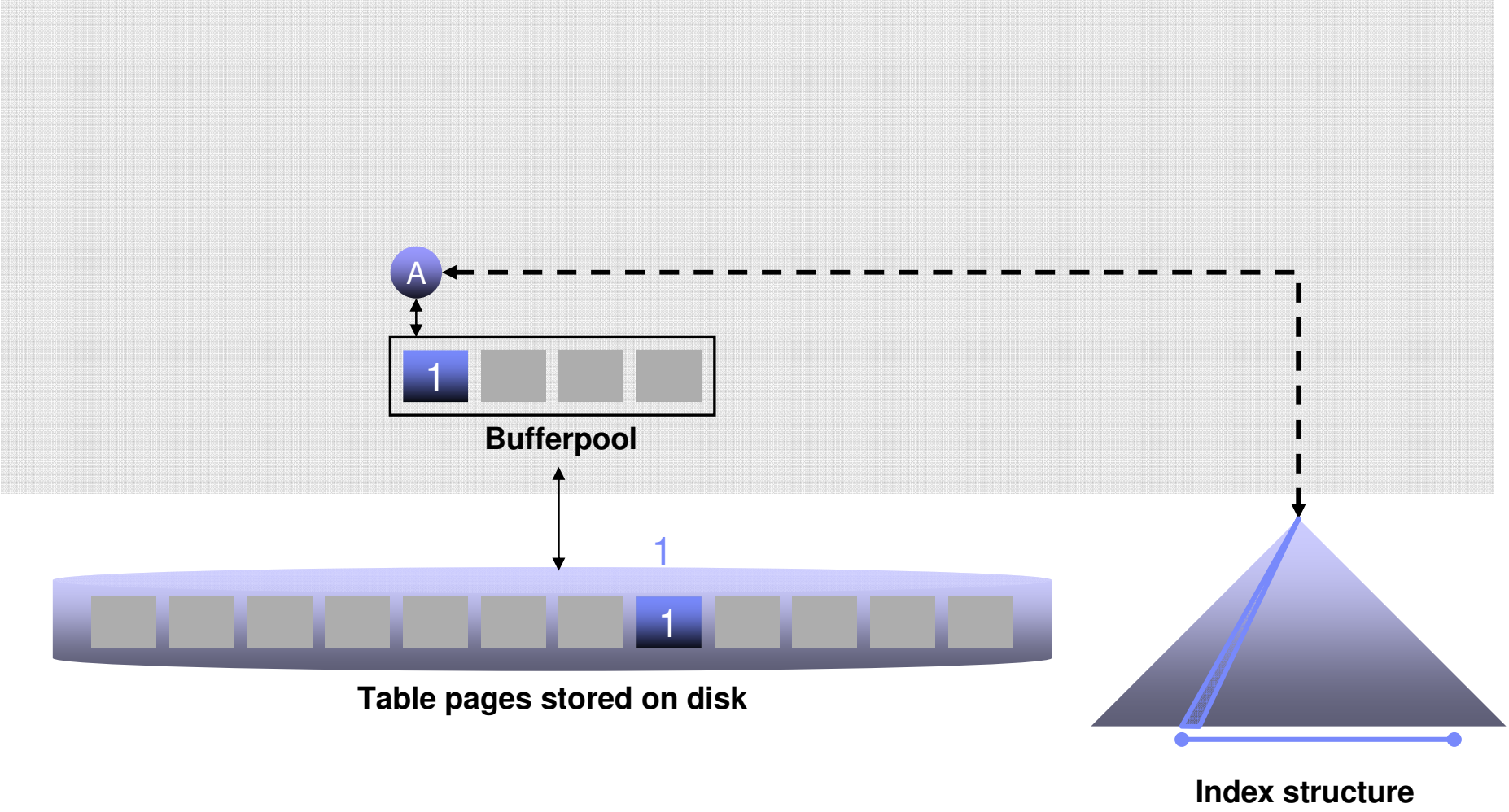
Current Index Scan Architecture



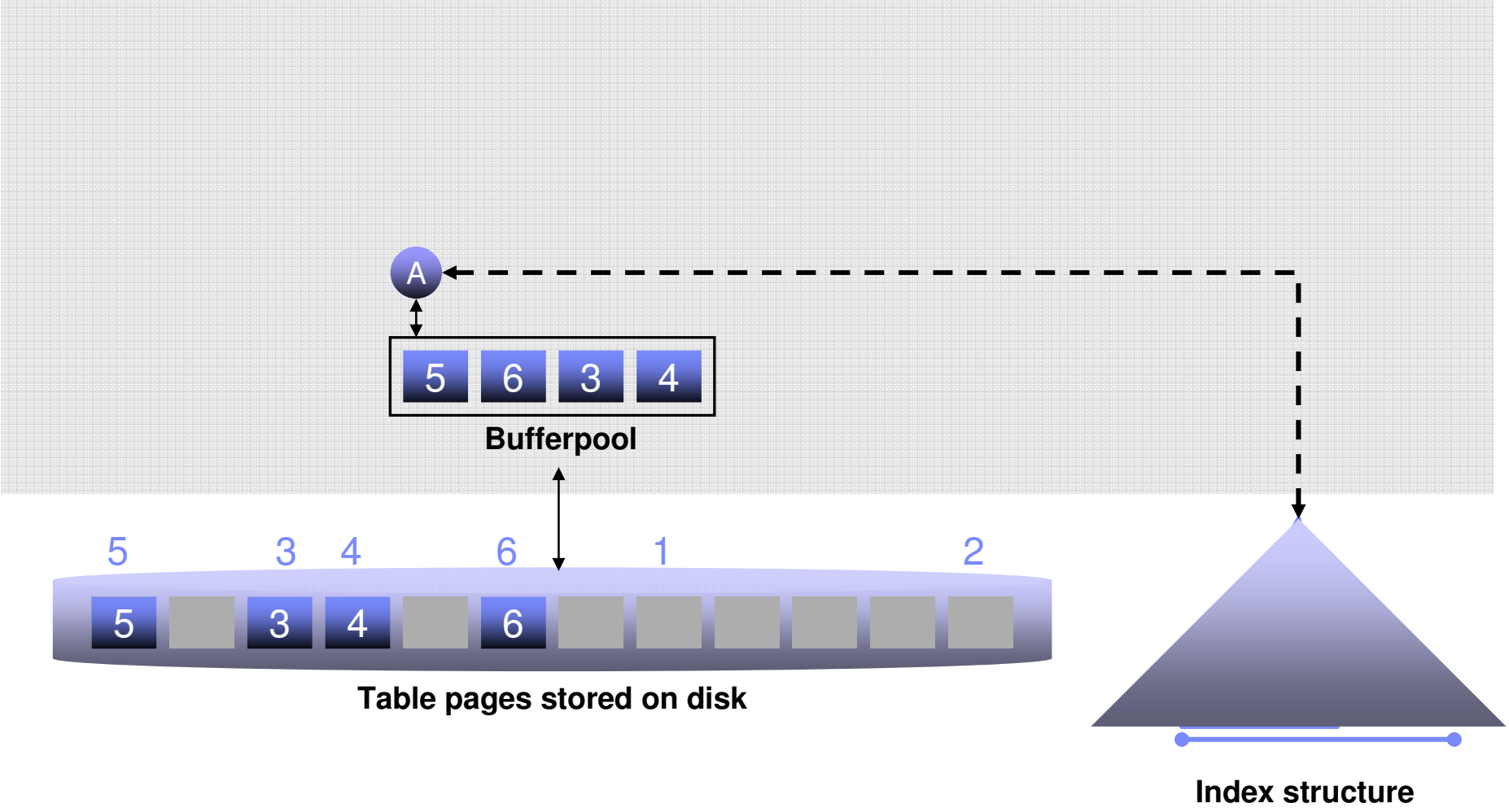
Current Index Scan Architecture



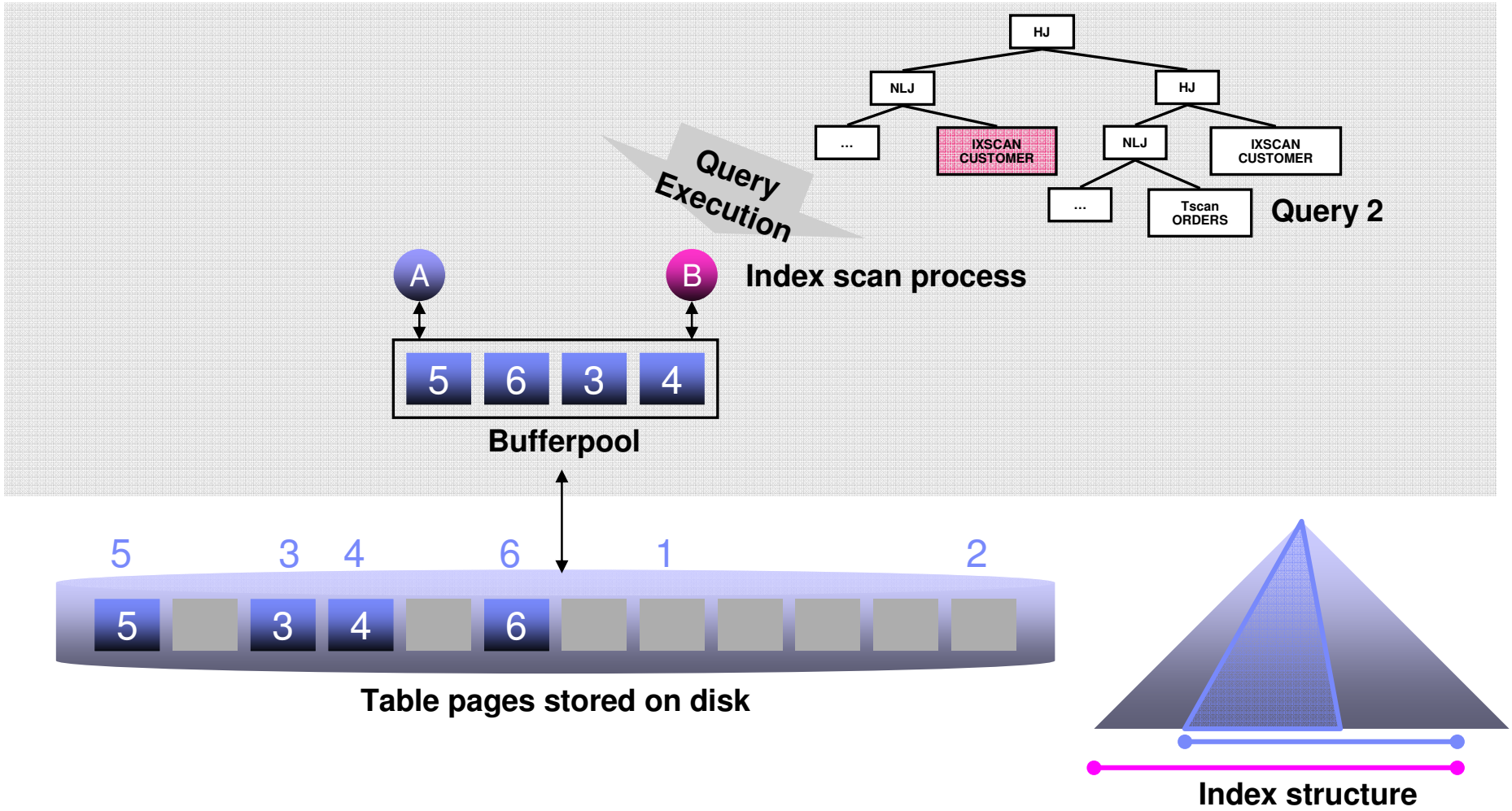
Current Index Scan Architecture



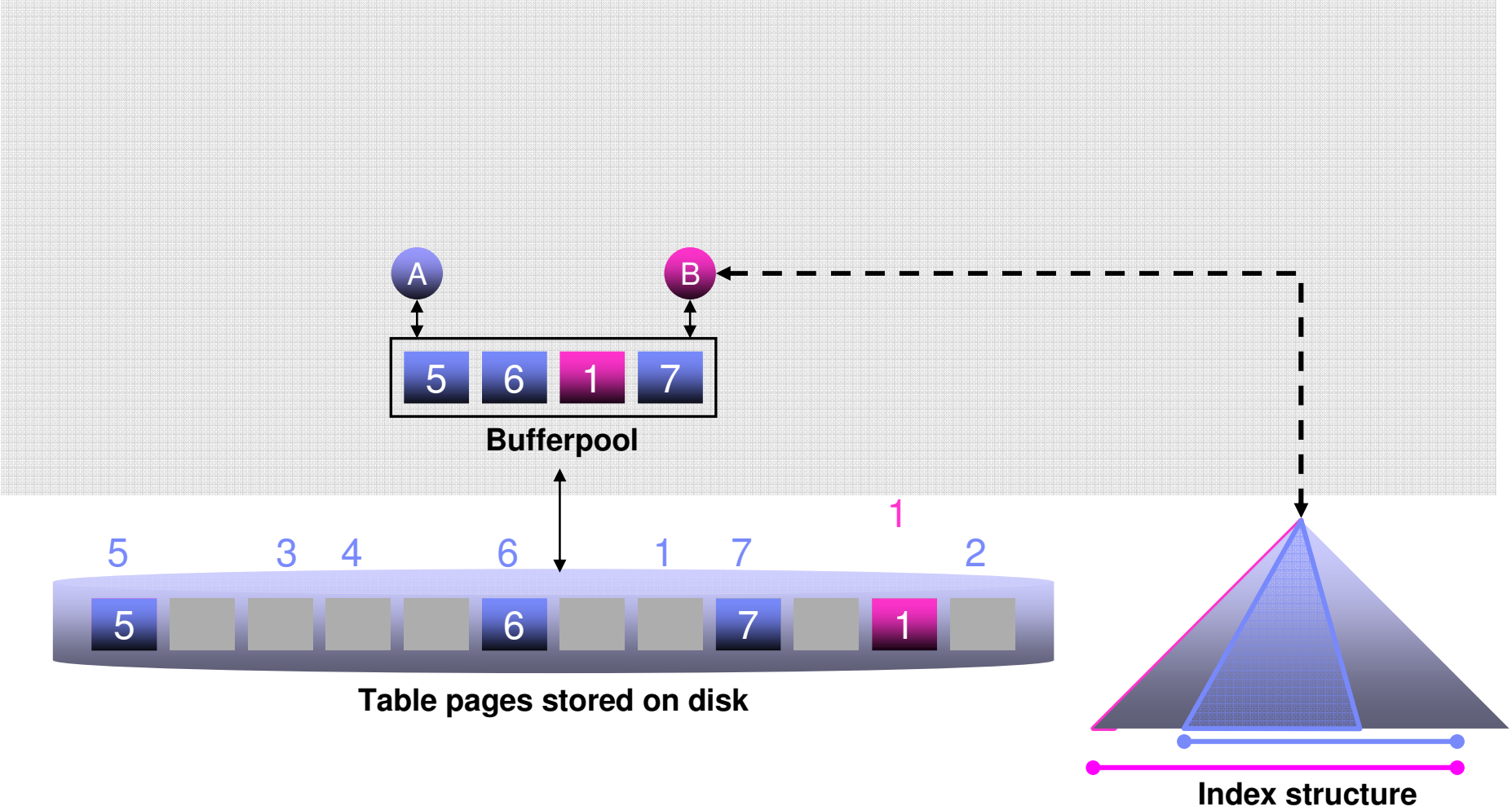
Current Index Scan Architecture



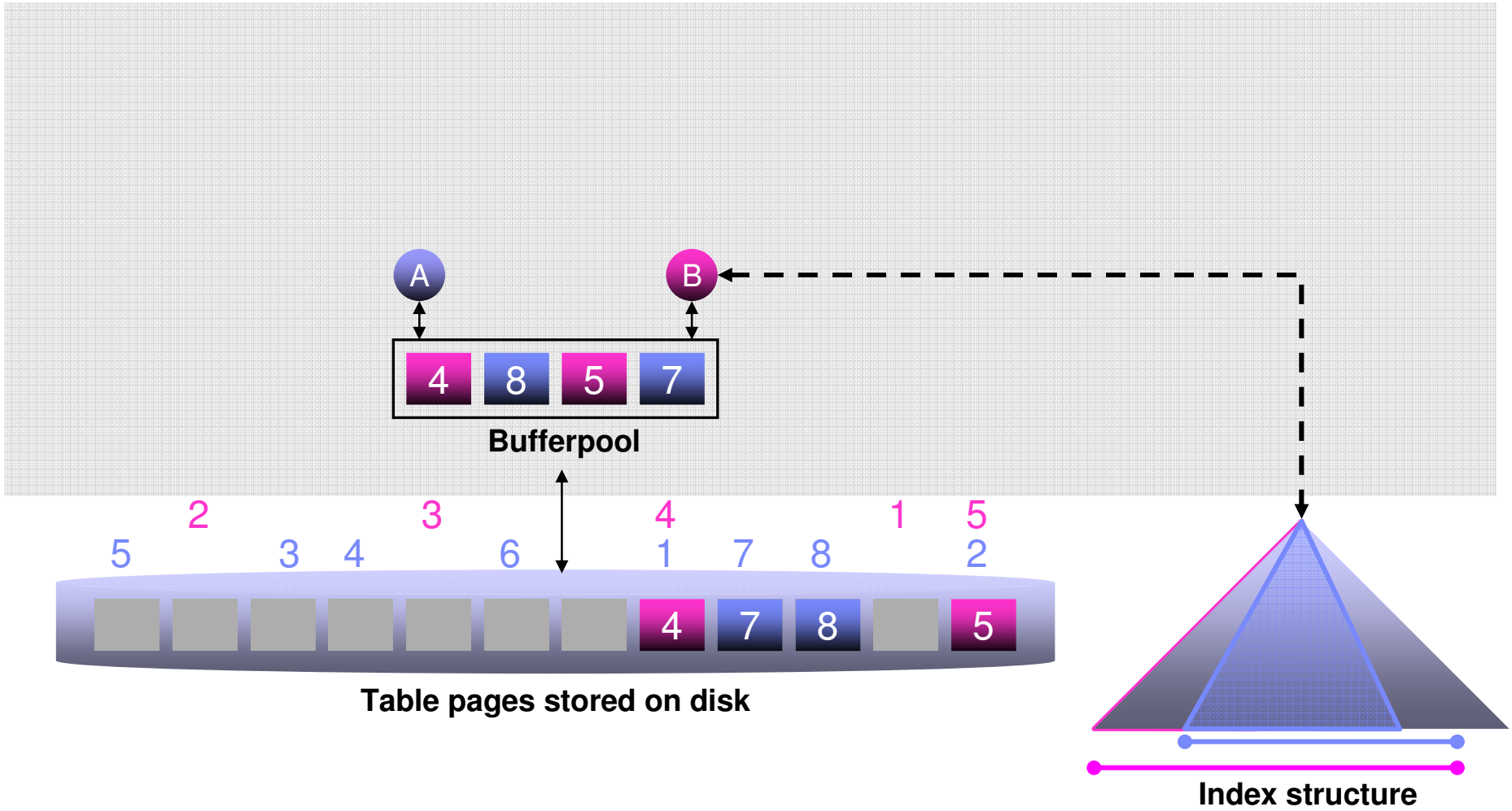
Current Index Scan Architecture



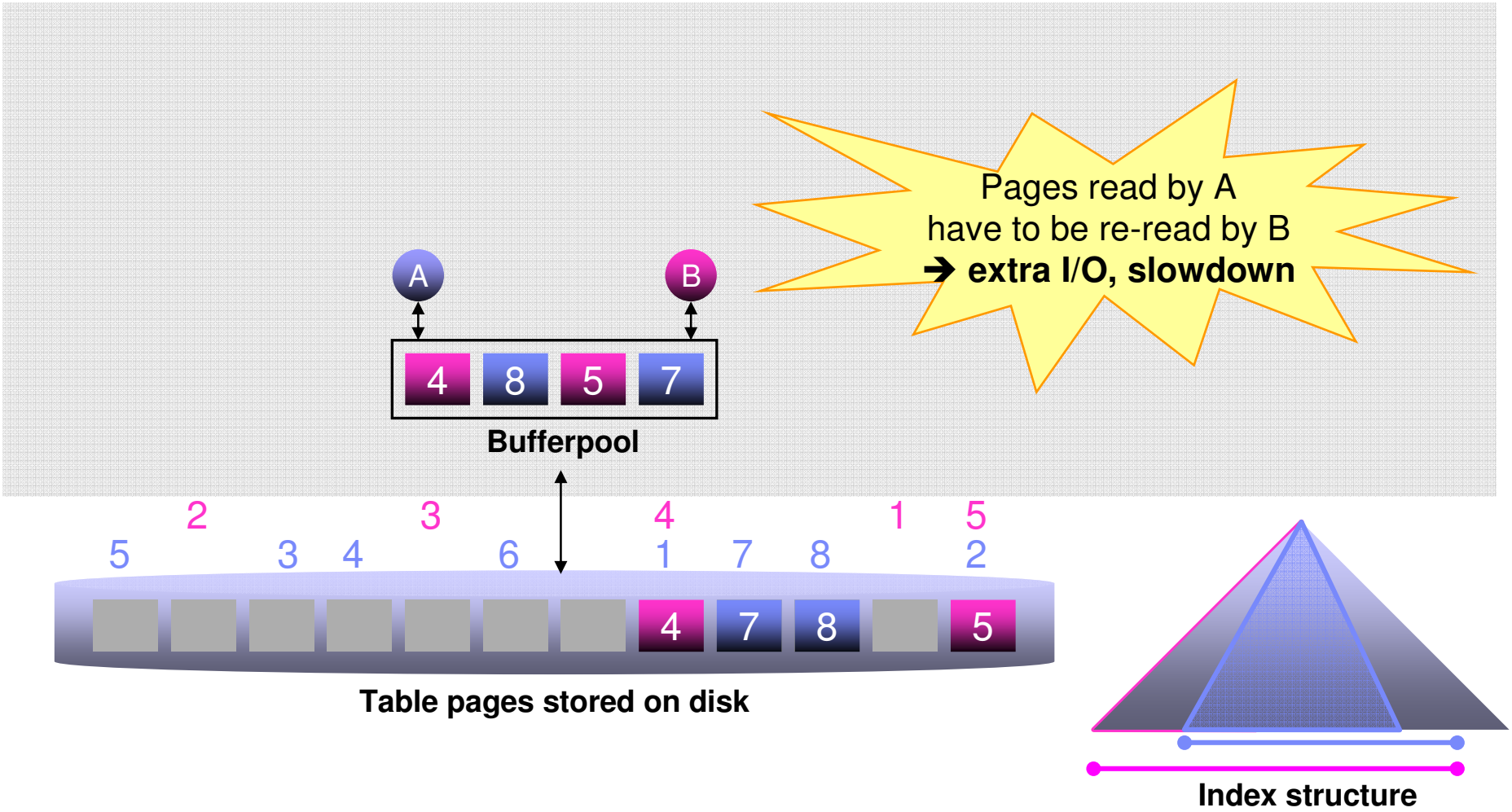
Current Index Scan Architecture



Current Index Scan Architecture



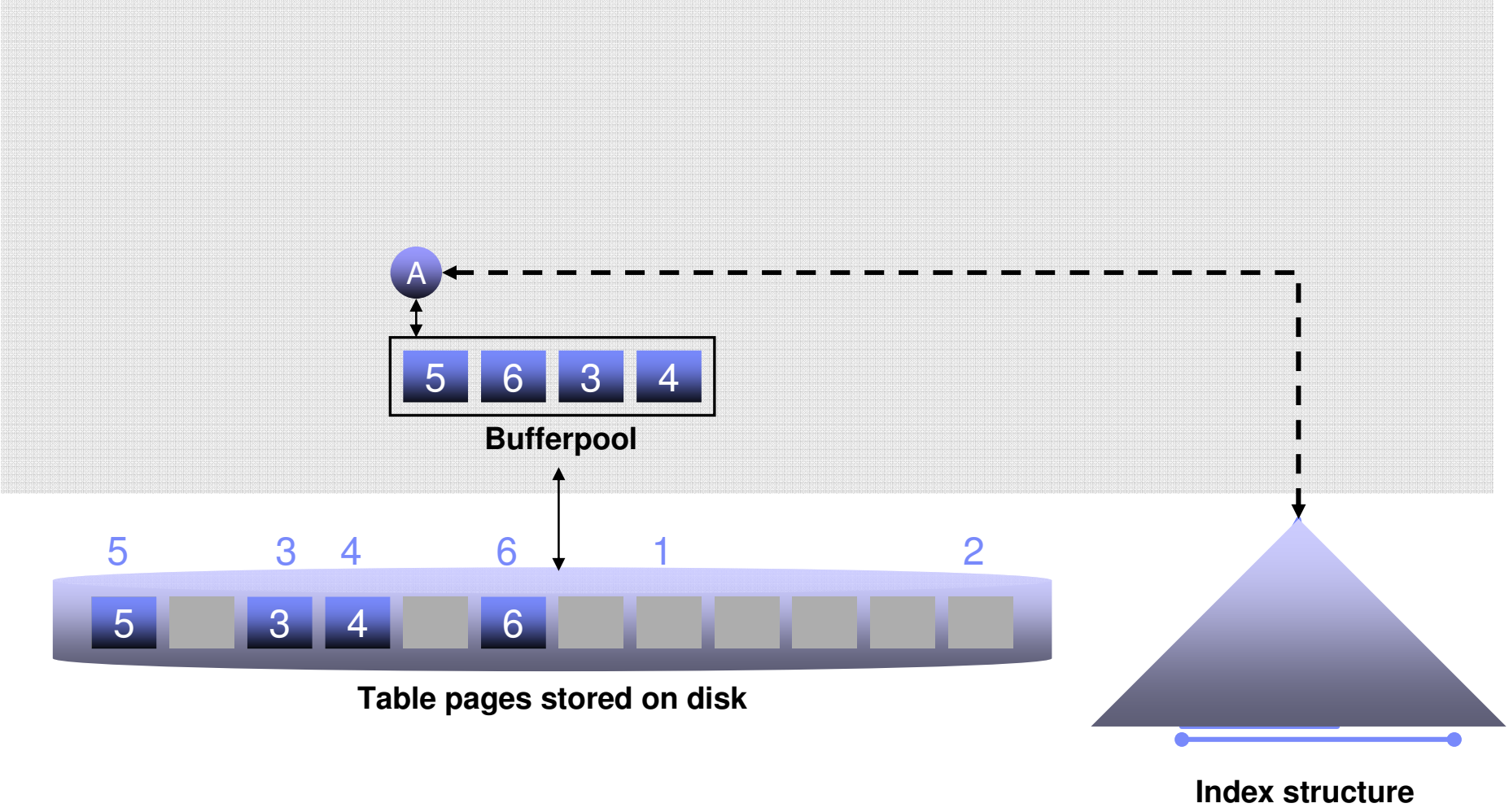
Current Index Scan Architecture



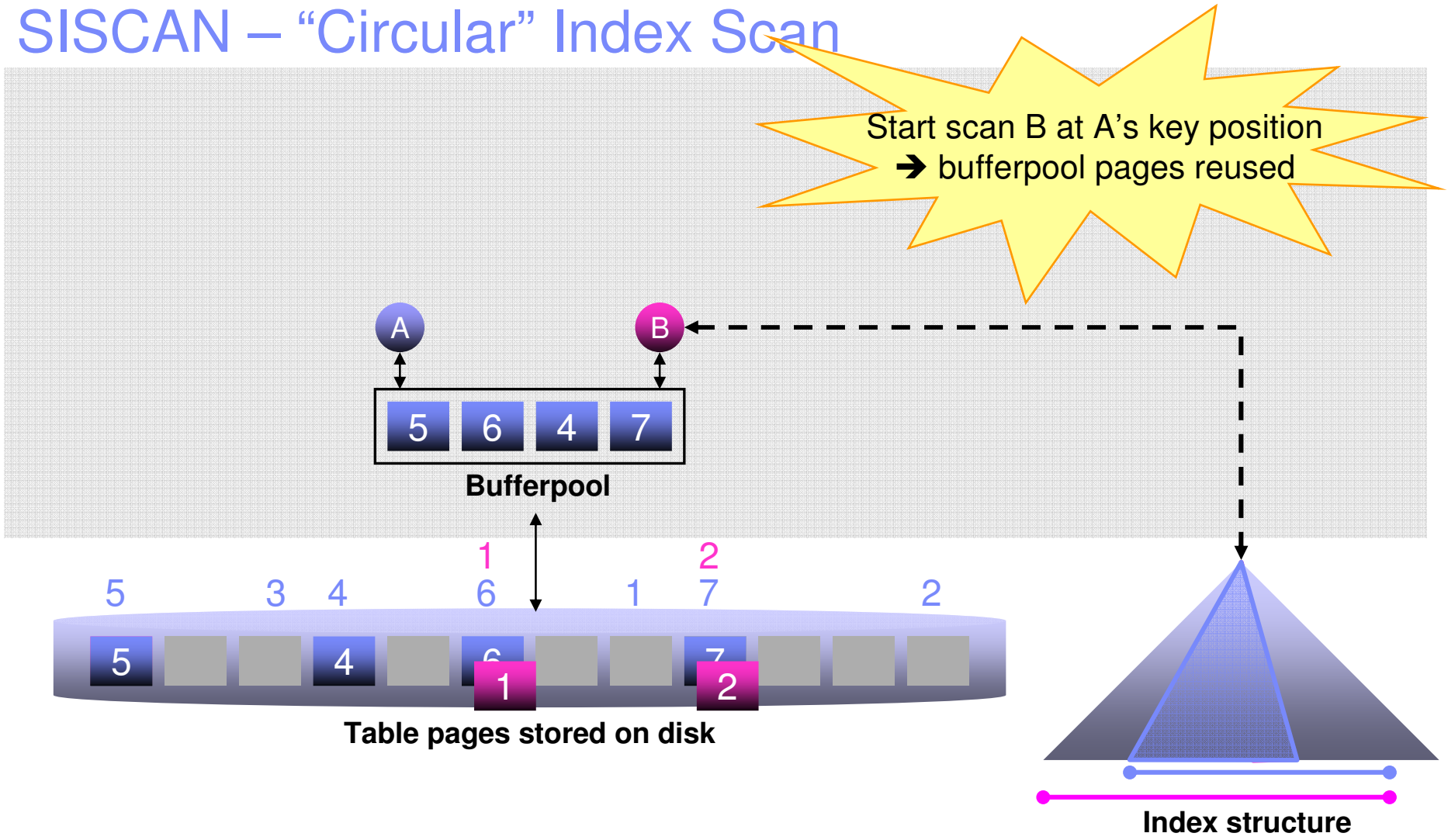
Outline

- Current Index Scan Architecture
- **SISCAN – "Circular" Index Scan**
  - **Placement**
  - **Speed Control**
- **Implementation Issues**
  - **Index-independent Relative SISCAN Location**
  - **"Bufferpool-independent" SISCAN-aware Caching**
- **Experimental Results**
- **Conclusions**

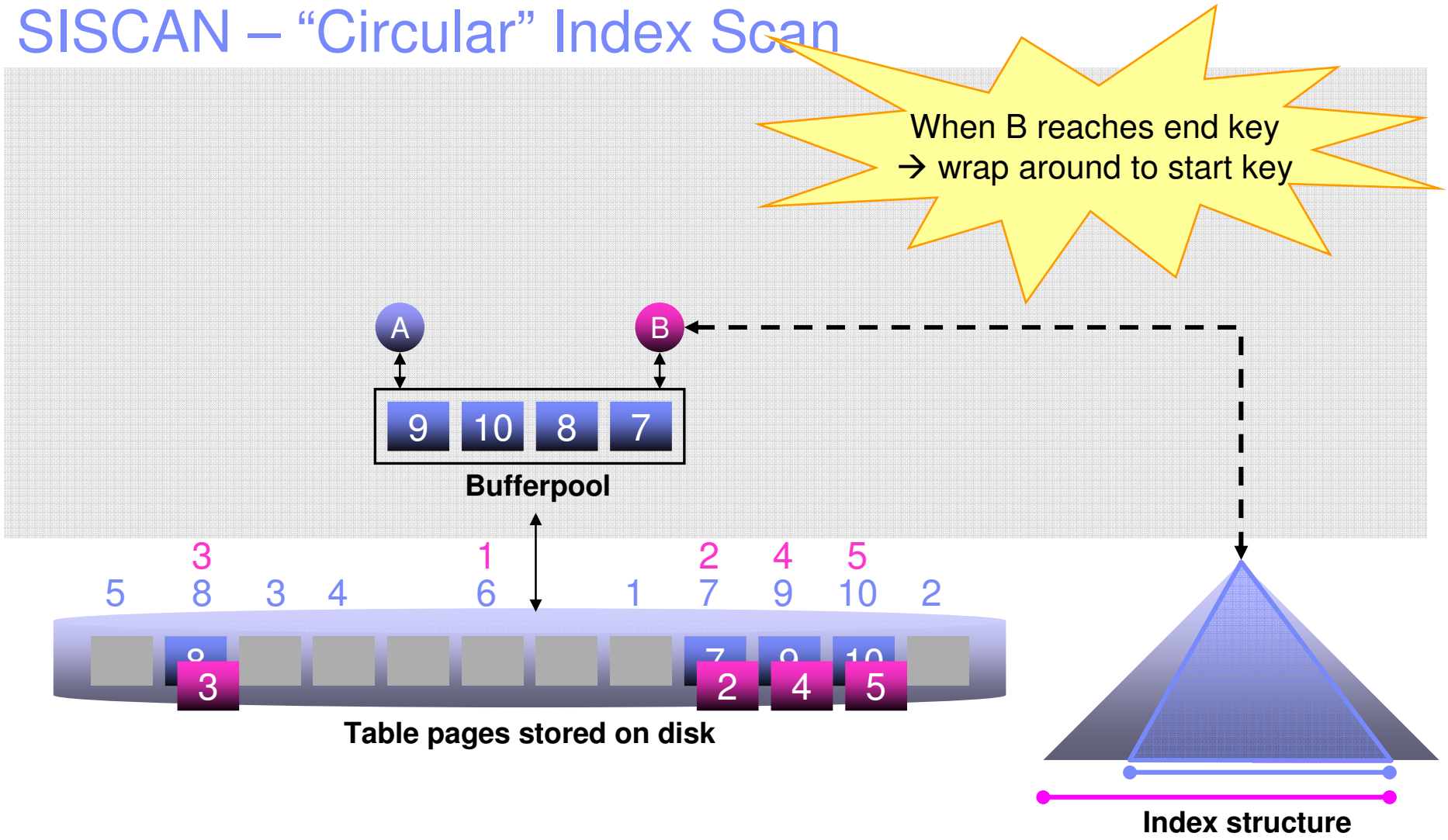
SISCAN – "Circular" Index Scan



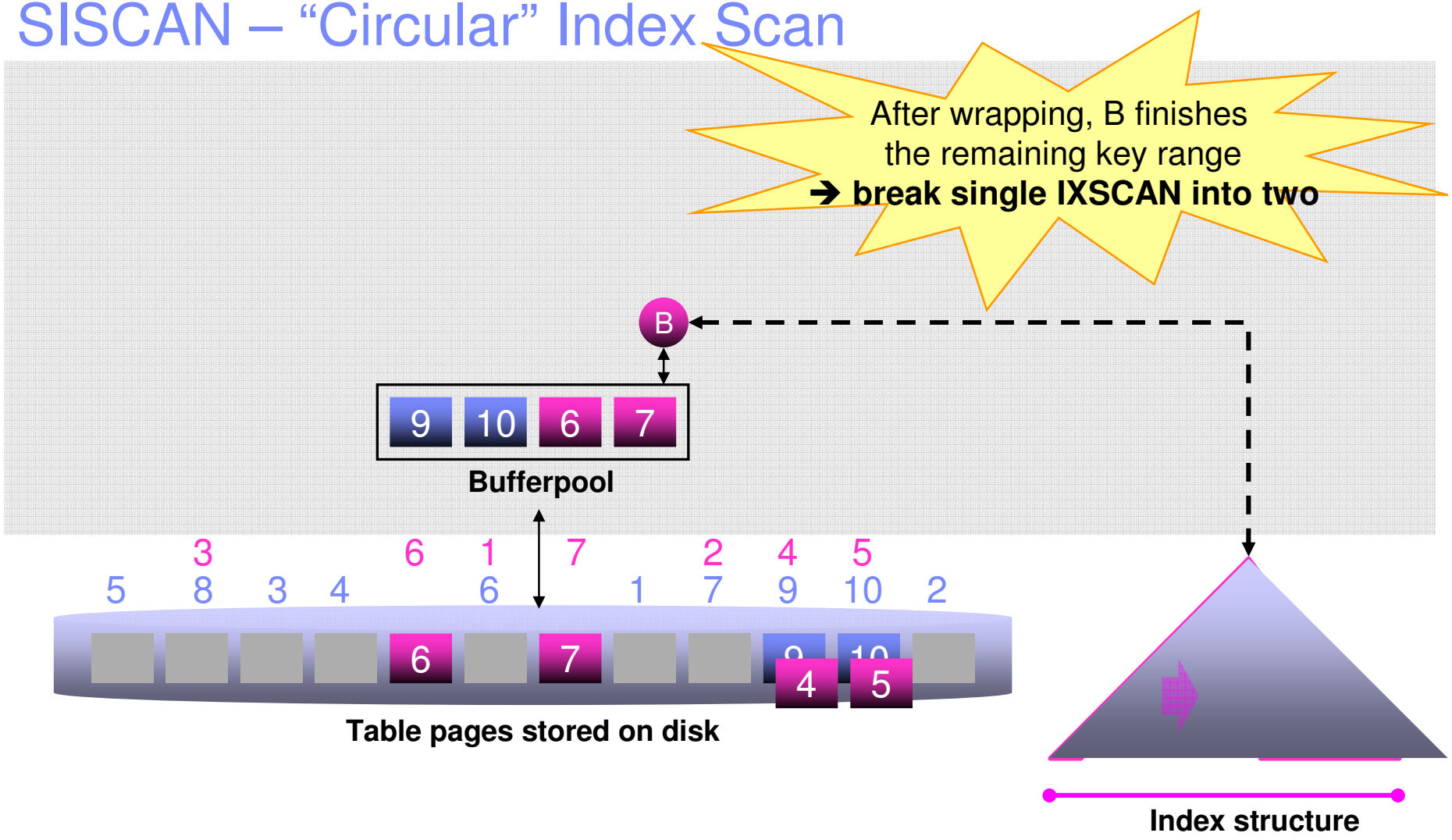
SISCAN – "Circular" Index Scan



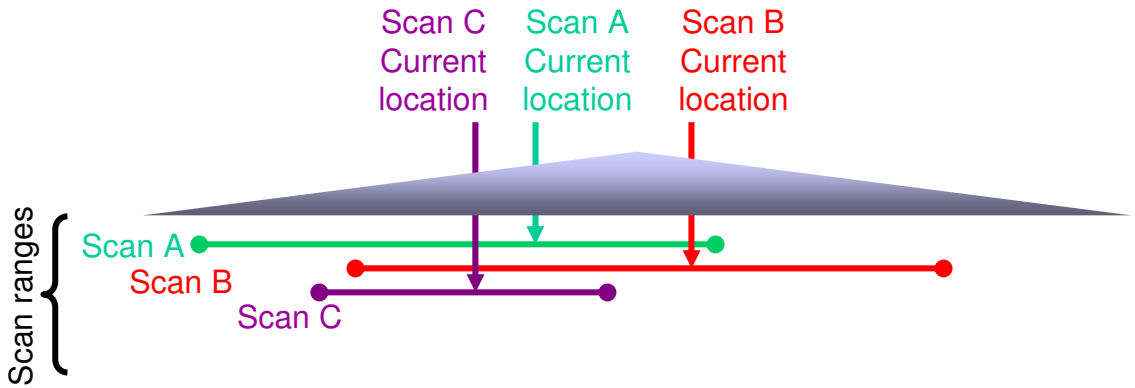
SISCAN – "Circular" Index Scan



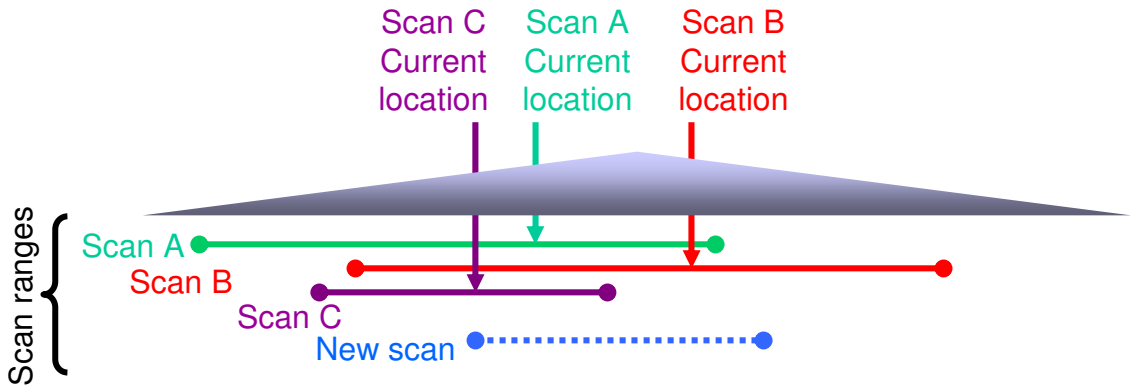
SISCAN – "Circular" Index Scan



Placement: Where to Start with Multiple Active SISCANs?

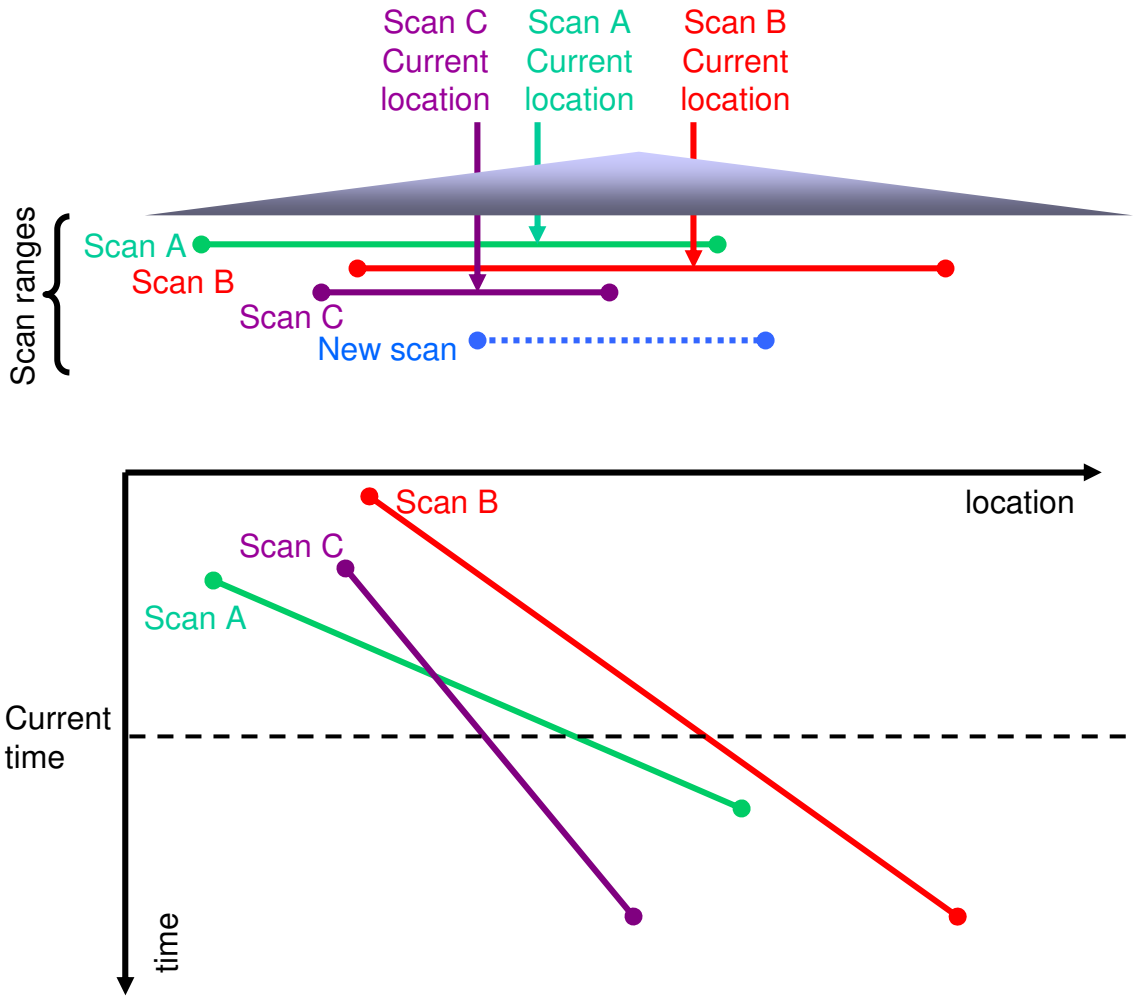


Placement: Where to Start with Multiple Active SISCANs?

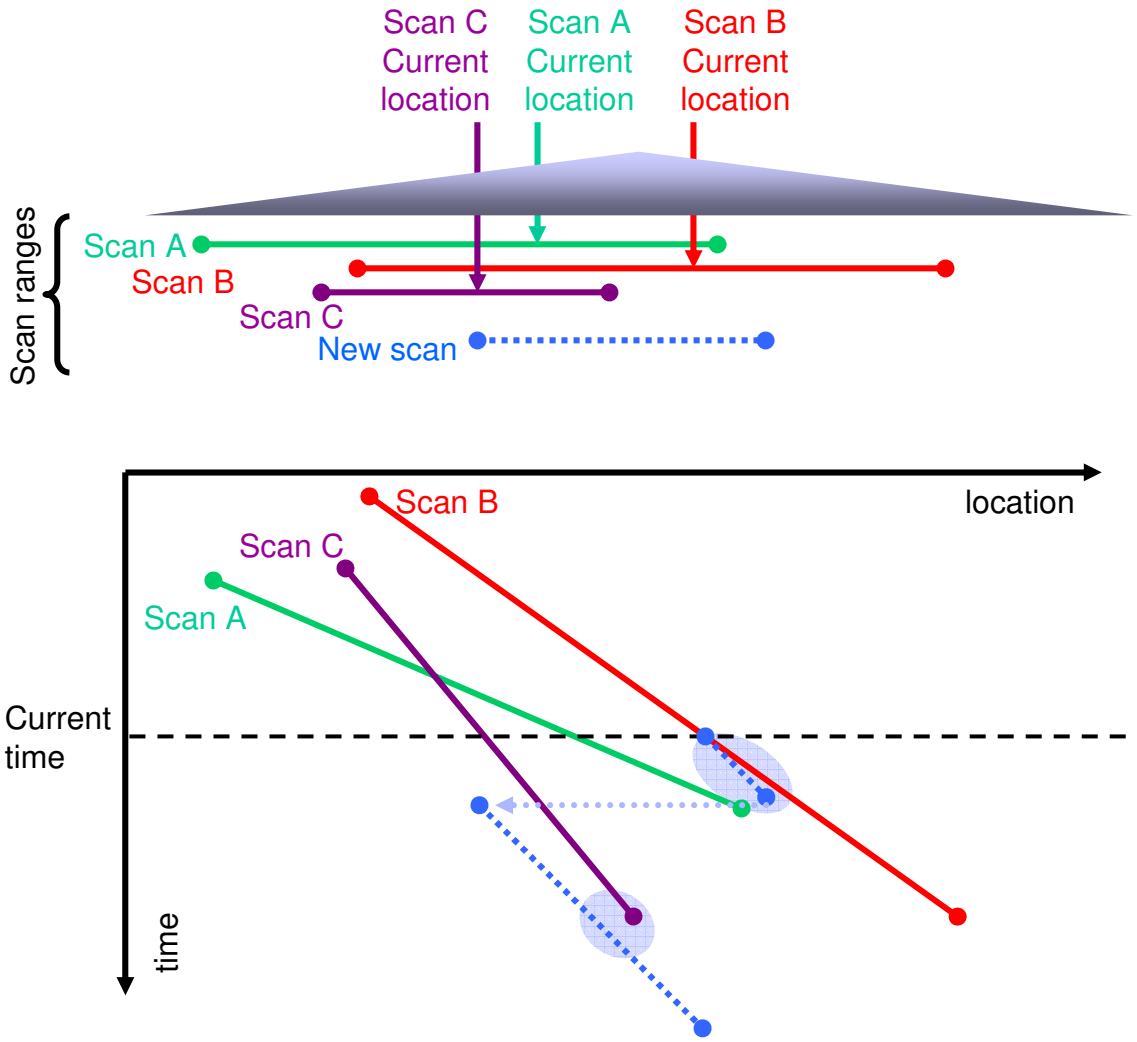


Where to start new scan?
→ need more information

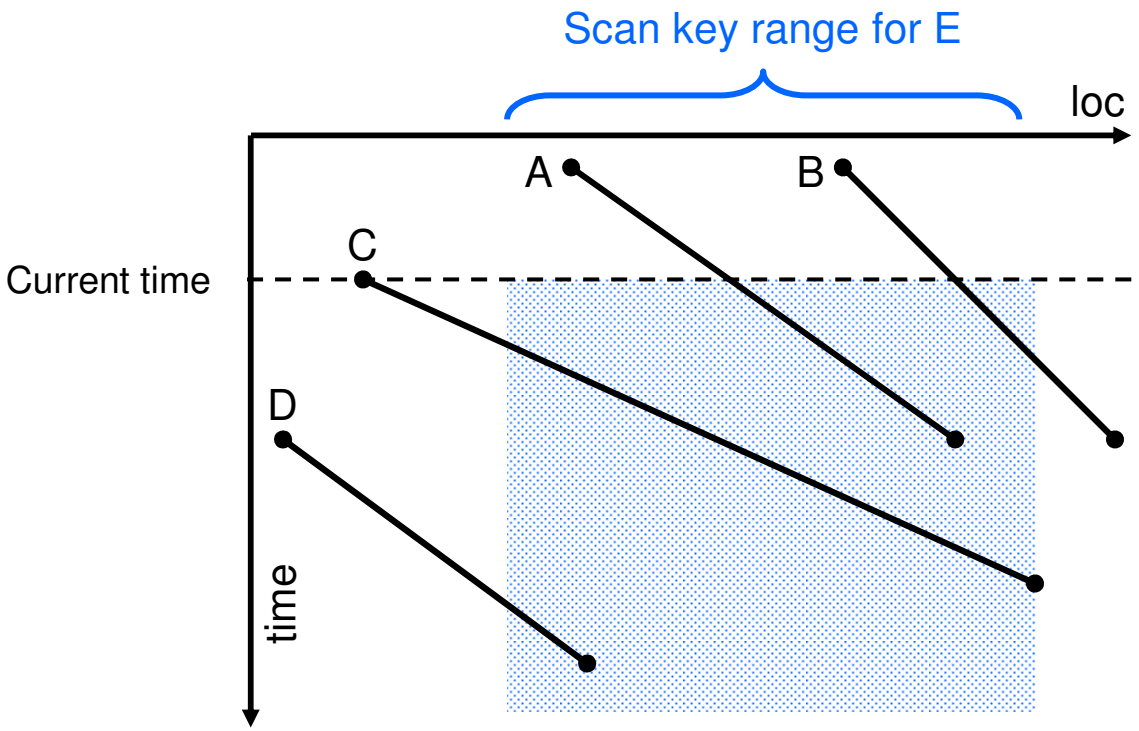
Placement: Where to Start with Multiple Active SISCANs?



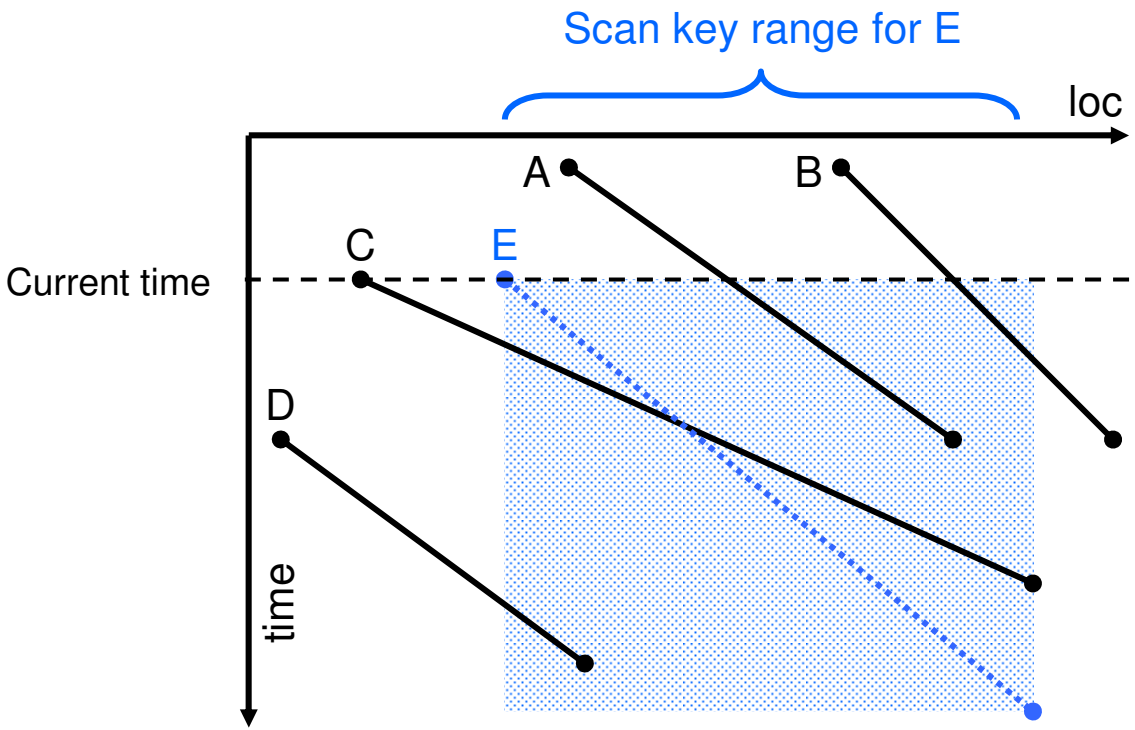
Placement: Where to Start with Multiple Active SISCANs?



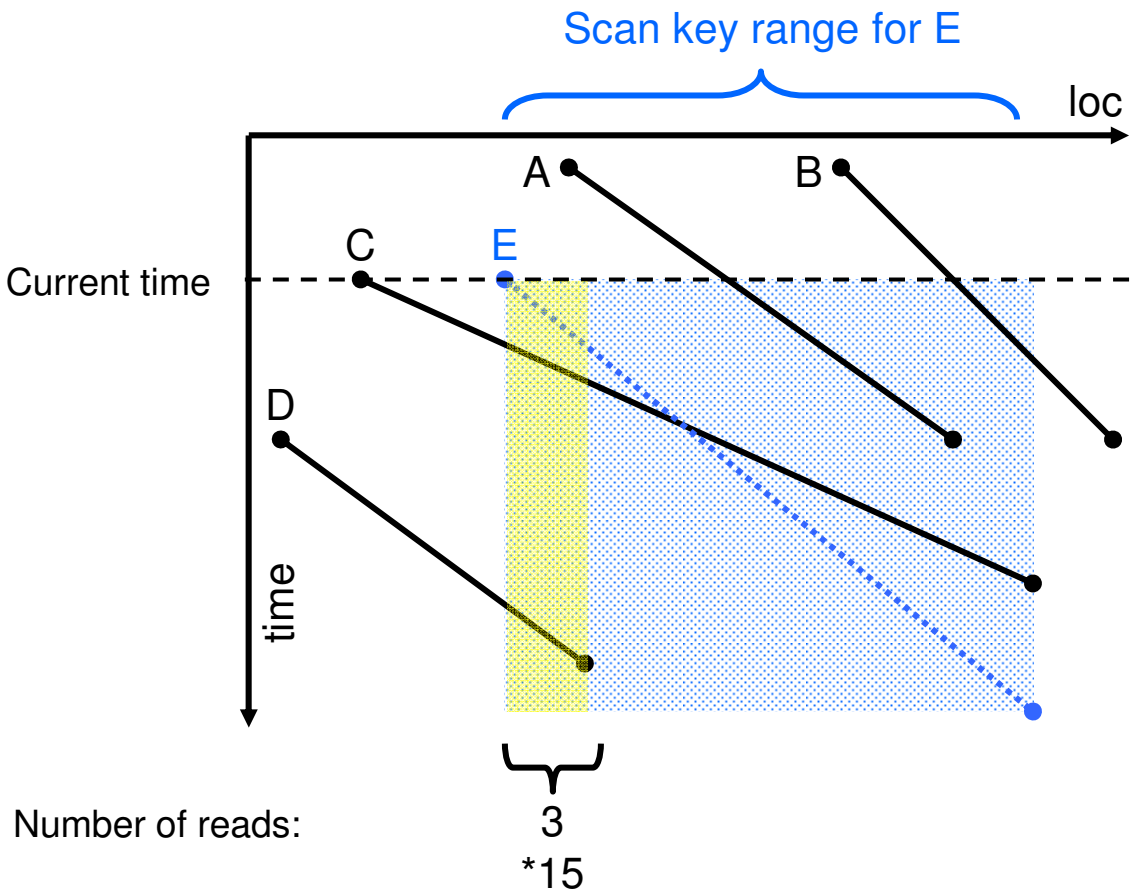
Estimating Sharing Potential



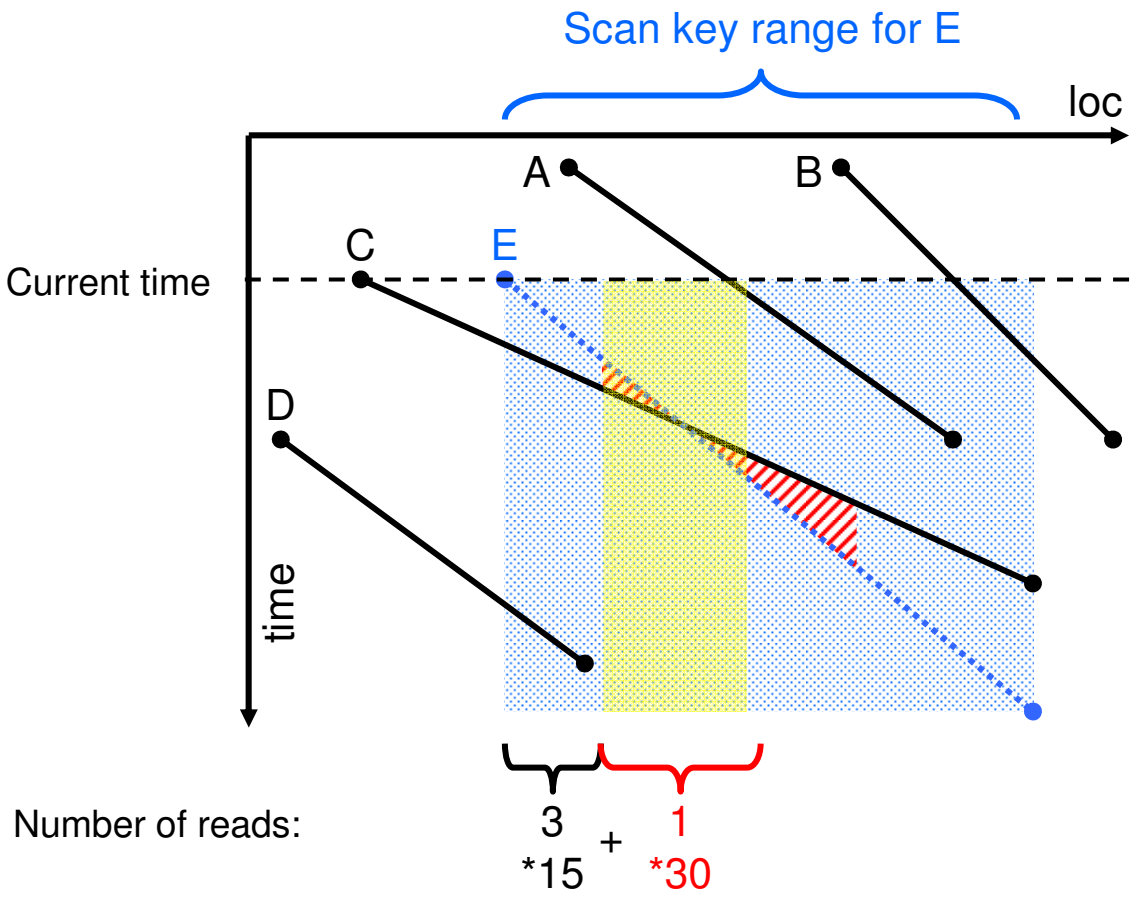
Estimating Sharing Potential



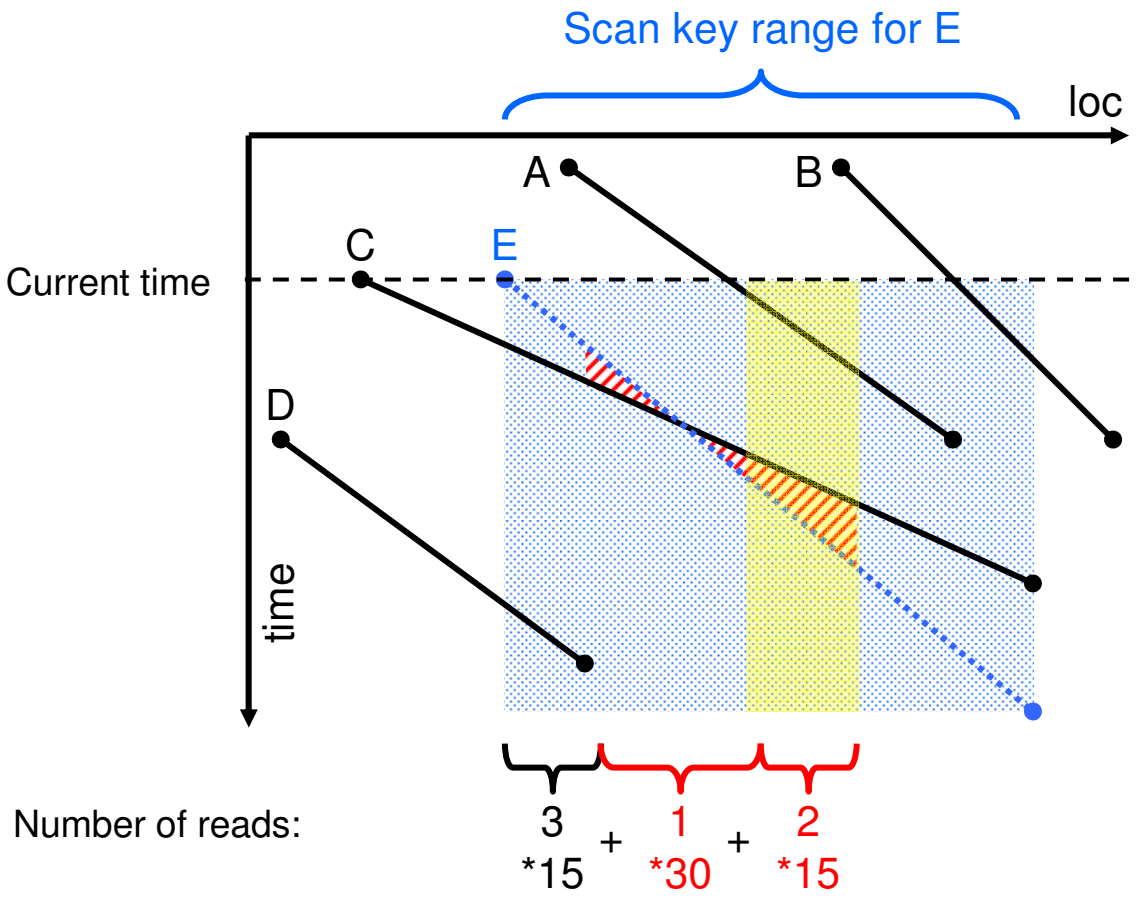
Estimating Sharing Potential



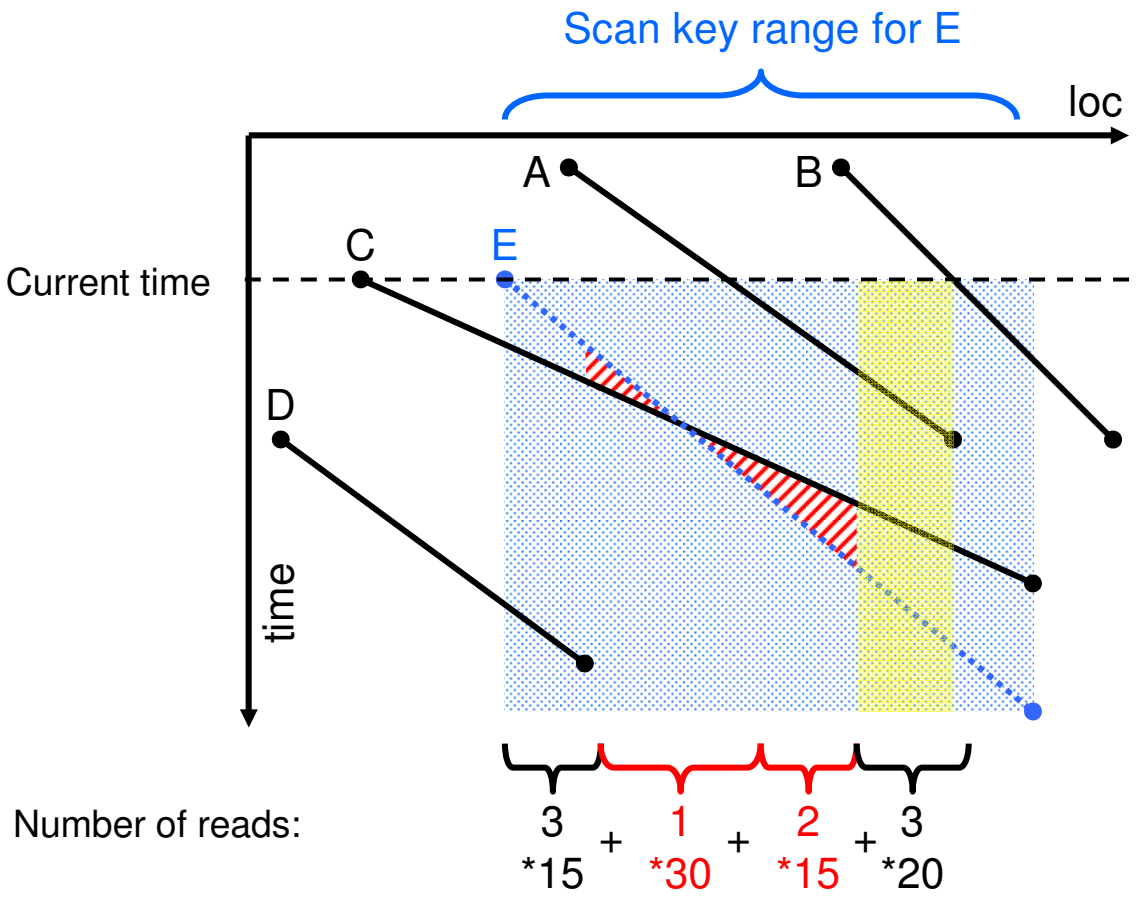
Estimating Sharing Potential



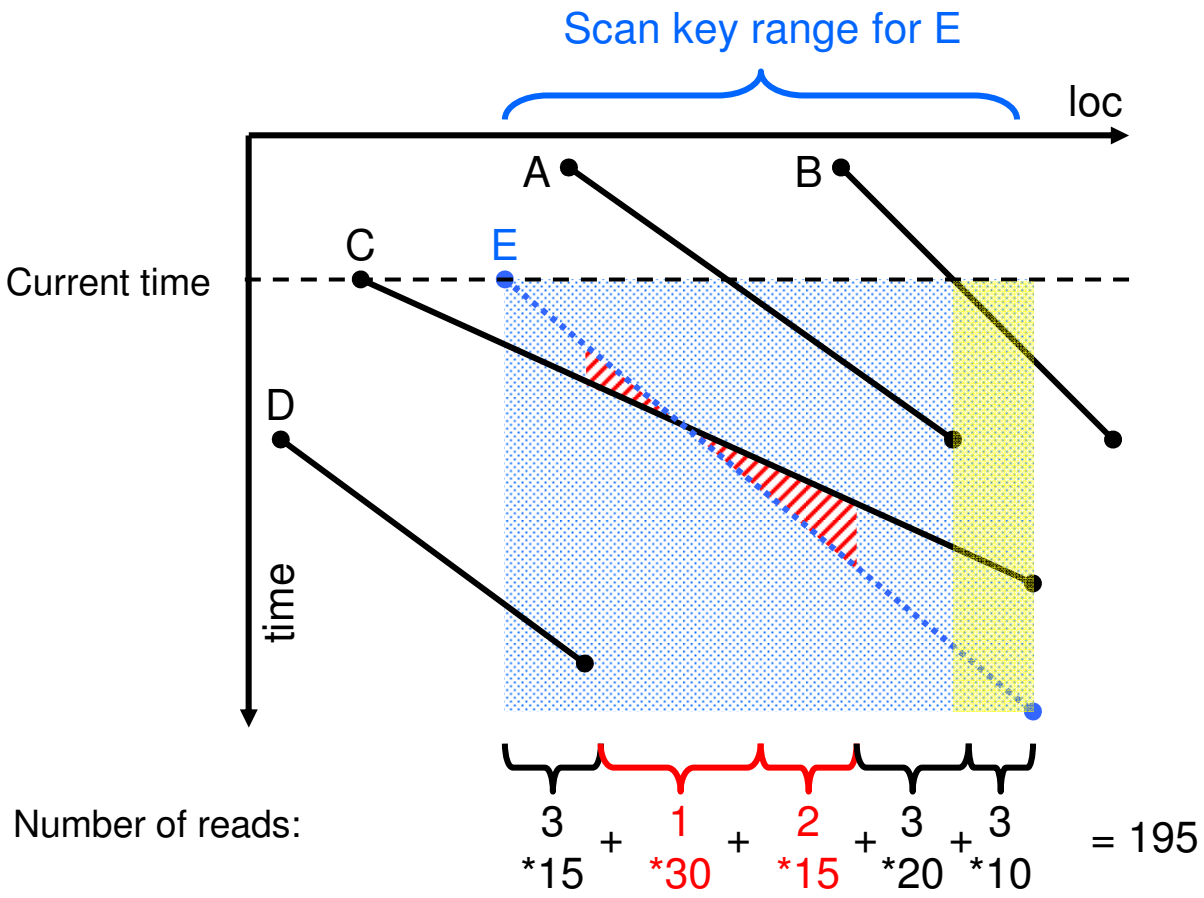
Estimating Sharing Potential



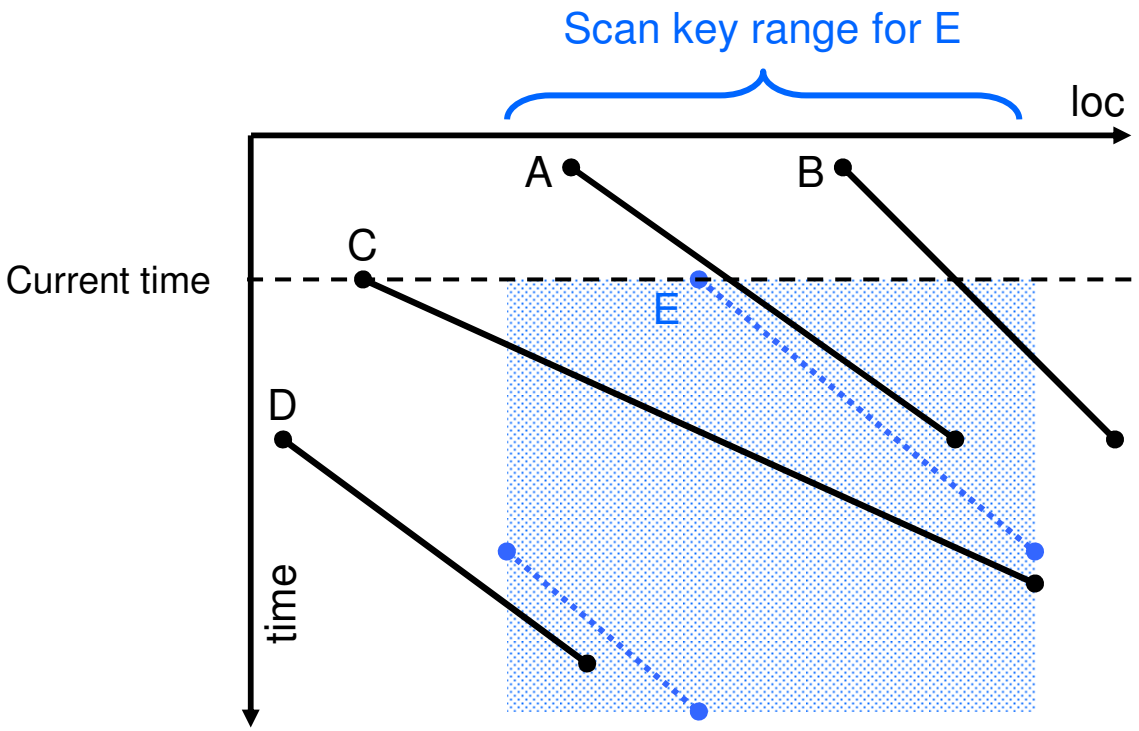
Estimating Sharing Potential



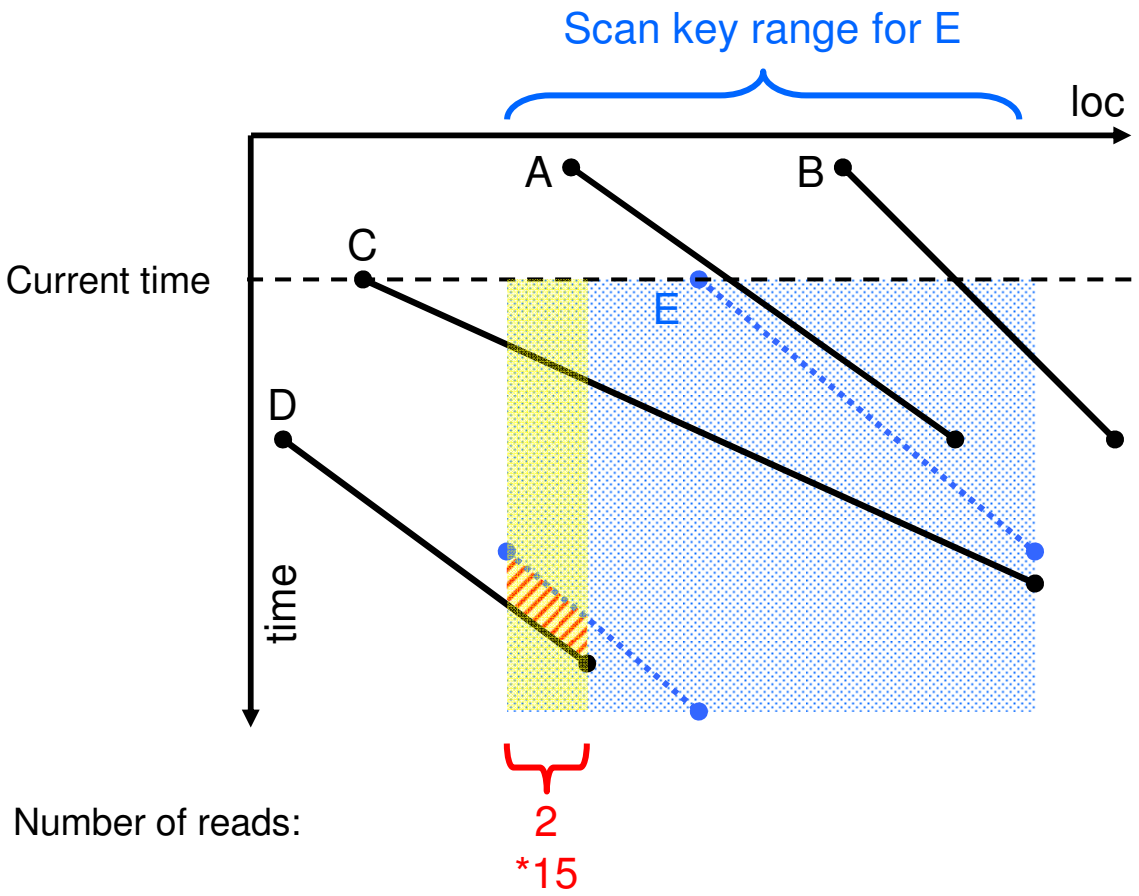
Estimating Sharing Potential



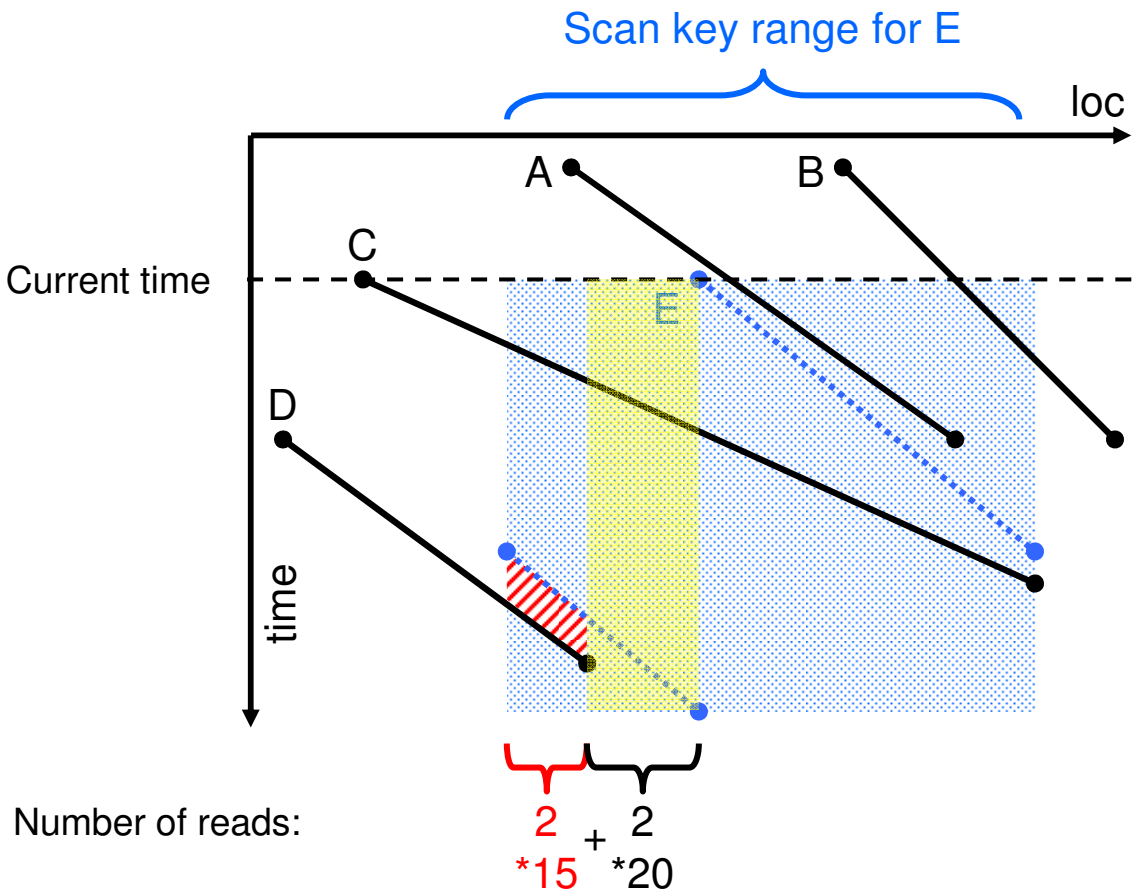
Estimating Sharing Potential



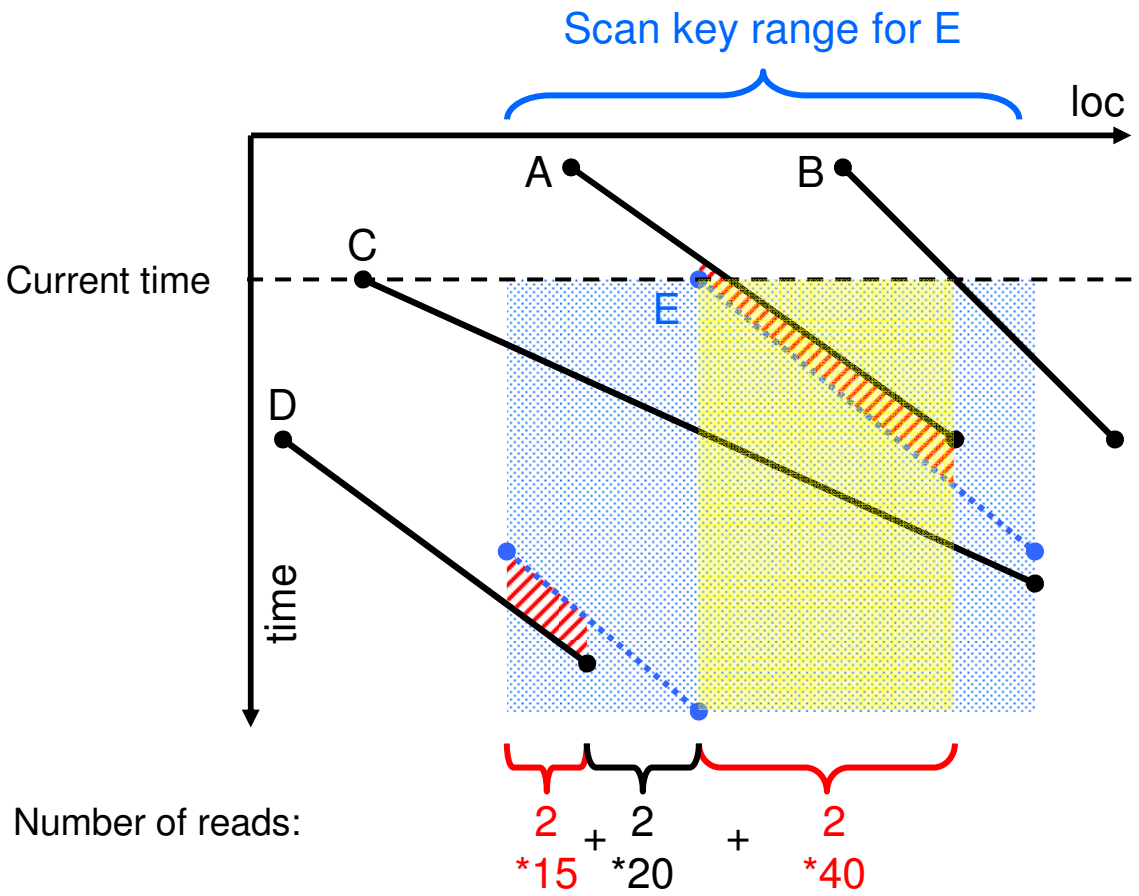
Estimating Sharing Potential



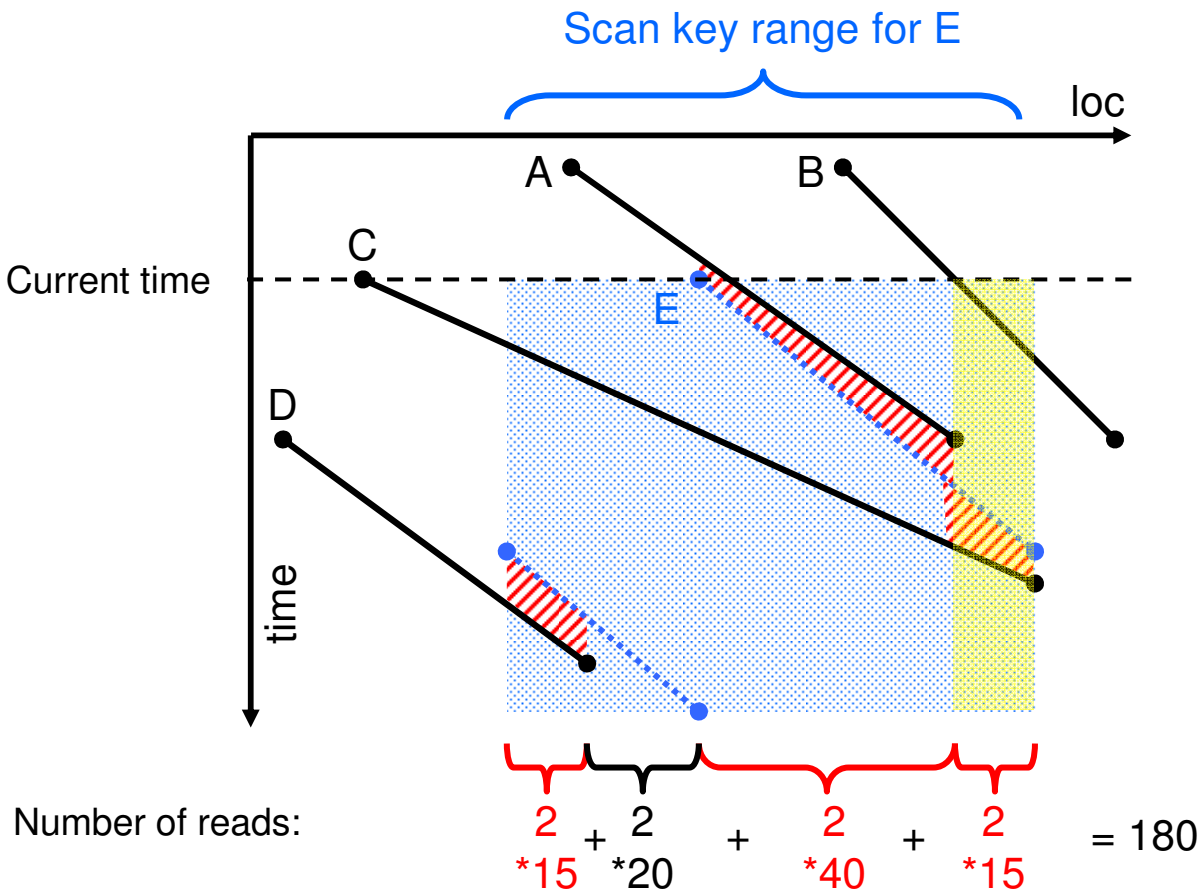
Estimating Sharing Potential



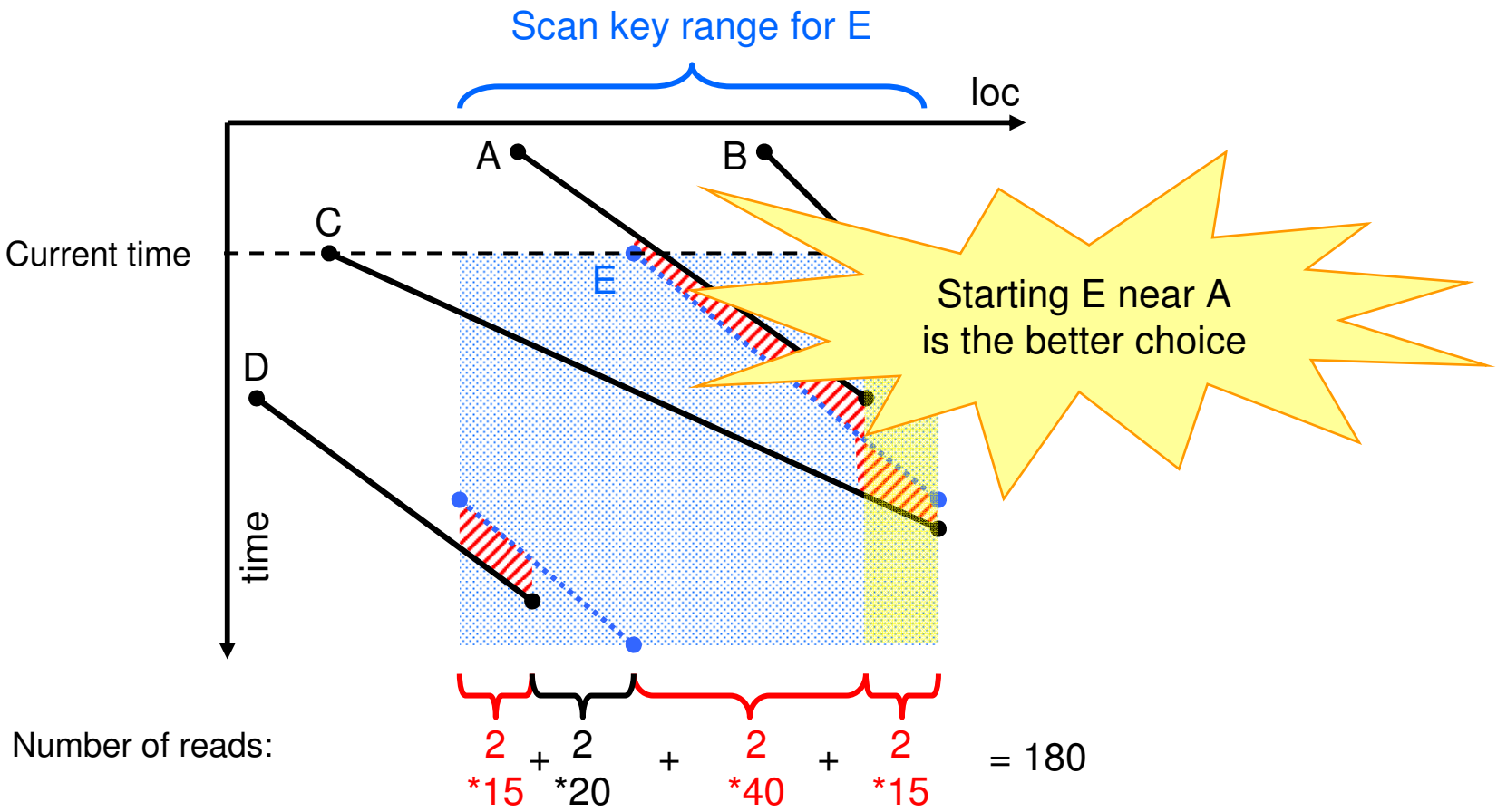
Estimating Sharing Potential



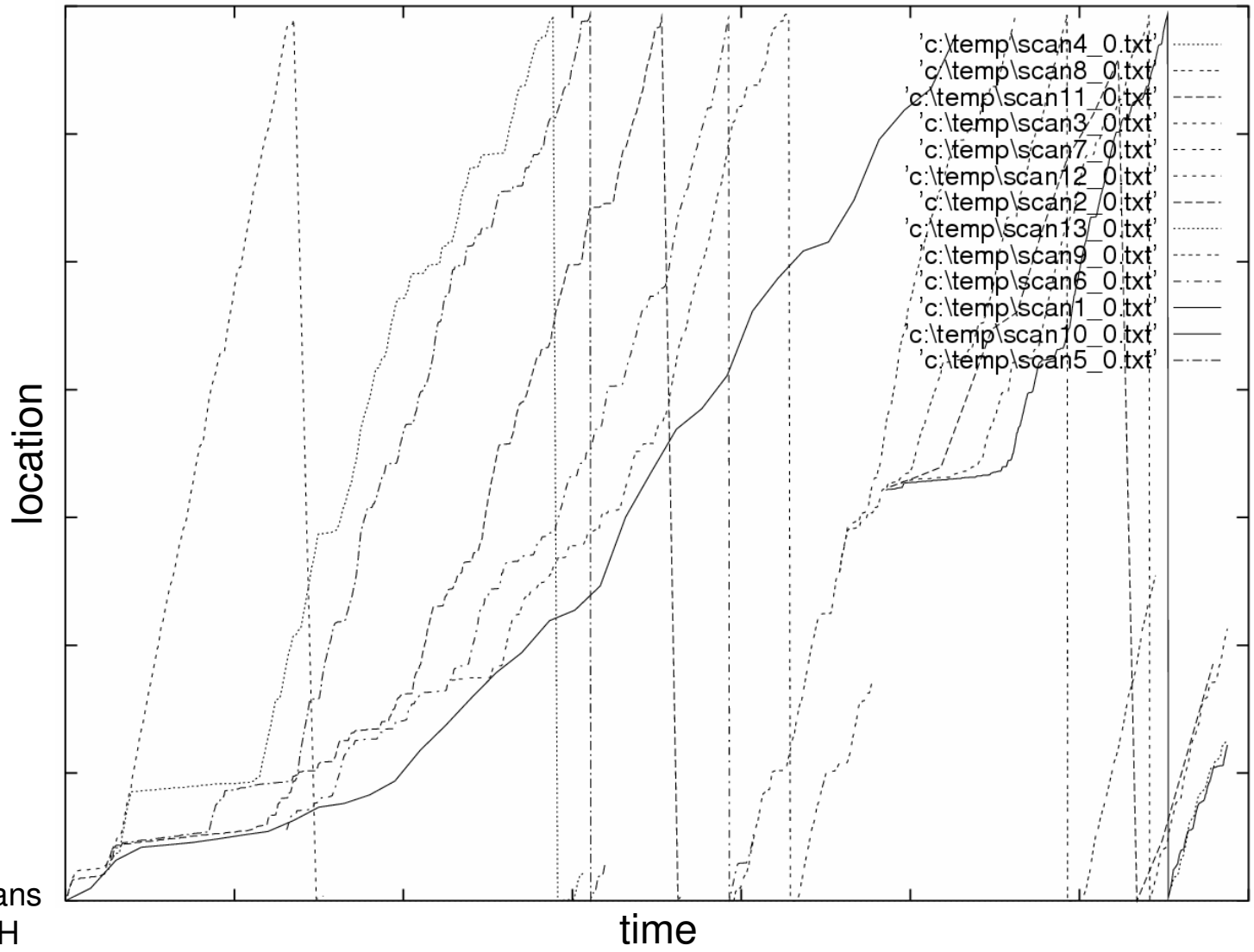
Estimating Sharing Potential



Estimating Sharing Potential

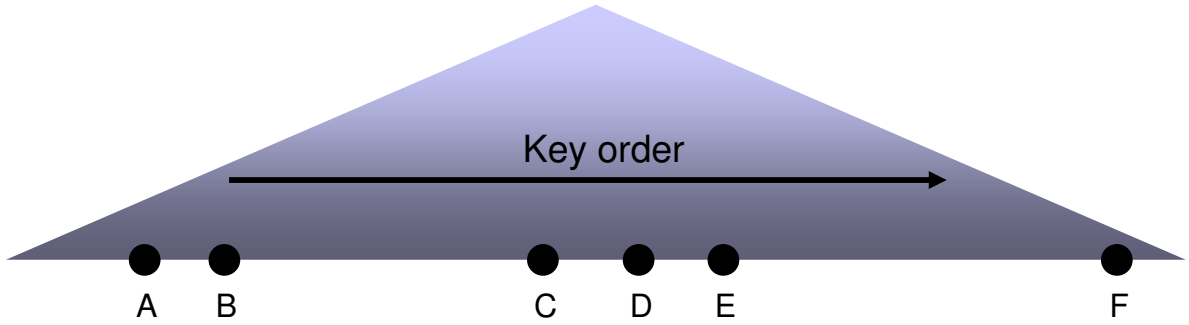


Problem solved? No, scans “drift” apart!



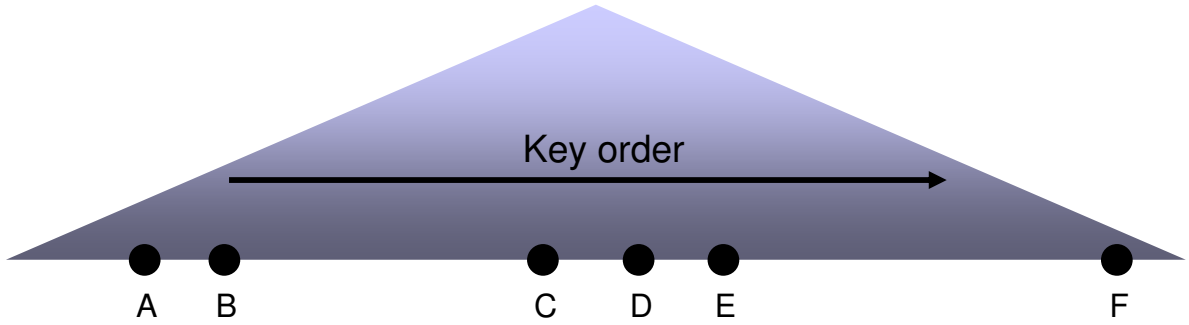
LINEITEM scans during TPC-H

Problem with Drift

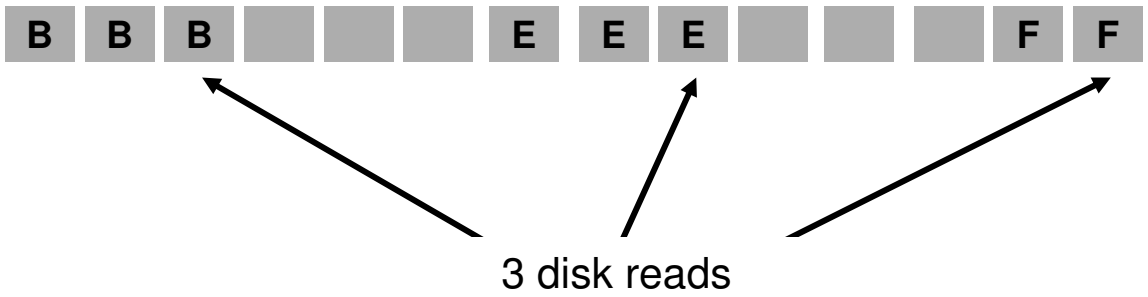


Initial reader of pages in key order

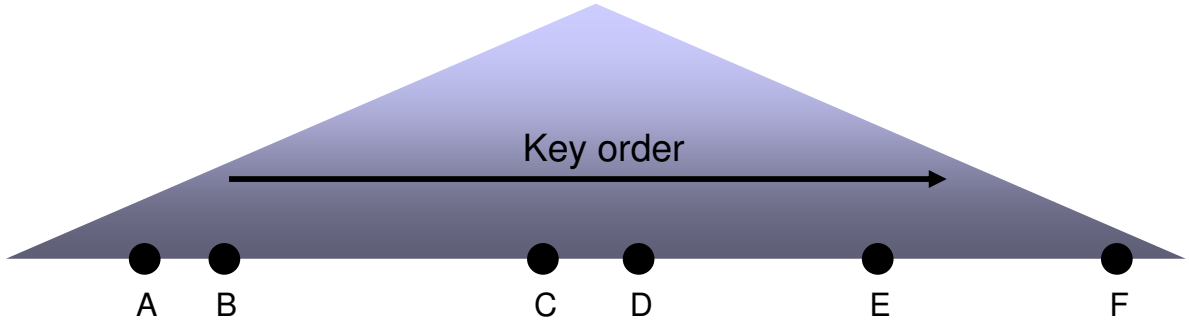
Problem with Drift

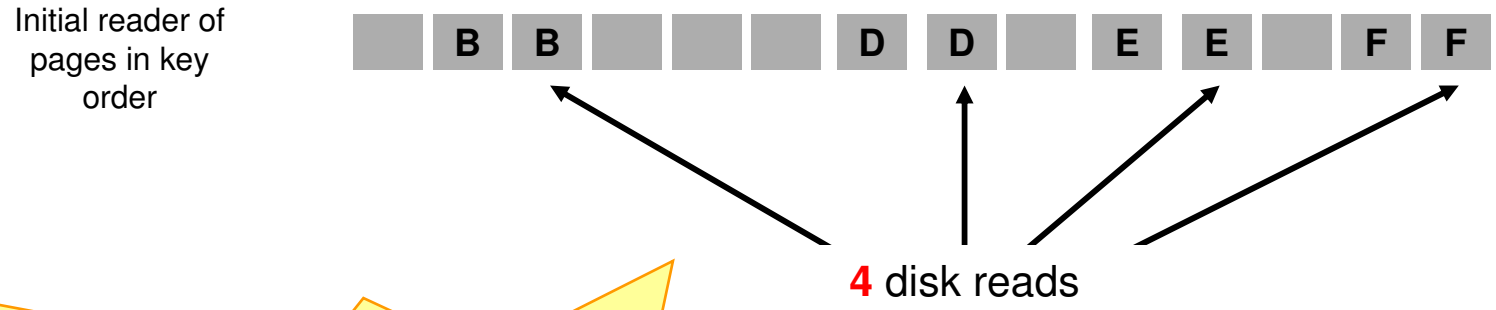


Initial reader of pages in key order



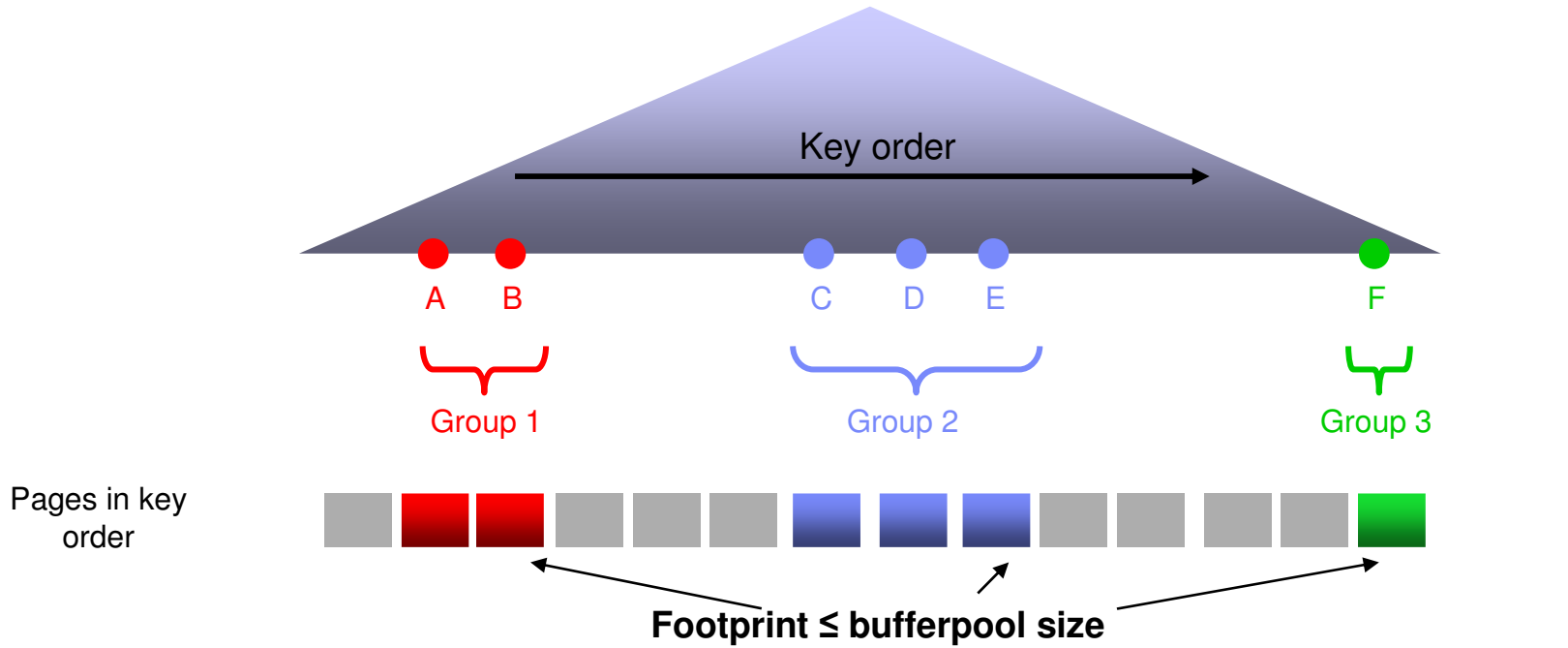
Problem with Drift





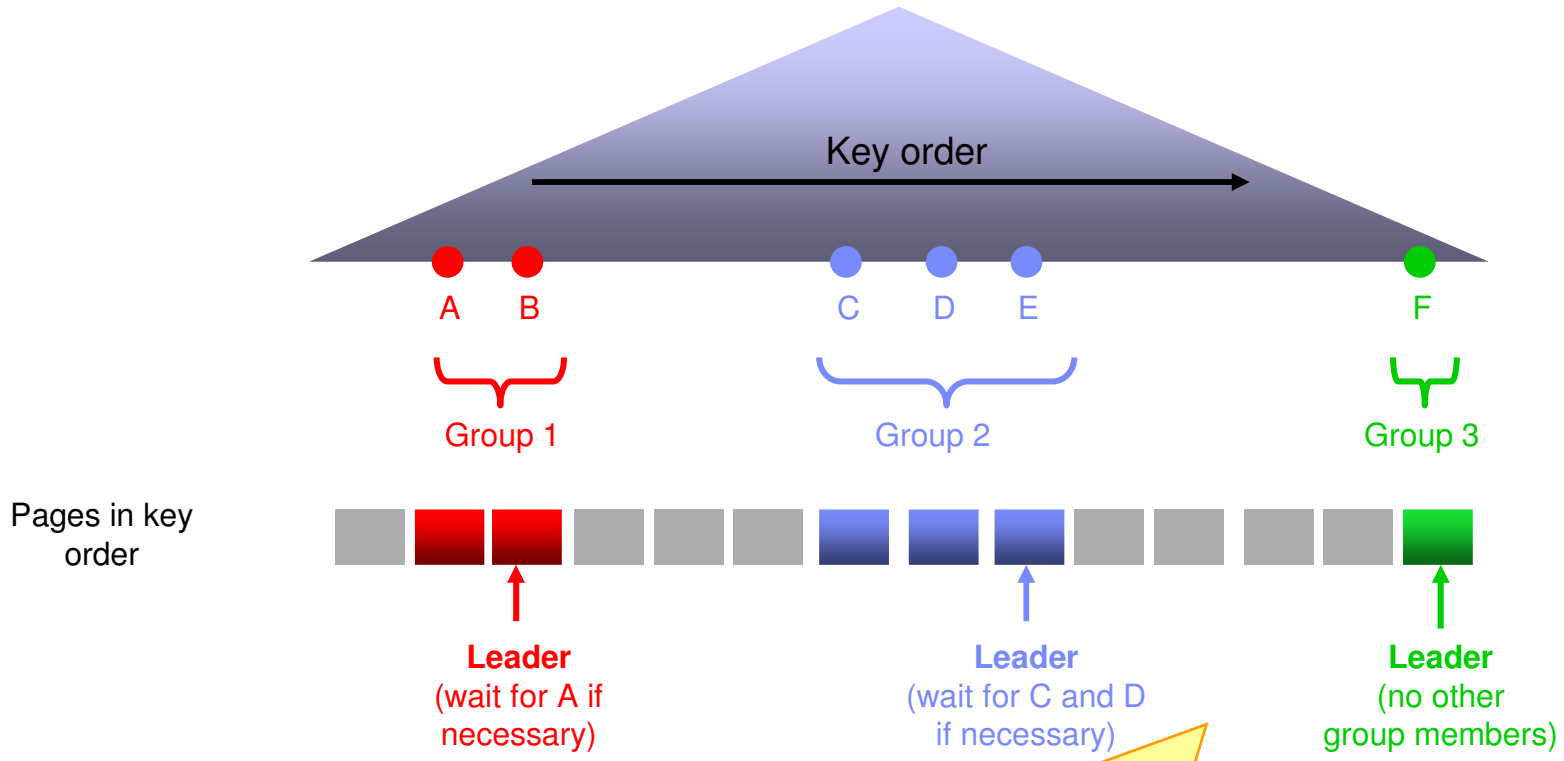
Drift leads to extra disk reads;
How to tolerate some drift without being too rigorous?

SISCAN Speed Control



Greedy algorithm:
group nearby SISCANs until
sum of group footprint exceeds bufferpool size

SISCAN Speed Control



Delay leader until group size ≤ bufferpool size / #groups
Upper bound on wait time per SISCAN;
Details similar to throttling for table scan sharing
[Lang et al., ICDE '07]

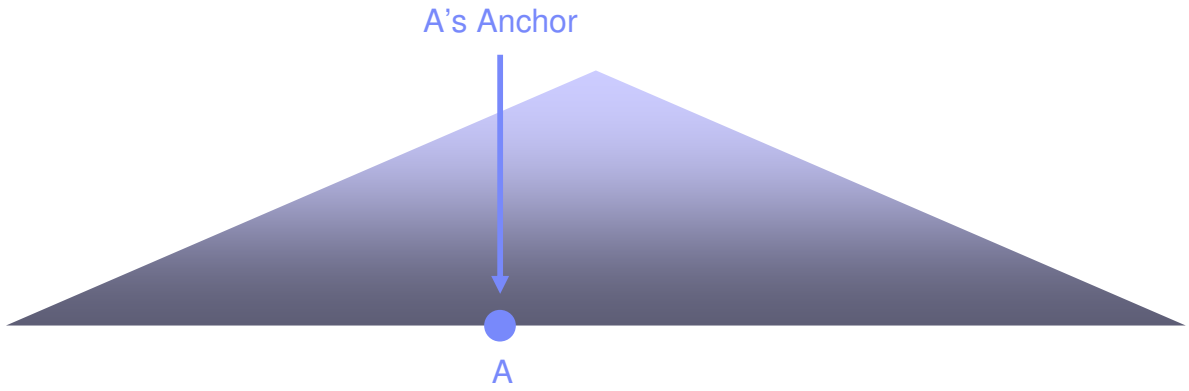
Outline

- Current Index Scan Architecture
- SISCAN – "Circular" Index Scan
  - Placement
  - Speed Control
- **Implementation Issues**
  - **Index-independent Relative SISCAN Location**
  - **"Bufferpool-independent" SISCAN-aware Caching**
- **Experimental Results**
- **Conclusions**

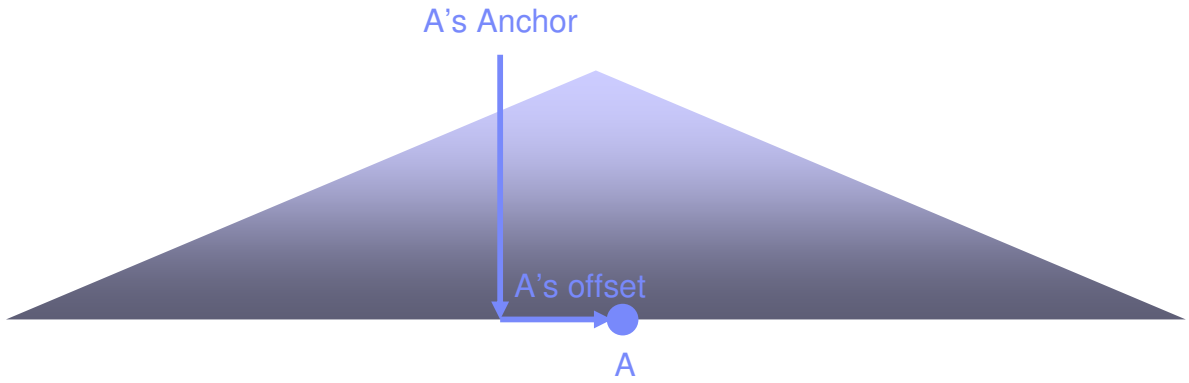
Index-Independent Relative SISCAN Locations

- Problem:
  hard to determine relative IXSCAN locations (while leaving the index a "black box")
- Example (key, page):
  ('Alice', 12), ('Bob', 38), ('Bob', 91), ('Carol', 2)
- What are the relative locations of these scans?
- How far apart are they?

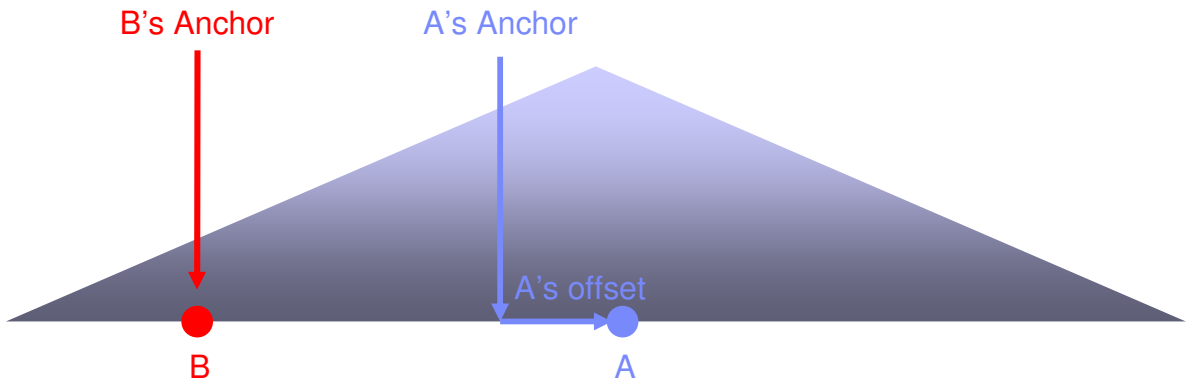
Index-Independent Relative SISCAN Locations



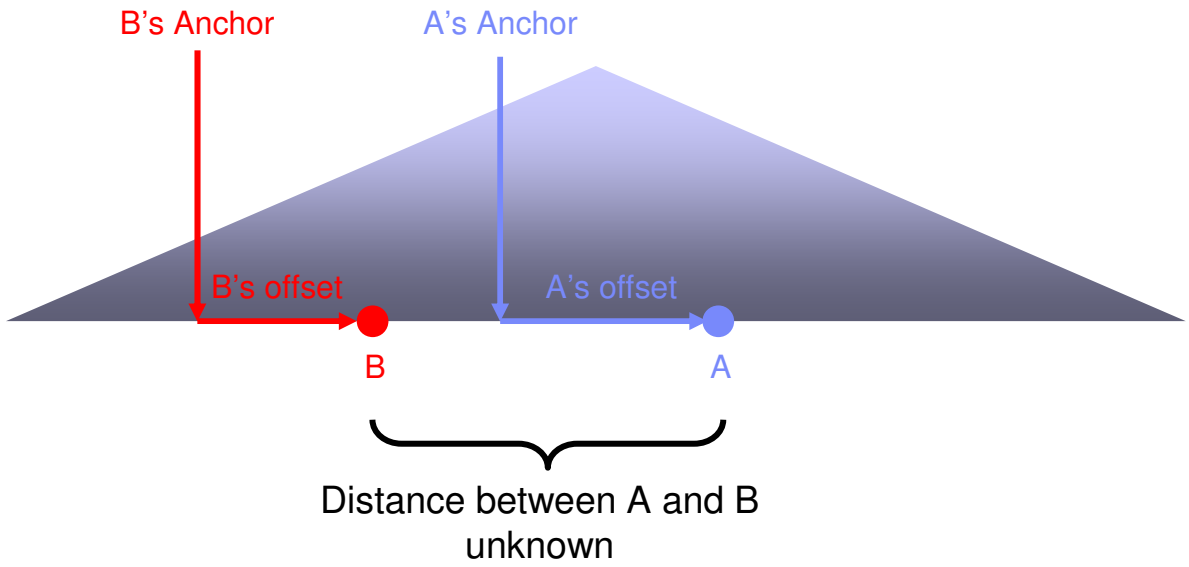
Index-Independent Relative SISCAN Locations



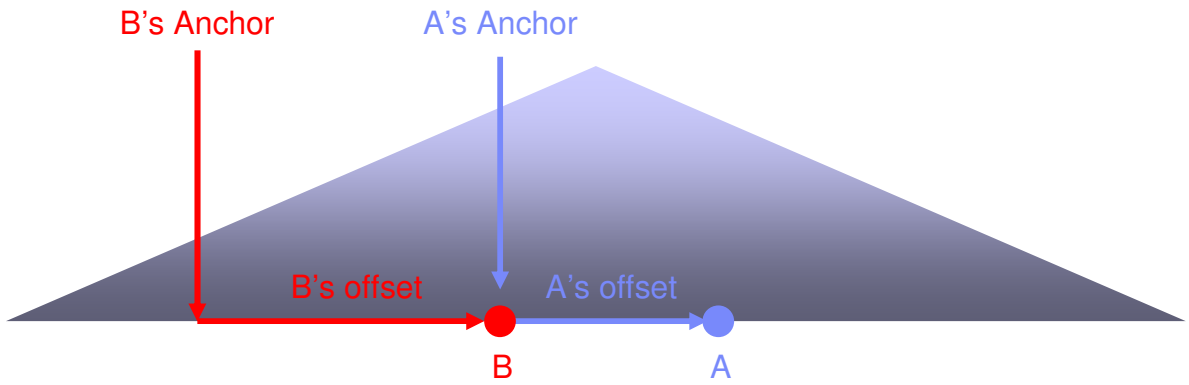
Index-Independent Relative SISCAN Locations



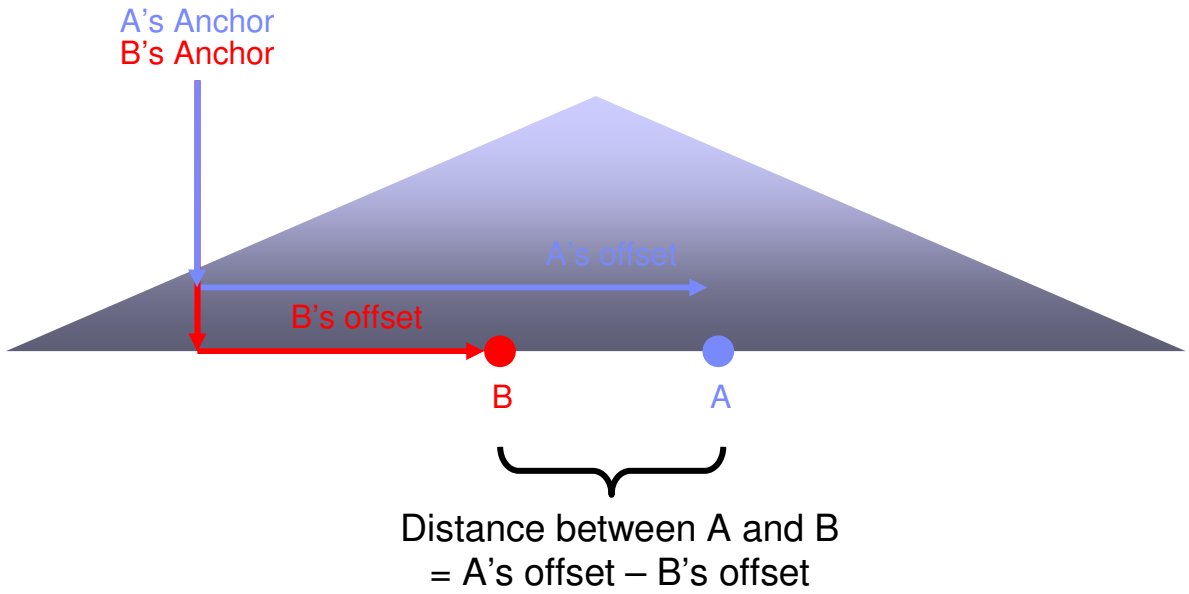
Index-Independent Relative SISCAN Locations



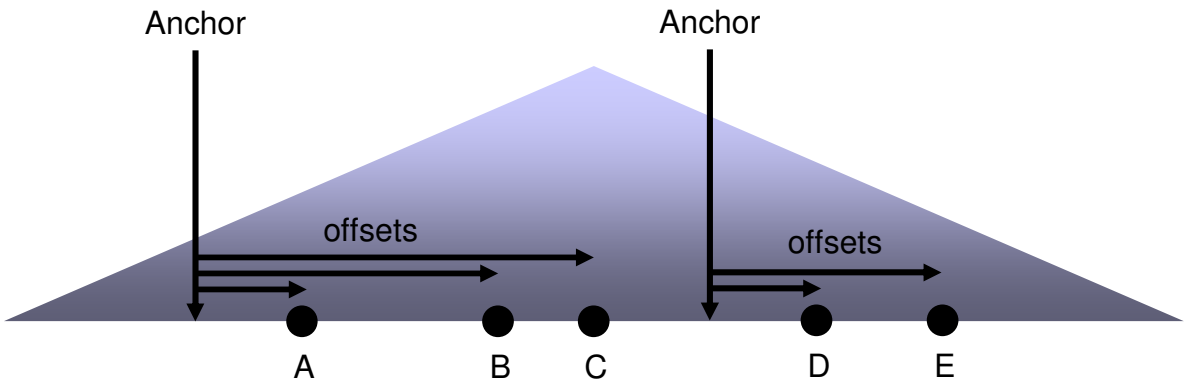
Index-Independent Relative SISCAN Locations



Index-Independent Relative SISCAN Locations

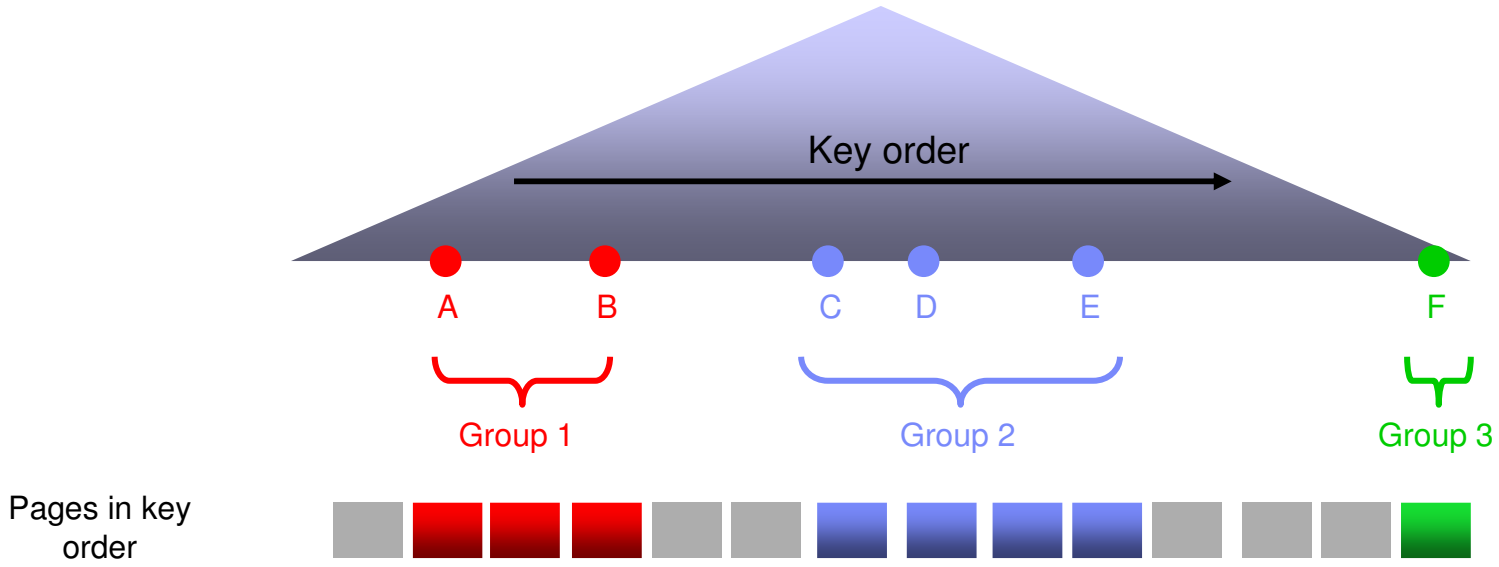


Index-Independent Relative SISCAN Locations

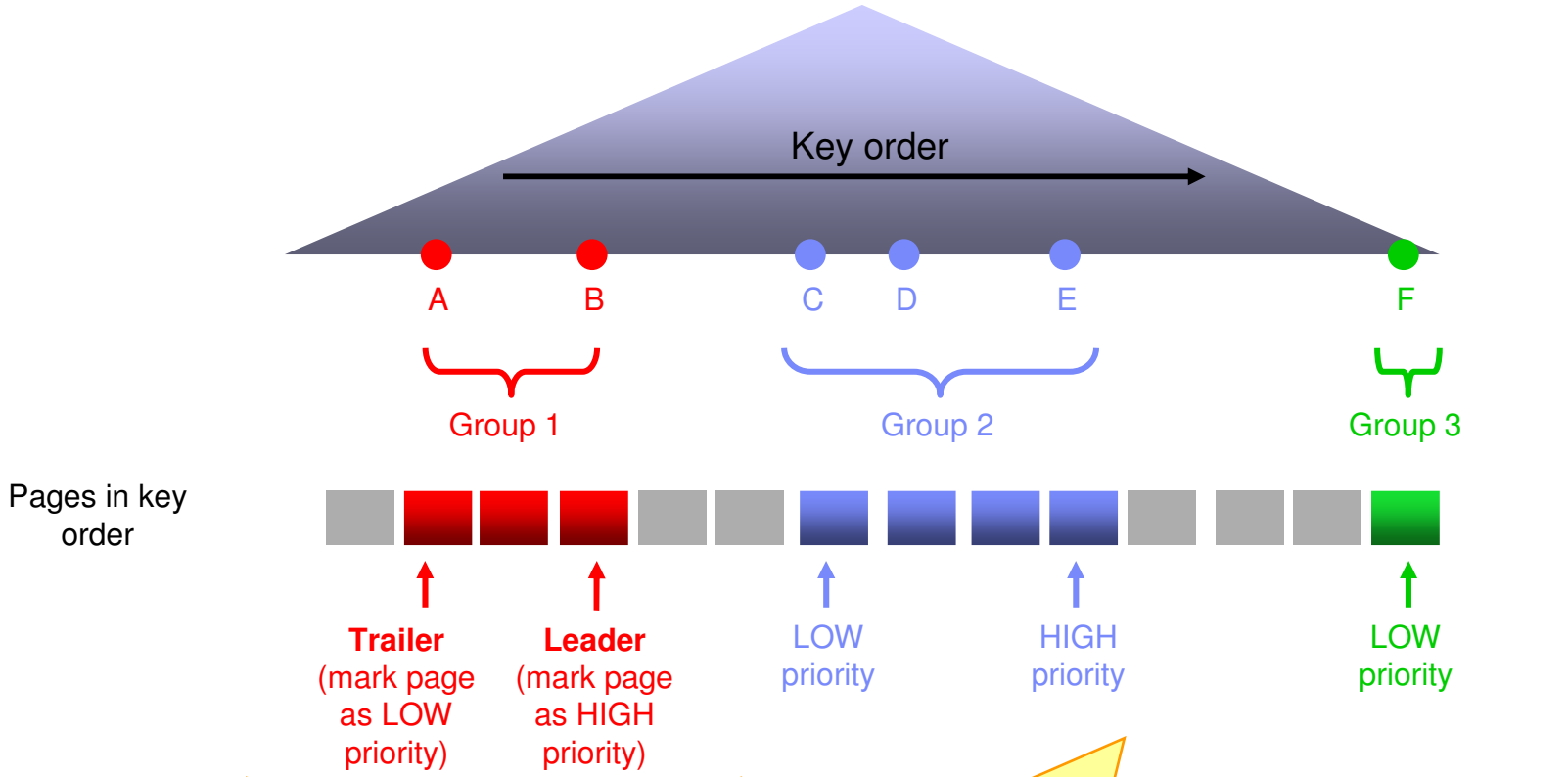


Partial ordering between SISCANs without details of the index structure

“Bufferpool-independent” SISCAN-aware Caching

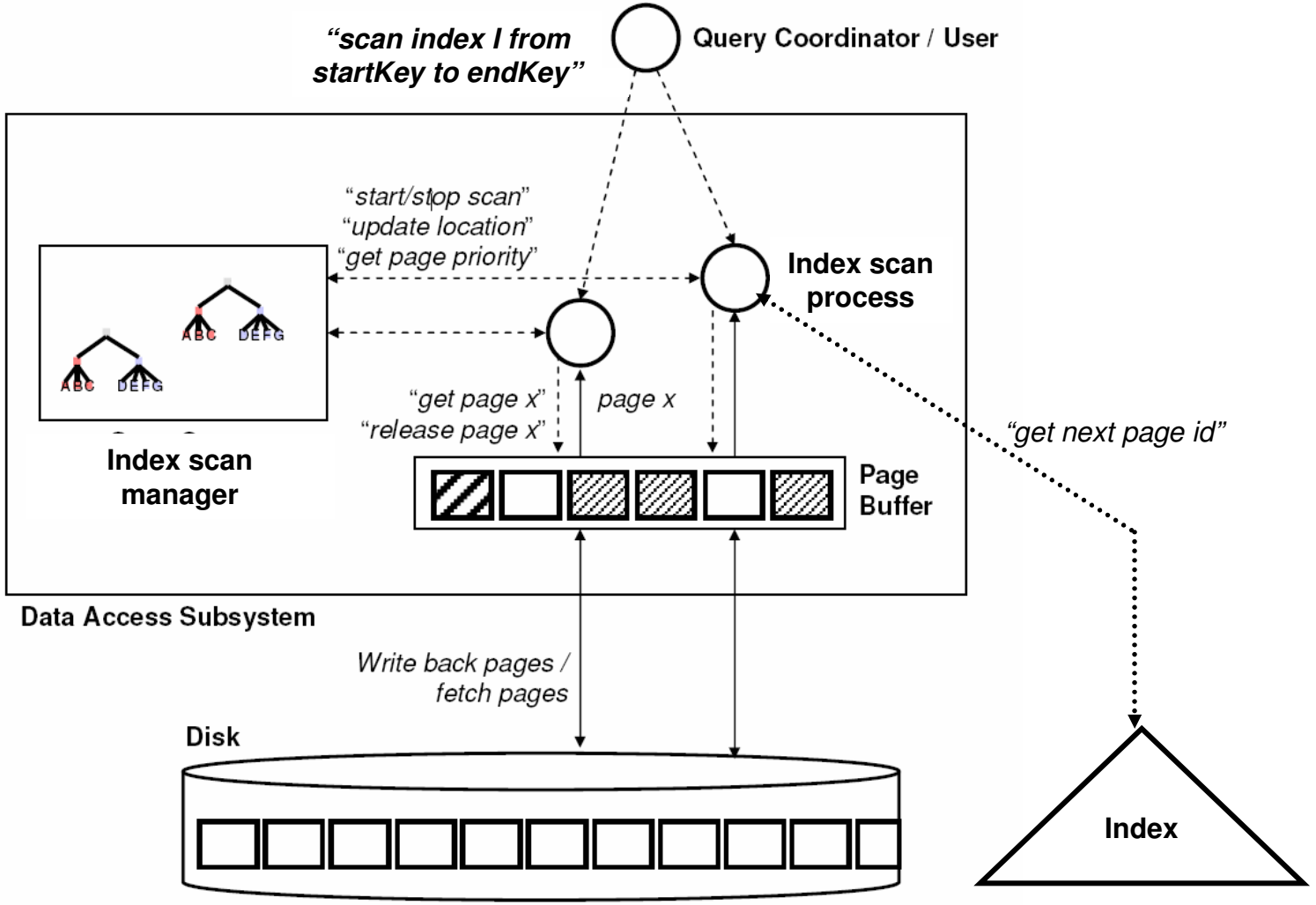


"Bufferpool-independent" SISCAN-aware Caching

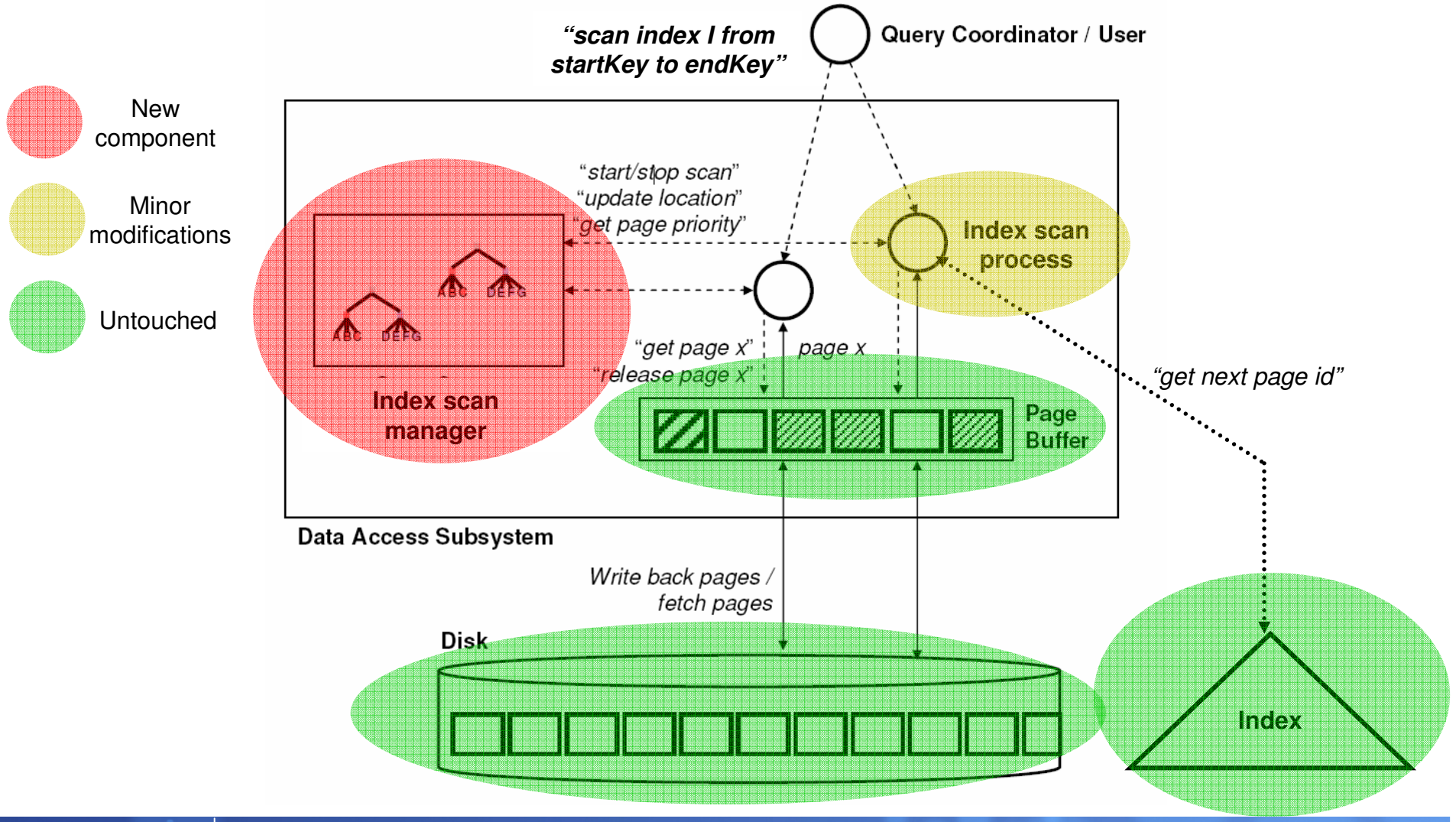


Don't need to change caching algorithm; need only LOW/HIGH priority hints; Details similar to [Lang et al., ICDE '07]

Architectural Changes



Architectural Changes



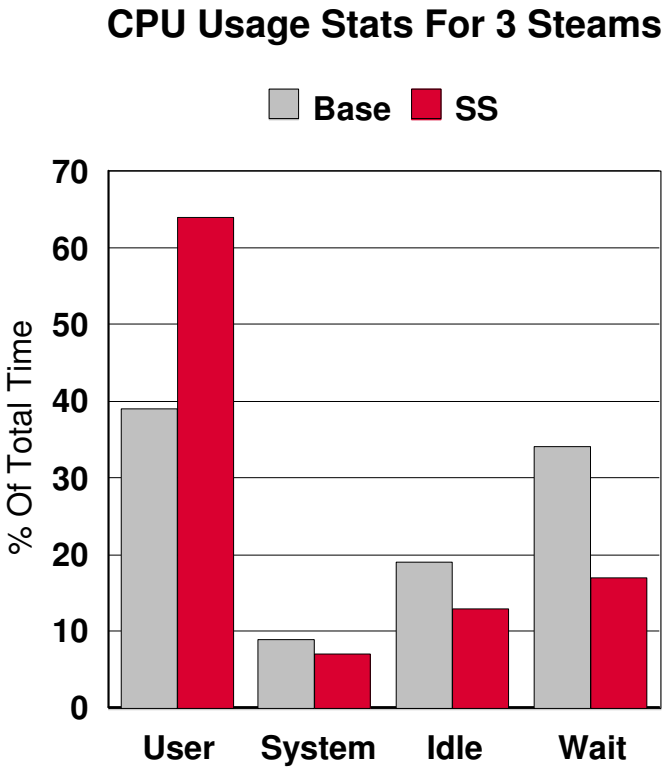
Outline

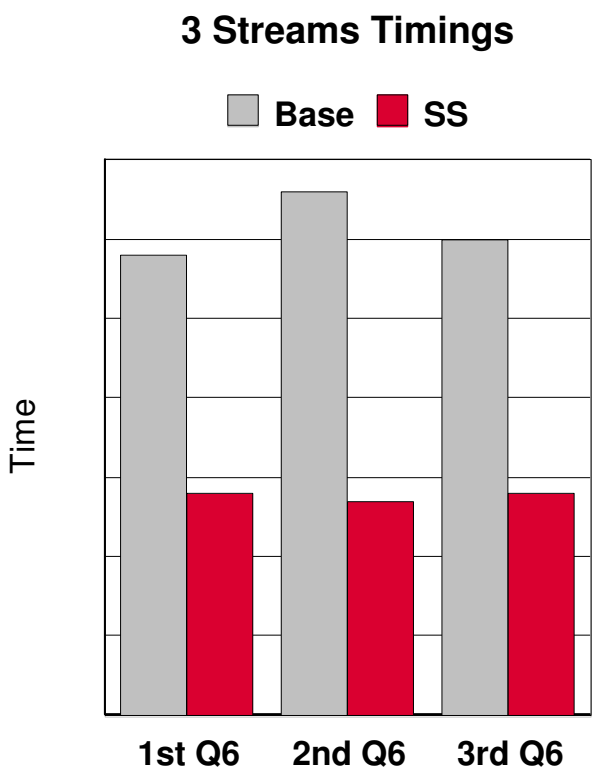
- Current Index Scan Architecture
- SISCAN – "Circular" Index Scan
  - Placement
  - Speed Control
- Implementation Issues
  - Index-independent Relative SISCAN Location
  - "Bufferpool-independent" SISCAN-aware Caching
- **Experimental Results**
- **Conclusions**

Experimental Results Setup

- Platforms:
  1. HP Integrity rx5670 (4 Itanium2 proc/1GHz, HP-UX, 15GB, FAStT)
  2. 8-node p660 cluster (4 PowerPC/600MHz, AIX, 8GB, 16 SSA disks)
- 100GB TPC-H database
- Bufferpool size ≈5% of DB size
- Standard MDC indexes / no hand-tuning

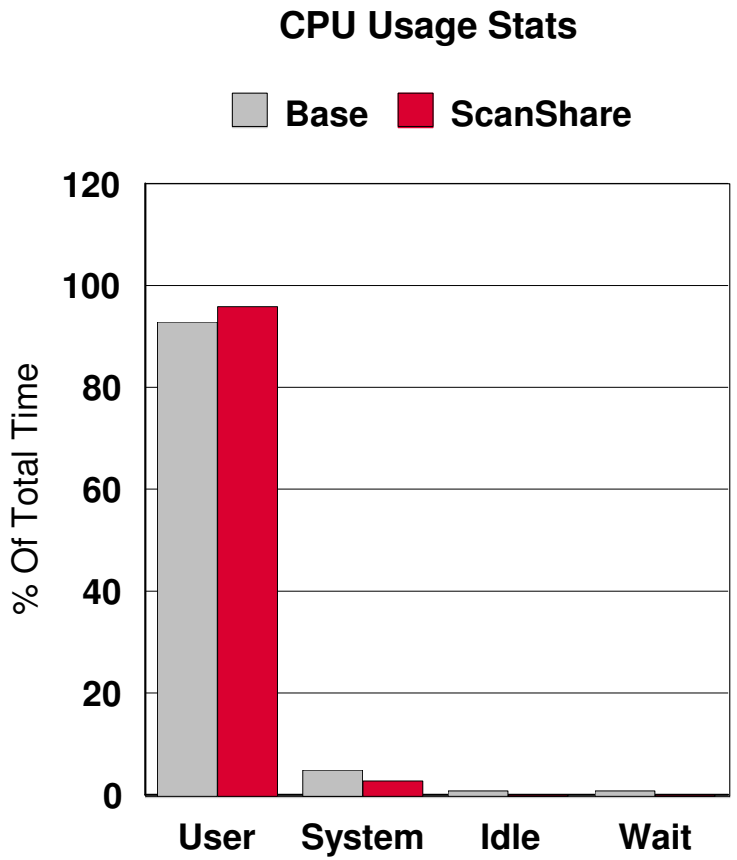
Staggered Q6 (I/O intensive)

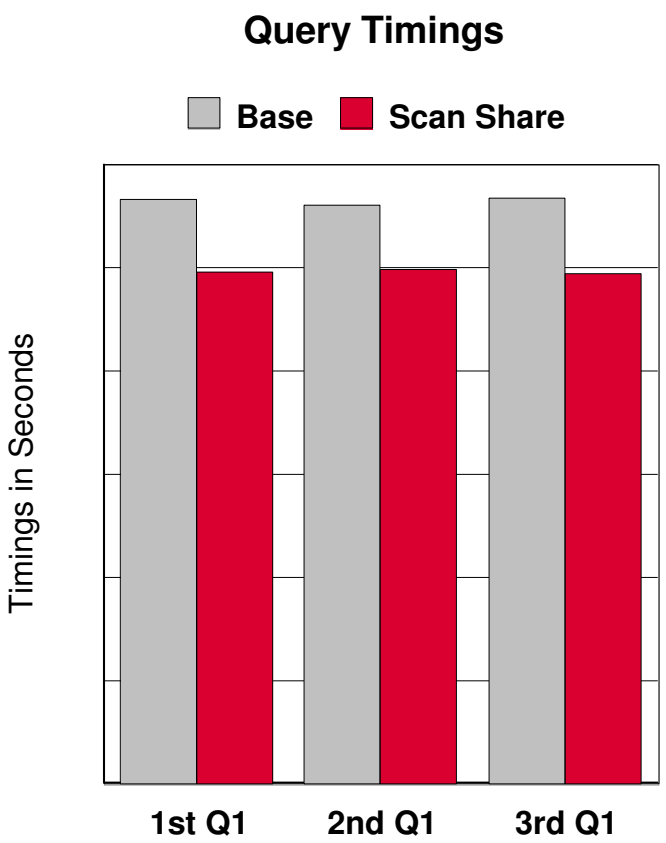




I/O wait reduced by 50%;
More than 50% gain in response time

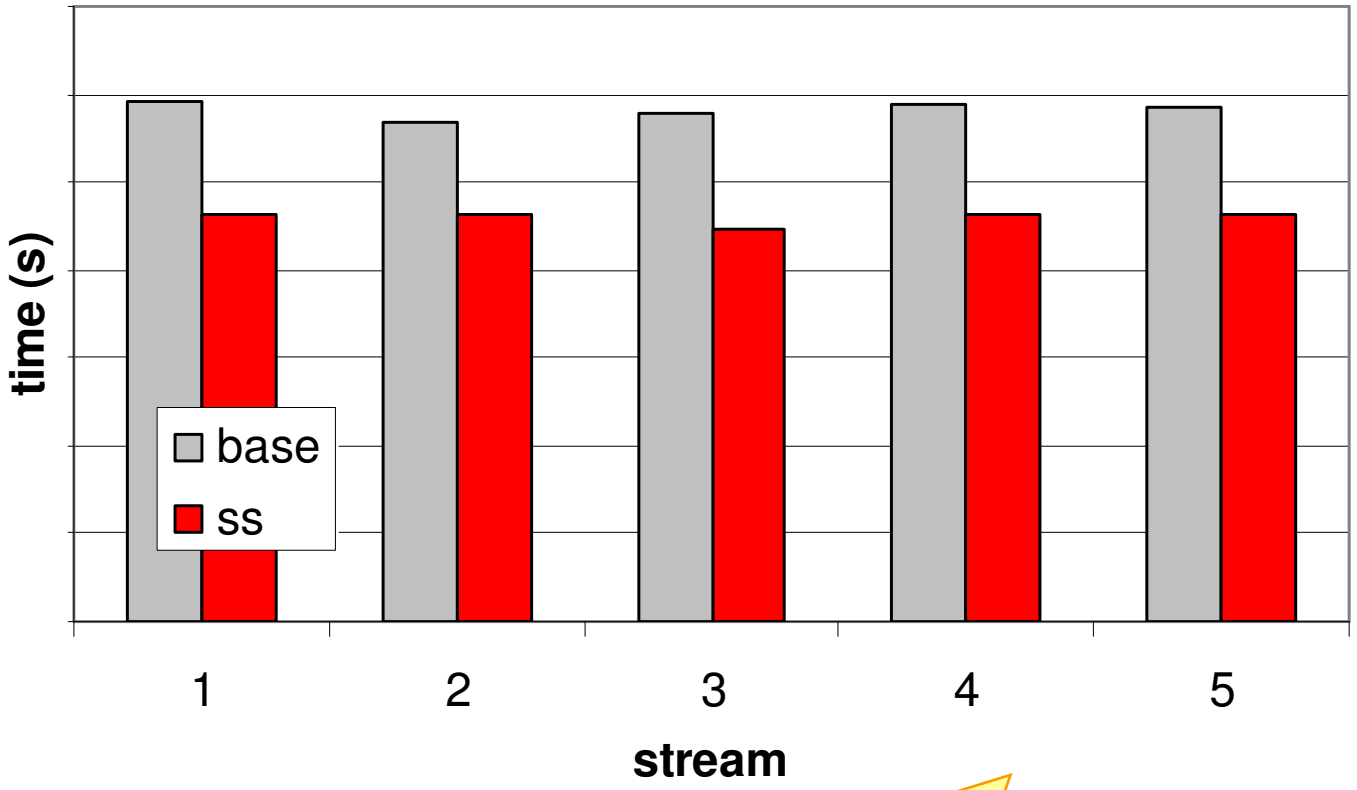
Staggered Q1 (CPU intensive)





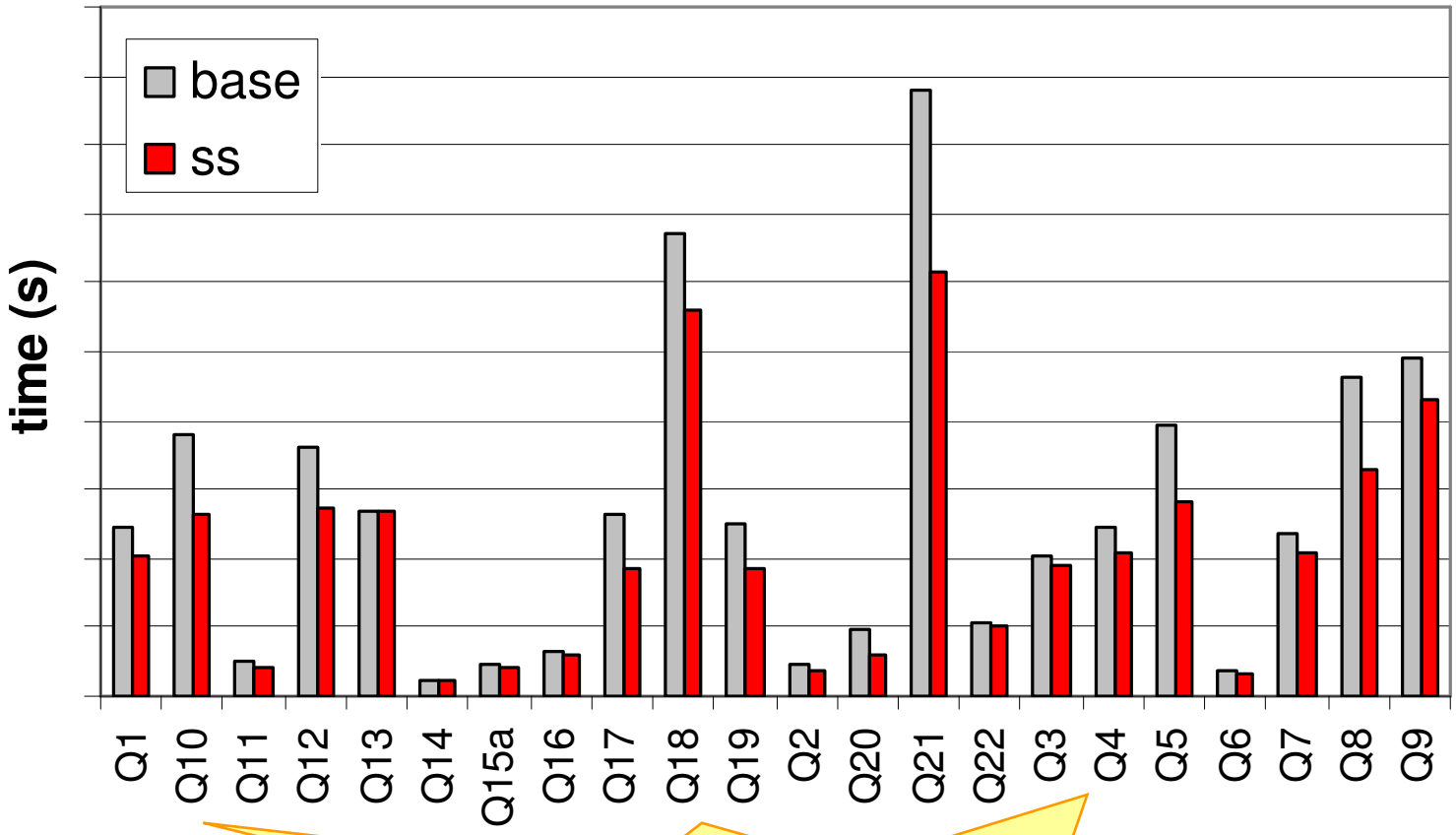
Noticeable reduction in response time even for CPU bound queries

TPC-H Throughput: Per-stream Gains



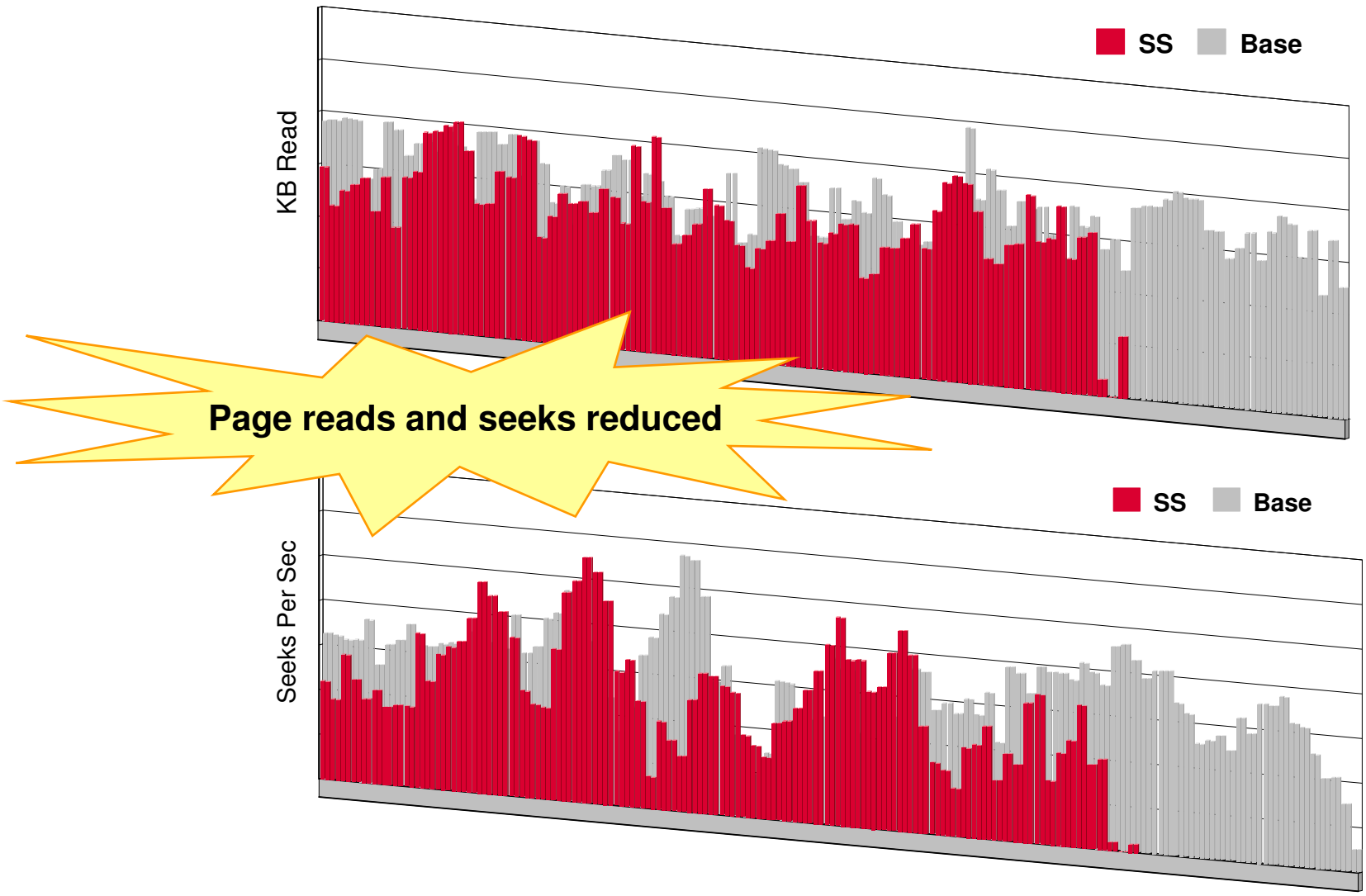
20% reduction in response time for all streams

TPC-H Throughput: Per-Query Gains



No deterioration for any query

TPC-H Throughput: Disk Behavior



Conclusions

- Mechanism for better cache reuse and reduced I/O to increase throughput and reduce latency for ad-hoc index scan-heavy multi-query workloads
- Inter-SISCAN cache locality improved via:
  - Starting new SISCANs near similar (speed/key range) running scans
  - Speed control of SISCANs to reduce drift
  - SISCAN-based priority hints to bufferpool manager
- Fulfills requirements:
  - Can handle dynamic “heterogeneous” workloads
  - Easy integration in architecture

Thank you!

Contact: langc@us.ibm.com

Database Research Group
IBM T.J. Watson Research Center

http://www.research.ibm.com/scalabledb/