#SHAREorg

SHARE
Technology · Connections · Results

Using List Prefetch Optimizer and Solid State Disk to Improve DB2 Perf and Avoid DB2 Reorgs

Jeffrey Berger
IBM

August 6, 2012
Session 11294

SHARE in Anaheim 2012

#SHAREorg

Abstract

Efficient I/O operations is the key ingredient of a well performing database management system. Ensuring optimal I/O performance is a time consuming and resources intensive work that regularly includes frequent data and index reorganization. Recent enhancements in System z and disk technology, combined with DB2 10 for z/OS features deserve a fresh look at how to achieve optimal I/O performance without continuous monitoring and tuning and with greatly reduced need for costly and obtrusive database reorganization.

#SHAREorg

Disclaimer

SHARE in Anaheim 2012

Performance Disclaimer

This document contains performance information based on measurements done in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the numbers stated here.

#SHAREorg

SHARE Technology · Connections · Results

Agenda

- Disorganized data versus organized data
- New disk technology enhancements, trends in System z
- DB2 10 improvements
- Future DB2 strategy to reduce the need for Reorgs
- REORG
  - The pain of REORG
  - Why do we use it
  - Which of these reasons can be alleviated if disorganized tables and indexes perform better
- Member Cluster

SHARE in Anaheim 2012

DB2 Prefetch Techniques

- Index scan
  - Organized indexes: dynamic prefetch (otherwise known as "sequential detection")
  - Disorganized indexes
    - Prior to DB2 10, DB2 did synch I/Os
    - DB2 10 uses list prefetch
- Index-to-data access
  - High cluster ratio (organized data)
    - Dynamic prefetch for clustered pages, synch I/O for unclustered pages
  - Low cluster ratio (disorganized data)
    - DB2 Optimizer may choose a sorted RID list and use list prefetch on that RID list

DB2 10 for z/OS Enhancements

- Index scans
  - Progressive prefetch quantity (read 8 pages, then 16 pages, then 32)
  - First dynamic prefetch I/O may be triggered after 5 Getpages
  - Use list prefetch for disorganized indexes
- Index-to-data access, RID list scans
  - The RID pool may spill over to a work file instead of falling back to a table scan
  - The default RID pool size (MAXRBLK) increased from 8 MB to 400 MB
- Index-to-data access, sequential detection
  - Row-level sequential detection, may trigger first dynamic prefetch I/O after 5 rows
  - Progressive prefetch quantity

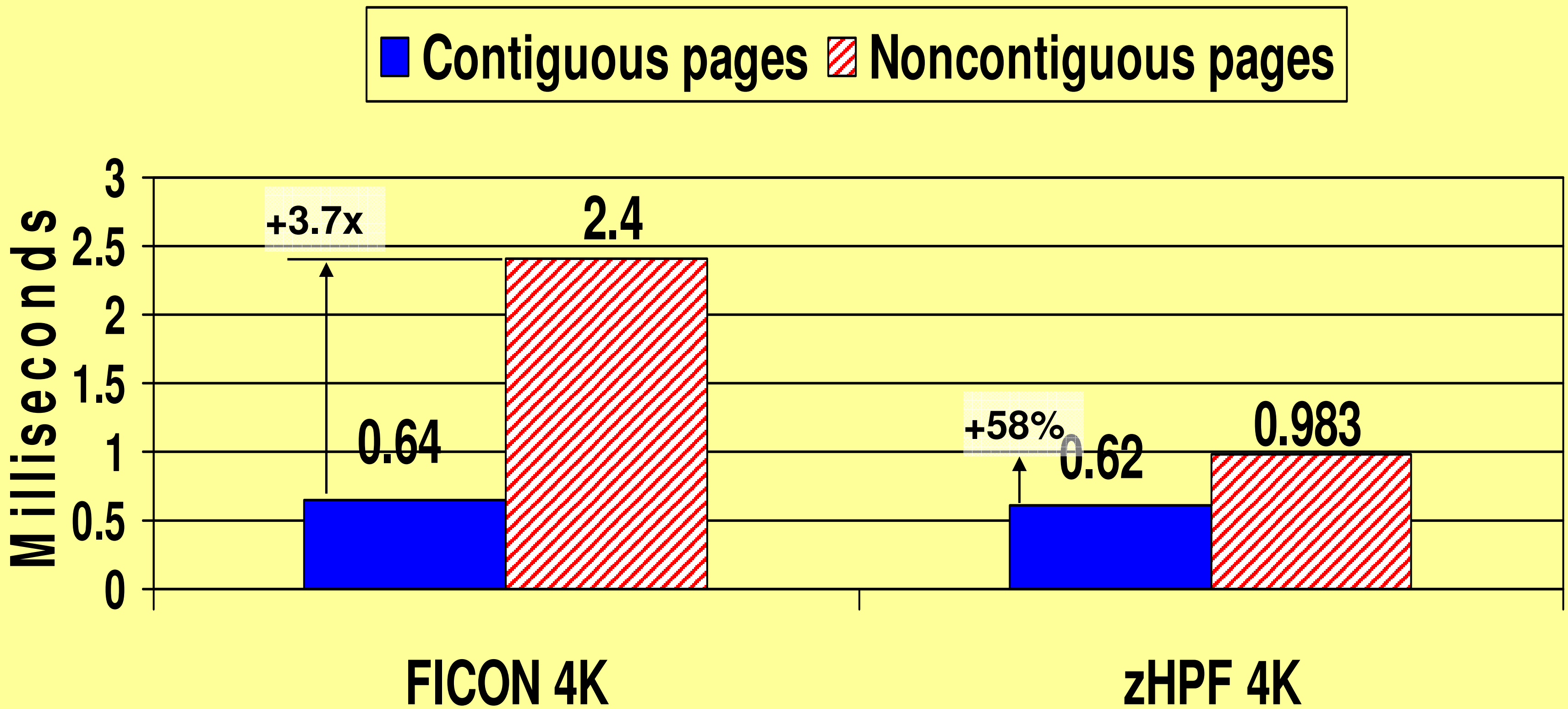
Hardware Positioning

- Solid State Disks
  - Introduced in 2009
  - Sub-milliseconds synch I/O response time
  - No mechanical parts, insensitive to data/index organization
- zHPF
  - Introduced in 2009
    - Initially limited to reads and update writes <=64K contiguous
    - The 64K limit was removed in the z196
    - In 2011, zHPF made applicable to all DB2 I/Os
      - *Format writes and list prefetch means faster DB2 utilities and queries*
- FICON Express 8S
  - Introduced in 2011 with z196 GA2 processor
  - Optimized for zHPF

List Prefetch

- List prefetch I/O is unique to z/OS
- zHPF list prefetch introduced in Nov., 2011
  - DB2 list prefetch I/Os are made eligible for zHPF
  - Improves channel performance of DB2 list prefetch
  - Requirements:
    - z196 processor and z/OS R11 or above (with PTFs)
    - IBM DS8700 or DS8800 with R6.2 or above
    - Non-IBM storage does not yet support zHPF list prefetch
- List Prefetch Optimizer (LPO) is the DS8000's caching algorithm, introduced in R6.2. LPO requires zHPF.
  - Improves the cache hit ratio by taking advantage of RAID 5 architecture to increase I/O parallelism

FICON Express 8S, z196, DS8800
I/O response time for 128K (cache hits)

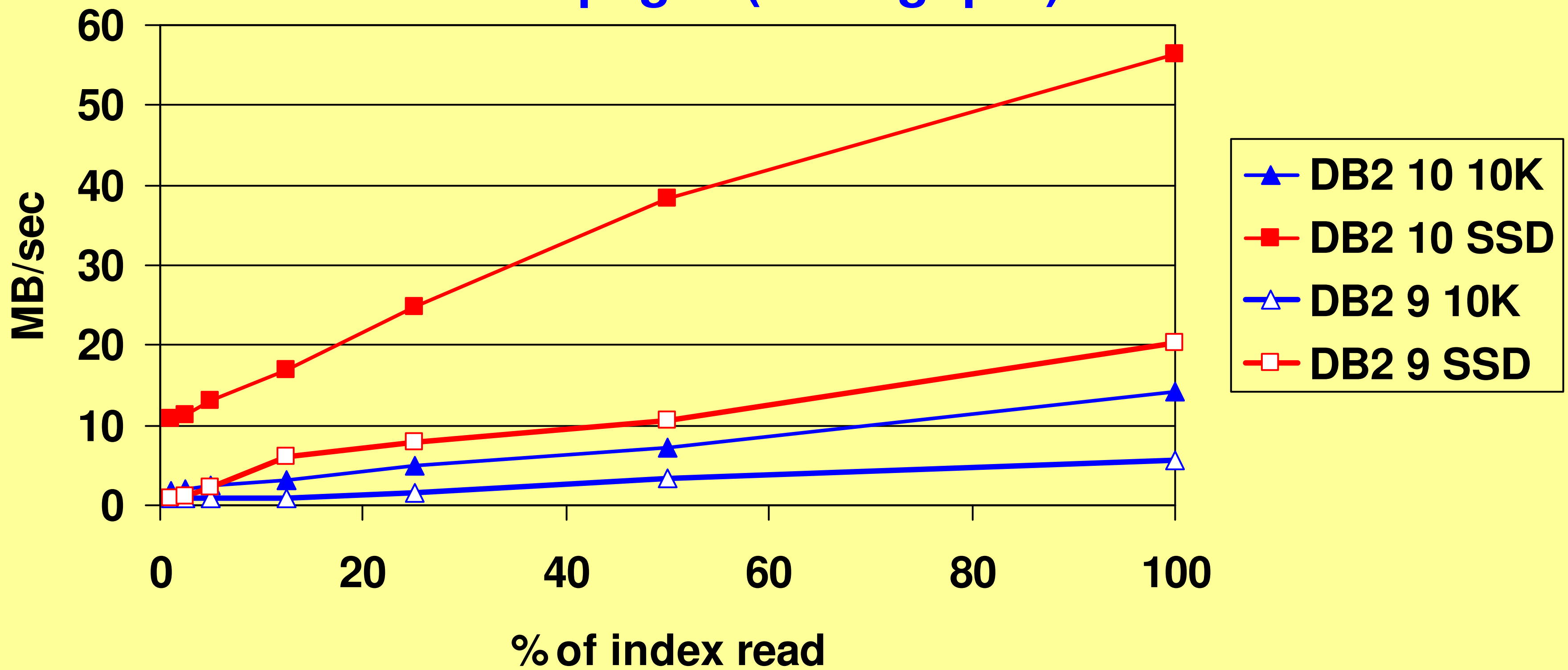


#SHAREorg

Index Scans

SHARE in Anaheim 2012

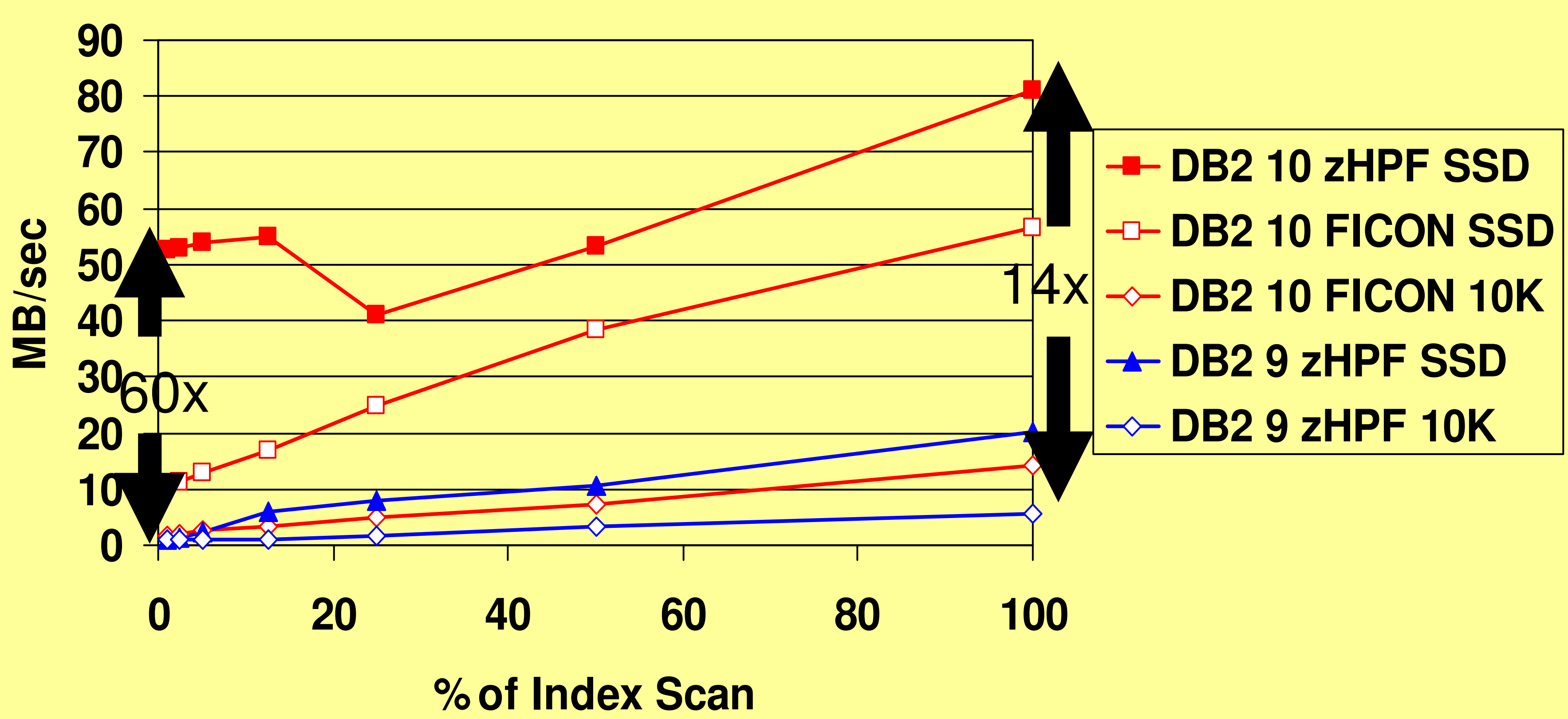
Disorganized index scan, cold cache DB2 10 versus DB2 9 with FICON 4K pages (throughput)



- DB2 10 is 2x to 10x faster

#SHAREorg

Disorganized index scans, cold cache, 4K pages Throughput



- Migration from 10K HDD to SSD ... zHPF is 14x to 60x faster

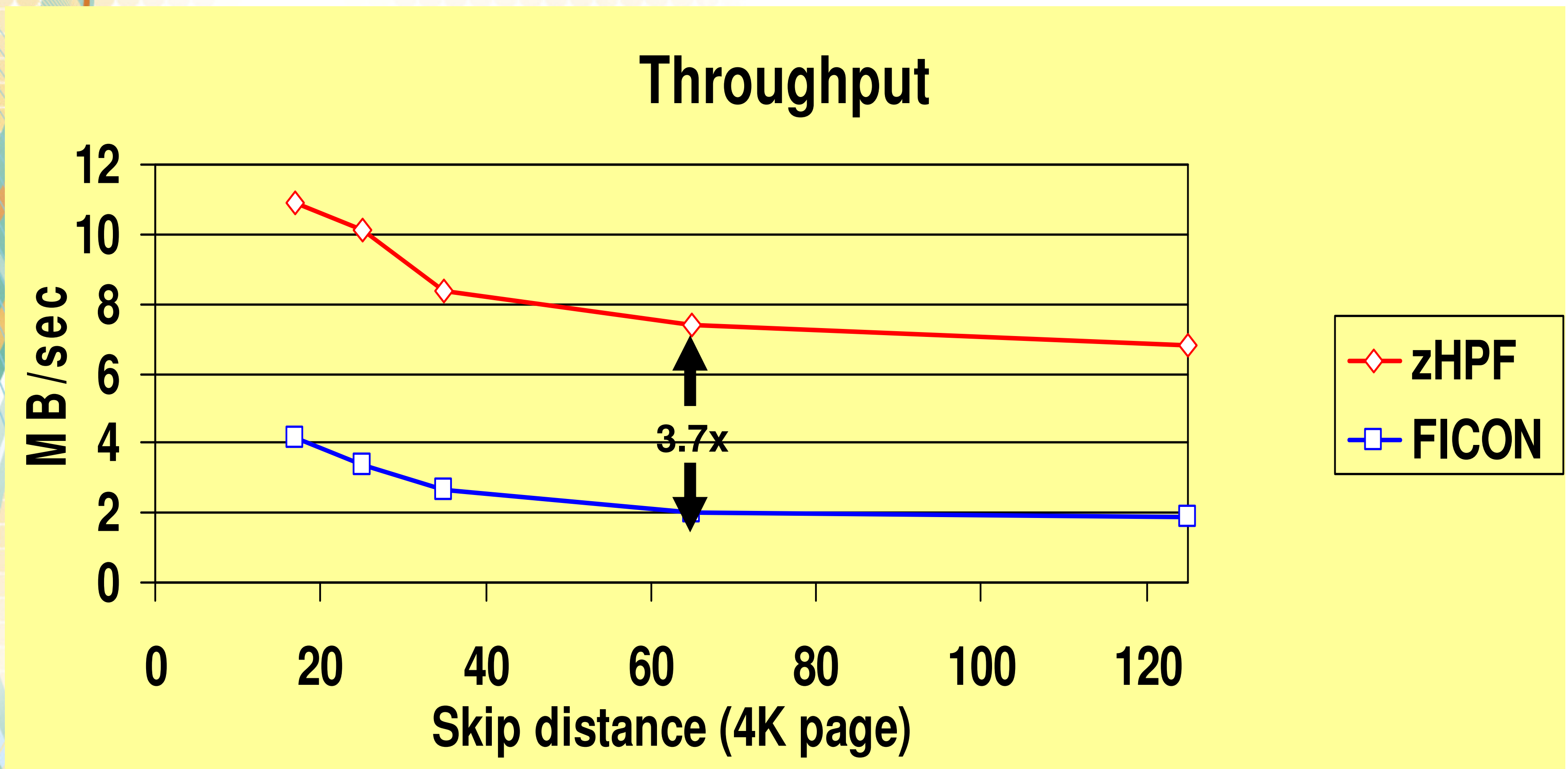
#SHAREorg

Index-to-data access

Sorted RID List Scans

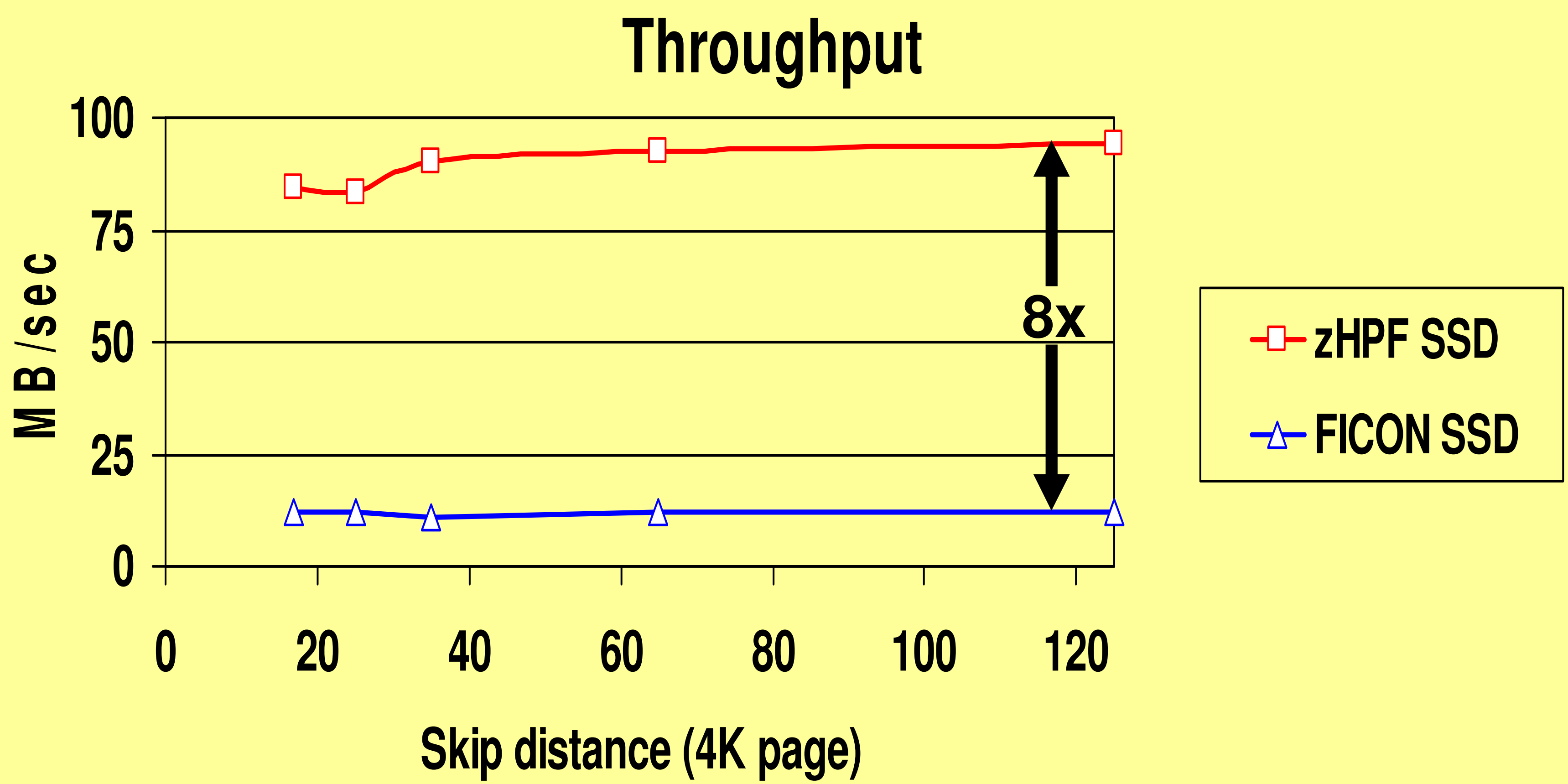
SHARE in Anaheim 2012

Sparse skip sequential using list prefetch 10K HDD

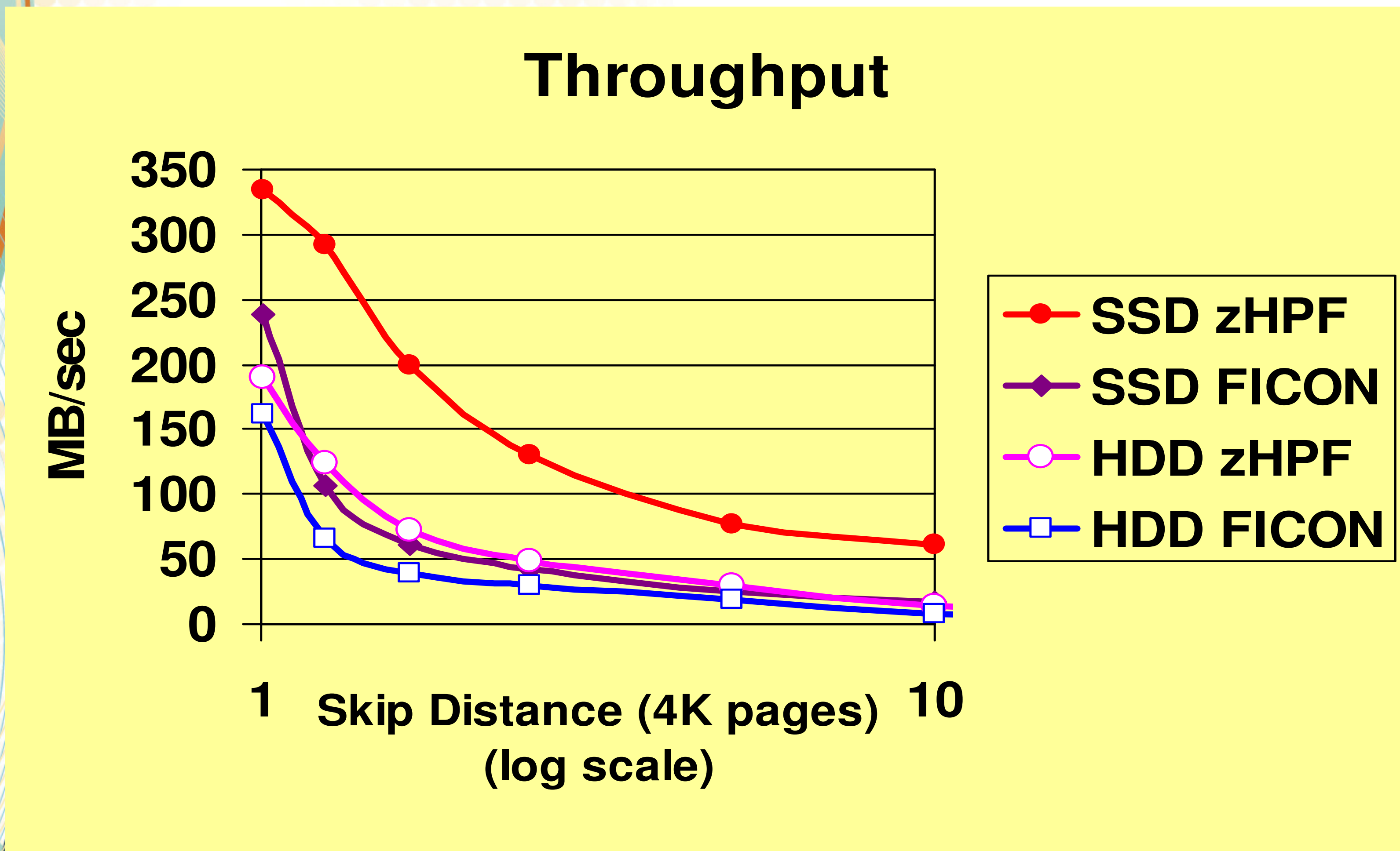


SHARE Technology · Connections · Results

Sparse skip sequential using list prefetch Solid State Disks



Dense skip sequential using list prefetch



- Migr[illegible]on from 10K HDD to SS[illegible]

#SHAREorg

SHARE
Technology · Connections · Results

Dynamic Prefetch

and

Sequential Detection

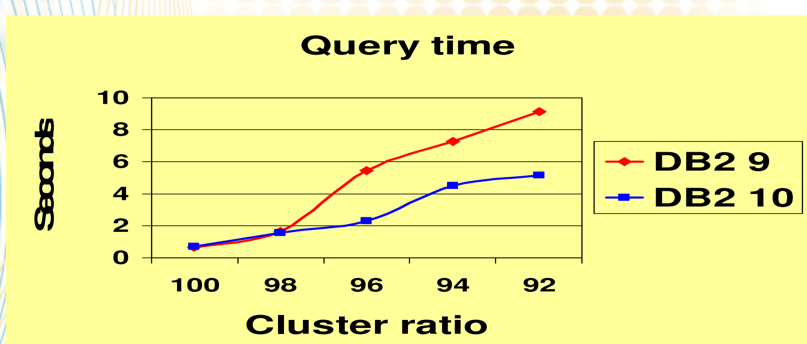
SHARE in Anaheim 2012

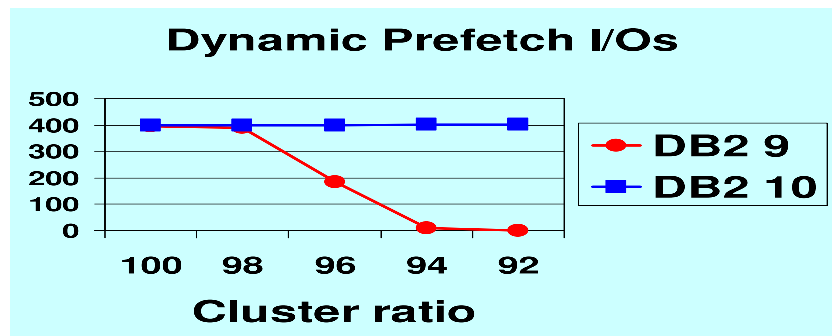
Dynamic prefetch: Index—>Data Range Scan

Row size = 49 bytes, page size = 4K (81 rows per page)

Test case	Cluster ratio	Cardinality	NPAGES
1	100%	20,000,000	253167
2	98%	20,200,000	256024
3	96%	20,400,000	258882
4	94%	20,600,000	261740
5	92%	20,800,000	264598

Read 10% of the rows in key sequential order





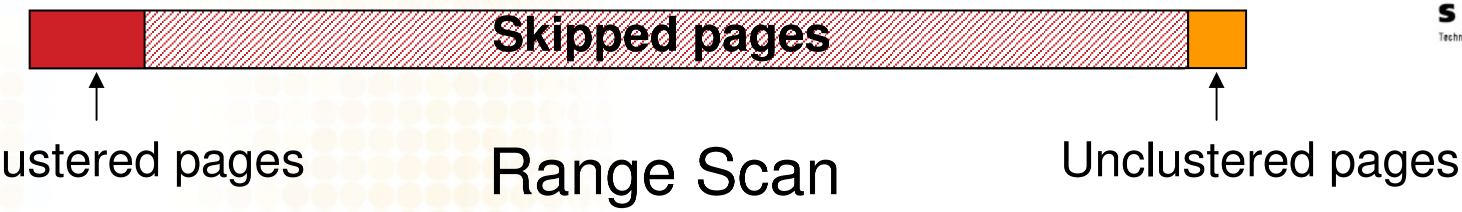
- Row level sequential detection (RLSD) preserves good sequential performance for the clustered pages

Dynamic Prefetch

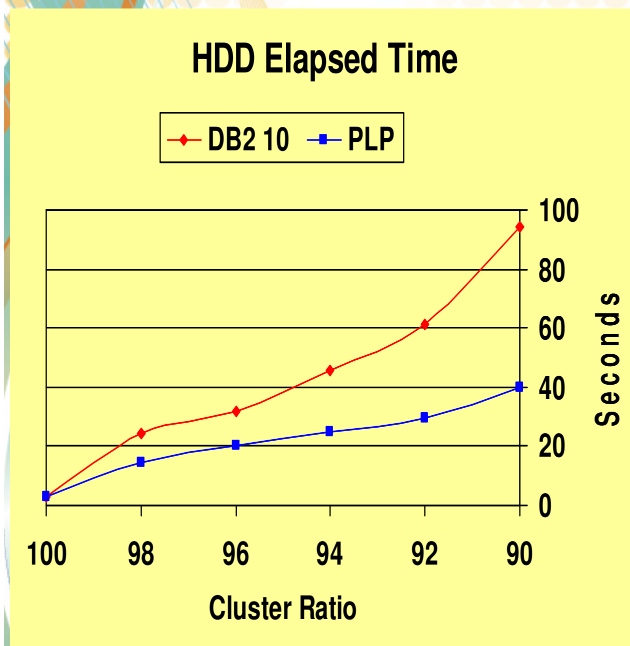
- DB2 10 introduced Row Level Sequential Detection and progressive prefetch quantity
  - When the number of rows per page is high (e.g. >40), RLSD preserves sequential I/O of clustered pages
  - Prefetch may be triggered after 5 *rows*, instead of 5 *pages*
  - First prefetch I/O reads 8 pages, then 16, then 32 pages thereafter
- Strengths of dynamic prefetch (compared to RID list scan)
  - Avoids some result set sorts when a query specifies ORDER/GROUP BY based on the index key
  - Avoids the storage requirements of a RID pool
- Deficiencies of dynamic prefetch
  - Sometimes many synchronous I/Os
  - Sometimes wastes buffers

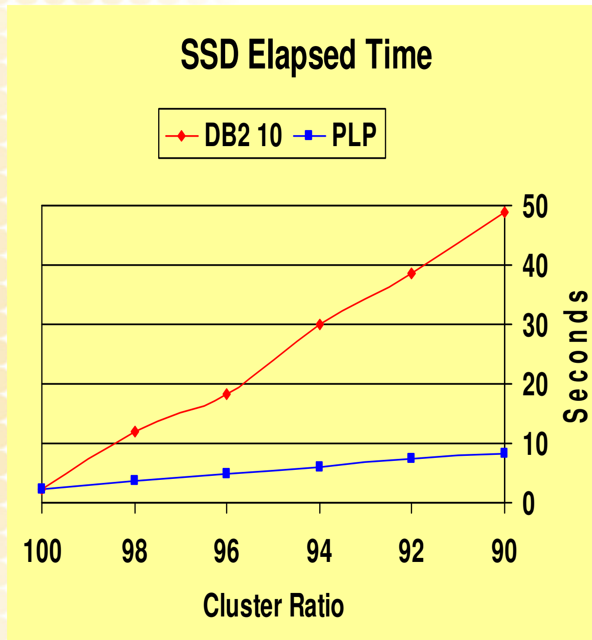
Piecemeal List Prefetch (PLP)

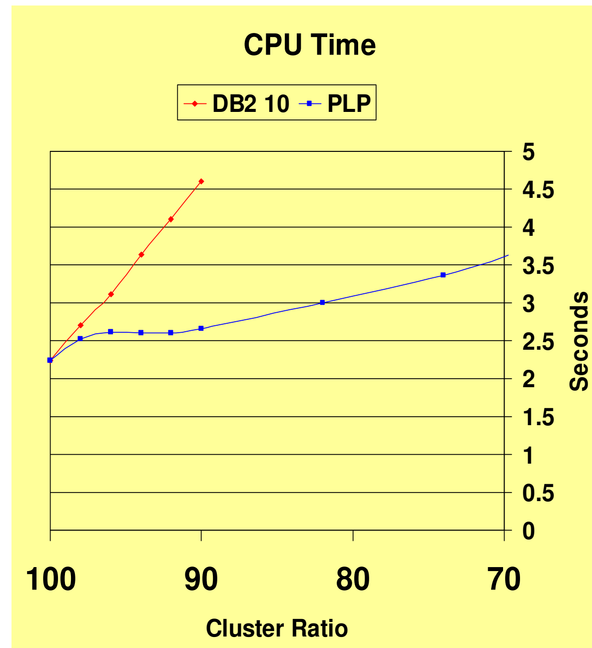
- Possible future strategy
- Performance objectives
  - Range scans
    - Elapsed time savings and CPU savings when the cluster ratio is slightly degraded or catalog statistics are out-of-date
  - Skip sequential access
    - Elapsed time, CPU savings
    - DB2 buffer savings, could improve the OLTP buffer hit ratio
- Hybrid of dynamic and list prefetch
  - Dynamic prefetch for clustered pages
  - List prefetch for unclustered pages and skip sequential, avoids synch I/Os and avoids wasting buffers
- Avoid using a RID pool
- Preserve index ordering of the rows
- Will not be supported for CURRENT DATA YES or ISOLATION RR



- 100 byte rows, MAXROWS 40
- Range scan reads 10% of the clustered rows and 10% of the unclustered rows
- z196, DS8800, SSD, List Prefetch Optimizer







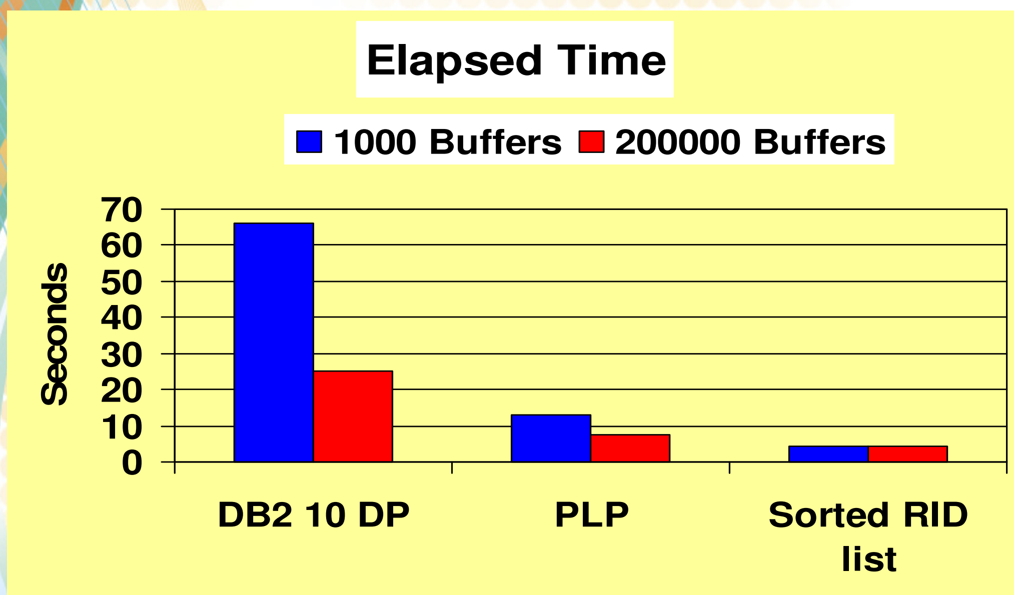
SHARE Technology • Connections • Results

Clustered pages

Range Scan

Unclustered pages

- 100 byte rows, MAXROWS 40
- Cluster ratio 90%, 1 unclustered page for every 20 clustered pages
- Range scan reads 10% of the clustered rows and 10% of the unclustered rows
- z196, DS8800, SSD, List Prefetch Optimizer

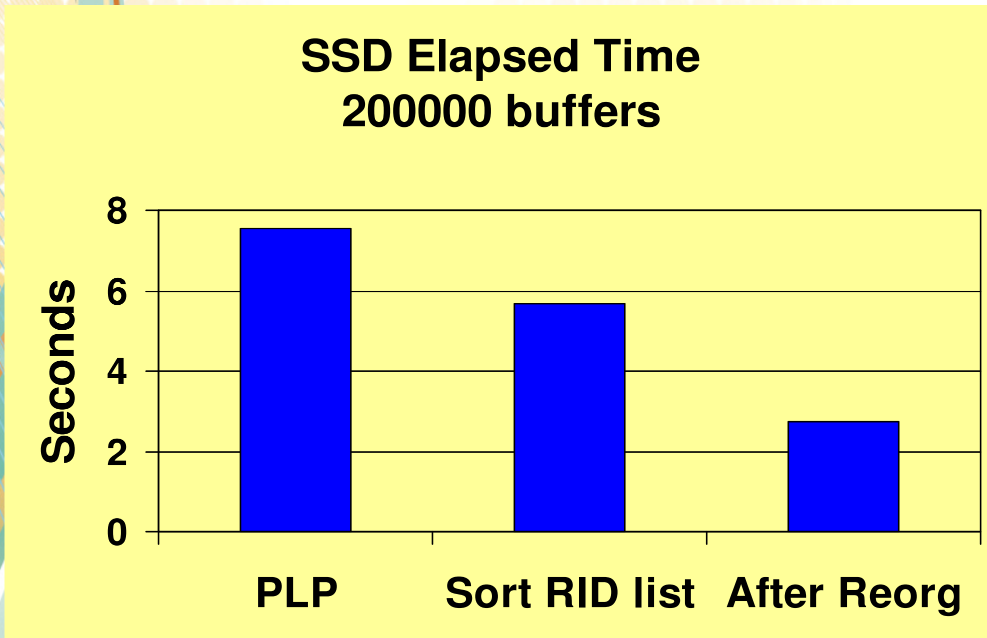


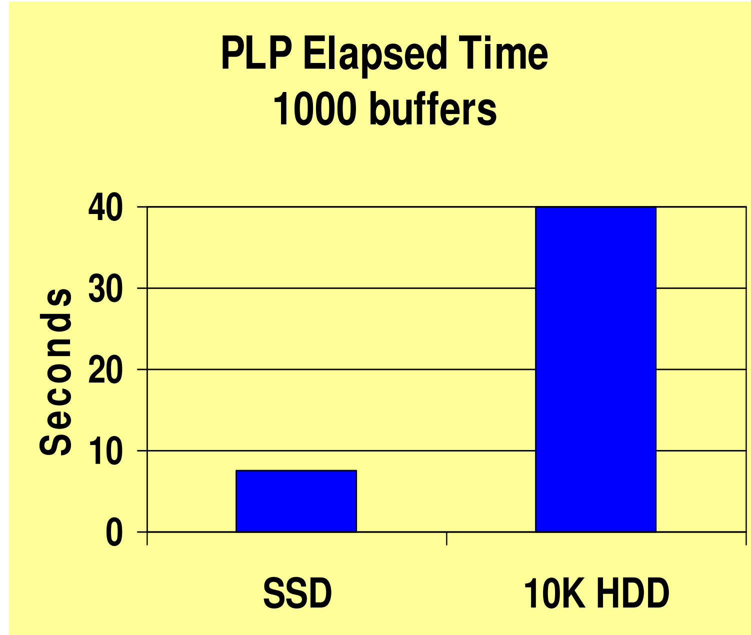
Conclusions:

- PLP closes the gap between vanilla Dynamic Prefetch and a sorted RID list, without the need of a RID pool
- The deficiencies of PLP relative to a sorted RID list are mitigated by a large buffer pool

This query required 791 RID blocks, about 26 MB. In contrast, PLP uses only 32 KB.

… Range Scans





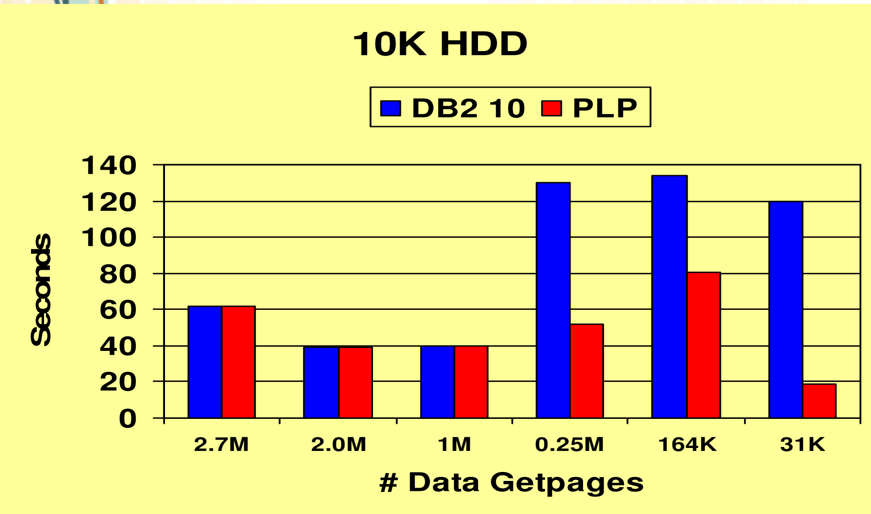
- Running RUNSTATS will encourage DB2 optimizer to choose a Sorted RID list. That is still the best strategy to avoid Reorgs.
- However, PLP can mitigate the performance problems until RUNSTATS can be run, or until REORG can be run.
- Since the sorted RID list in this case is largely sequential, SSD is not critical for a sorted RID list, but it is critical for Piecemeal List Prefetch

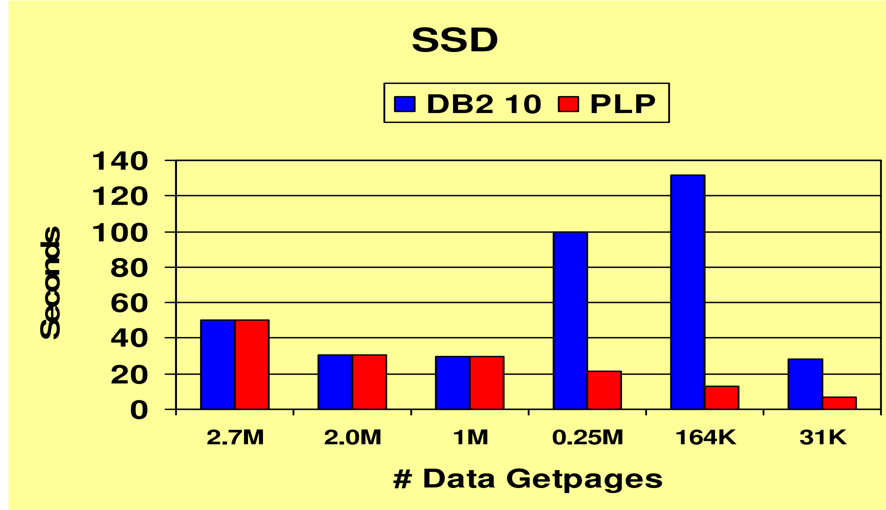
Skip Sequential Test Cases

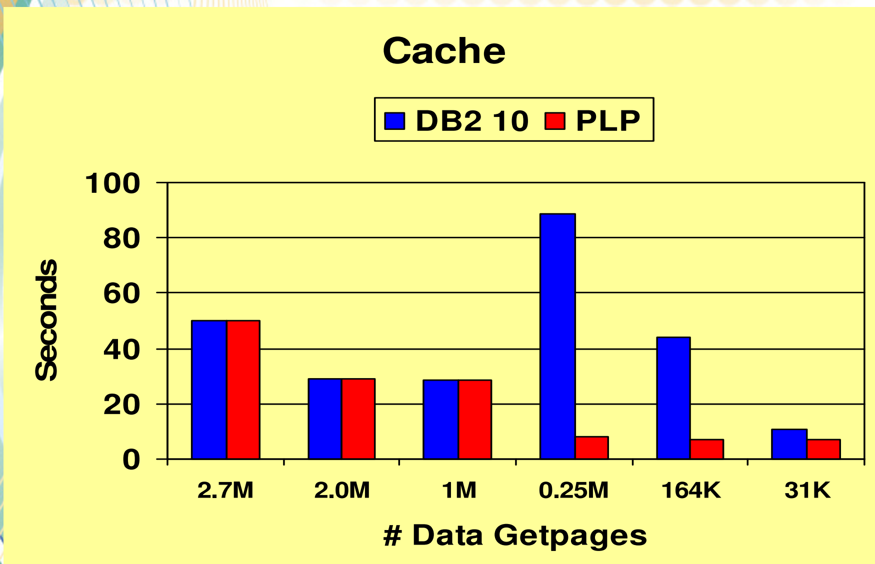
- Organized index and data
- Index key is 8 bytes
  - All queries do 445,432 index Getpages
  - Cluster ratio is 100%
- Data
  - 108 byte rows, 37 rows per page
  - Getpages are uniformly distributed and spread across 2 million data pages in 6 test cases:
    1. 2.7M Getpages (process all 37 rows per page, i.e. range scan)
    2. 2M Getpages (process 1 row per page)
    3. 1M Getpages
    4. 250,000 Getpages
    5. 164,286 Getpages
    6. 31,250 Getpages
- SELECT SUM(non-indexed column) WHERE KEY1<=x AND KEY2>=y

#SHAREorg

Elapsed time







- With case 4 and 5, DB2 10 allocates 2.7MB buffers, whereas PLP only allocates one buffer per Getpage

Conclusions:

- PLP improves the performance and, in some cases, saves buffer storage
- SSD far out-performs HDD
- REORG doesn't help this case, although a sorted RID list would

REORG

Why is it painful and why do we do it

The Pain of REORG

- Consume large amounts of I/O and CPU resources
- Can impact transaction response times when trying to break in to switch to the “shadow” objects
  - Completing the REORG switch phase is sometimes impossible without quiescing workloads
- REORG makes it harder to take advantage of storage tiering solutions like IBM’s Easy Tier
- Must be scheduled and monitored
- Can flood wide area networks (WAN) with changed traffic when disaster recovery replication is used

REORG

- What problems does REORG actually solve?
  - Reclaim space
  - Re-establish (reserve) distributed free space for insert
  - Clean up “indirect references”
  - Restore data row clustering which has deteriorated
  - Re-establish optimal performance and logging after alter schema change
  - Materialize ‘deferred alters’ which are pending (V10)
- What problems does REORG INDEX solve?
  - Reclaim space
  - Re-establish (reserve) distributed free space for insert
  - Organize the leaf pages so that an index scan will be sequential
  - Clean up “pseudo deleted” RIDs to improve query processing
  - Materialize ‘deferred alters’ which are pending (V10)

.....REORG

- DB2 10 with LPO and SSD largely eliminate the problems of a disorganized index
- LPO and SSD, combined with PLP or traditional RID list scans, largely mitigate the problems of a “sub-optimal” cluster ratio
- Sequoia will mitigate the problems of psuedo-deleted index entries
- Indirect references will persist as a problem
  - Indirect references occur when a variable length row (or compressed row) is updated, the row length increases, and the row no longer fits on its original page
  - Indirect references cause synchronous I/Os

.....REORG

- Non-performance reasons will always remain
  - Reclaiming space
  - Deferred alters
  - Restore clustering in order to optimize the buffer hit ratio

OLTP buffer hit ratios

- Sometimes the buffer hit ratio is affected by the cluster key.
  - For example, if the cluster key is based on time, and the most recent inserts are most likely to fetched
- A big decrease in the buffer hit ratio can have a big effect on CPU time, no matter how fast the I/Os is.
  - Adding memory may help compensate for a loss of clustering
- Prefetch cannot help singleton SELECTs

Other option if we abandon clustering

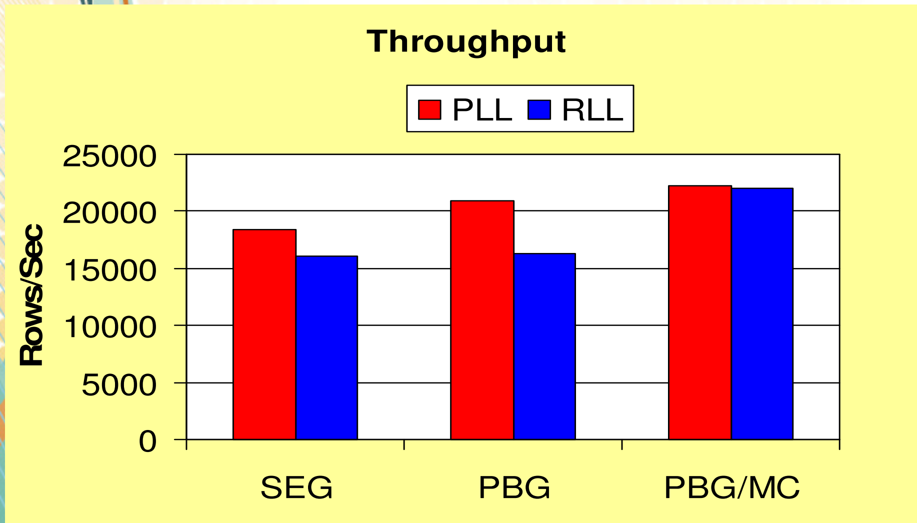
- MEMBER CLUSTER (MC) organization
  - Very useful for improving the performance of highly concurrent inserts in data sharing
  - DB2 9 supported MC for classic Partitioned Table Space only
  - DB2 10 provides for all Universal Table Spaces (PBR and PBG)

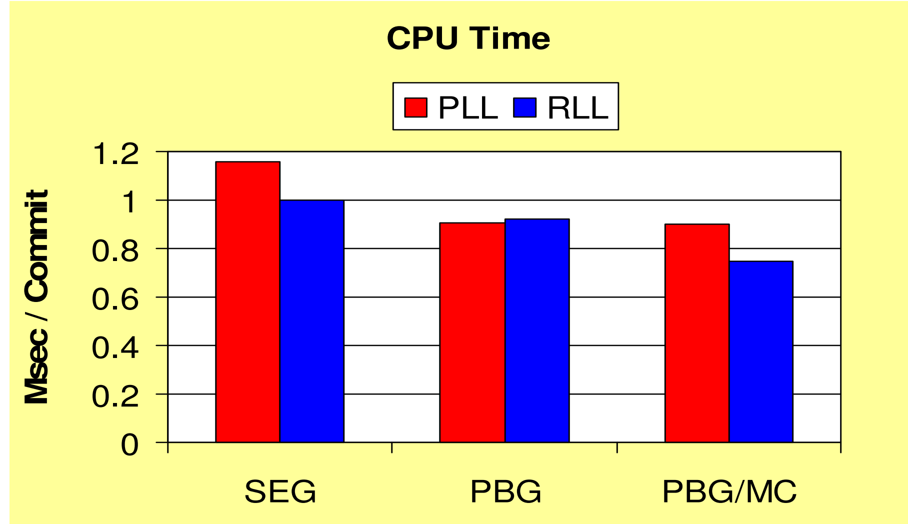
#SHAREorg

DB2 10 Non-range Defined Table Spaces

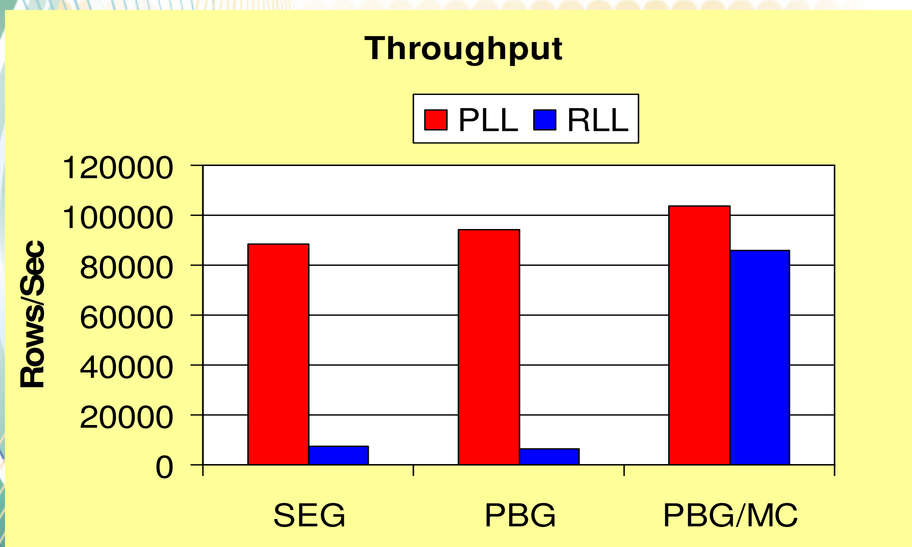
SHARE Technology · Connections · Results

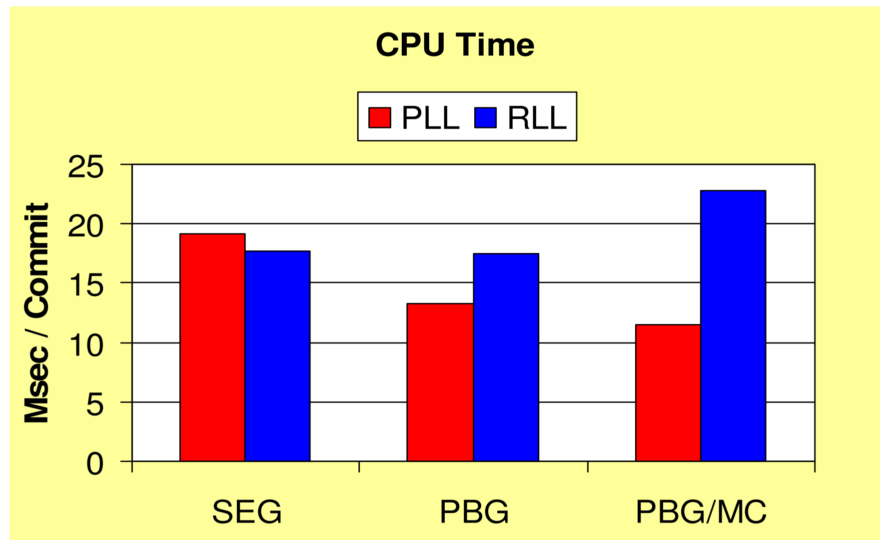
-----------------------------Random Inserts-----------------------------





-----------------------------Sequential Inserts-----------------------------





Conclusions

- New IBM storage hardware advancements are key to improving DB2 query performance
- DB2 10 improves query performance when index are disorganized and when doing sorted RID list scans
- Piecemeal list prefetch will further improve the performance of dynamic prefetch access paths. Also will save CPU and could improve the OLTP buffer hit ratio.
- All of this technology mitigates the performance cost of not reorganizing the data frequently, and SSD is a critical component needed to achieve that goal
- Be aware that a loss of clustering can affect your OLTP buffer hit ratios
- Be careful with indirect references

#SHAREorg

SHARE
Technology • Connections • Results

References

- **IBM Redpaper: DB2 for z/OS and List Prefetch Optimizer**
  http://www.redbooks.ibm.com/redpieces/abstracts/redp4862.html

SHARE in Anaheim 2012

Questions?

Email: bergerja@us.ibm.com