GUIDE SHARE EUROPE UK REGION

GSE UK Conference 2018

Better, stronger, faster; The Mainframe….. the Machine!

In-Memory Trends and Db2 for z/OS

Larry Strickland

DataKinetics

November 2018

Session LF

Introduction

Larry Strickland

Chief Products Officer

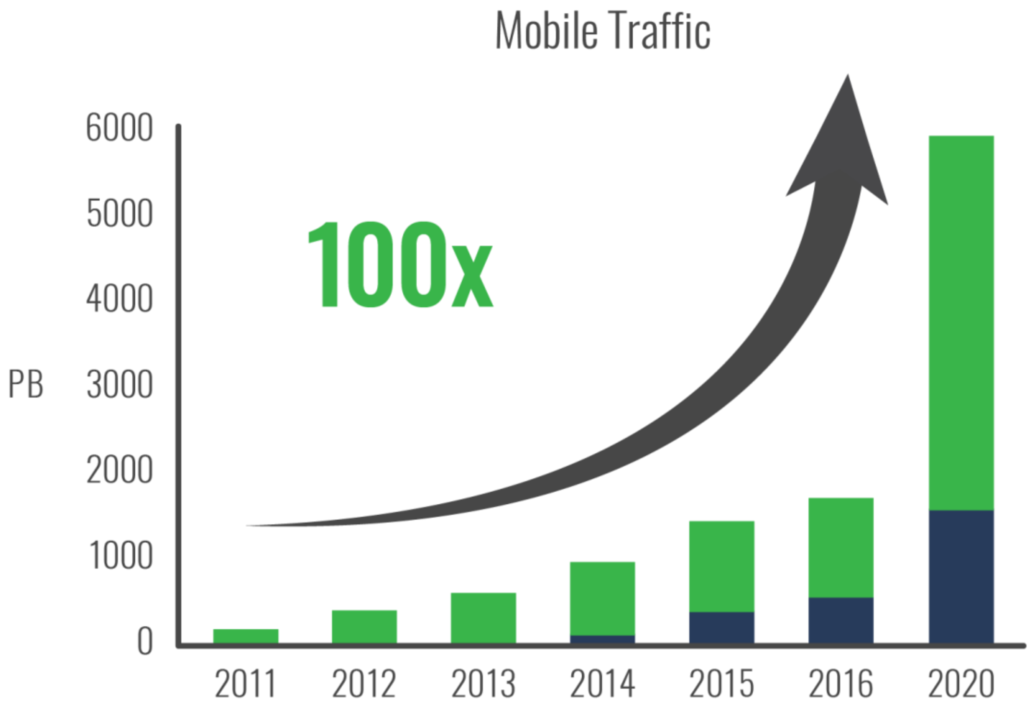
In-Memory Trends and Db2 for z/OS

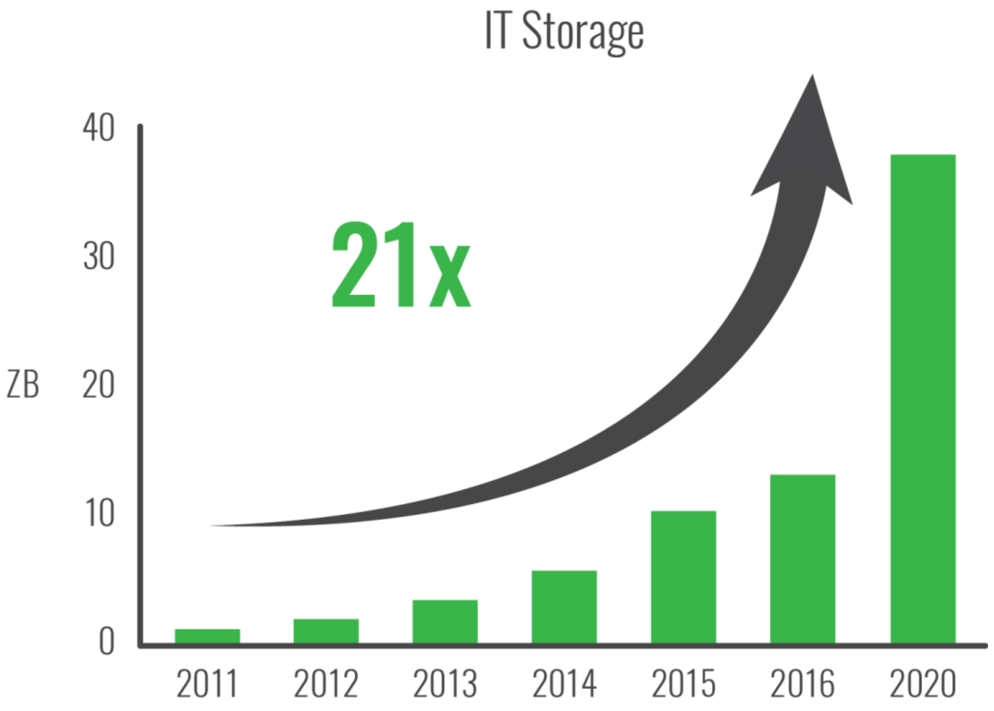
- Industry Trends
- In-Memory Benefits
- In-Memory Database Systems
- Db2 12 for z/OS In-Memory Capabilities
- In-Memory Tables

Industry Data Trends

- The amount of data we store and manage continues to expand at a rapid pace
- The pace accelerates as we access data
- Organizations are looking to process, analyze, and exploit this data accurately and quickly

And This Growth Will Continue





*1ZB = 1 Trillion Gigabytes

How Do We Manage It?

- Just live with increasing costs and decreasing customer satisfaction?

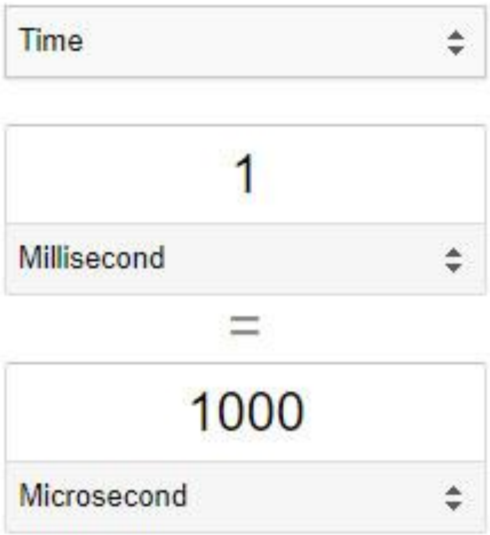
- Or do something about it – leveraging the fastest data storage we have – memory

Memory Costs are Decreasing

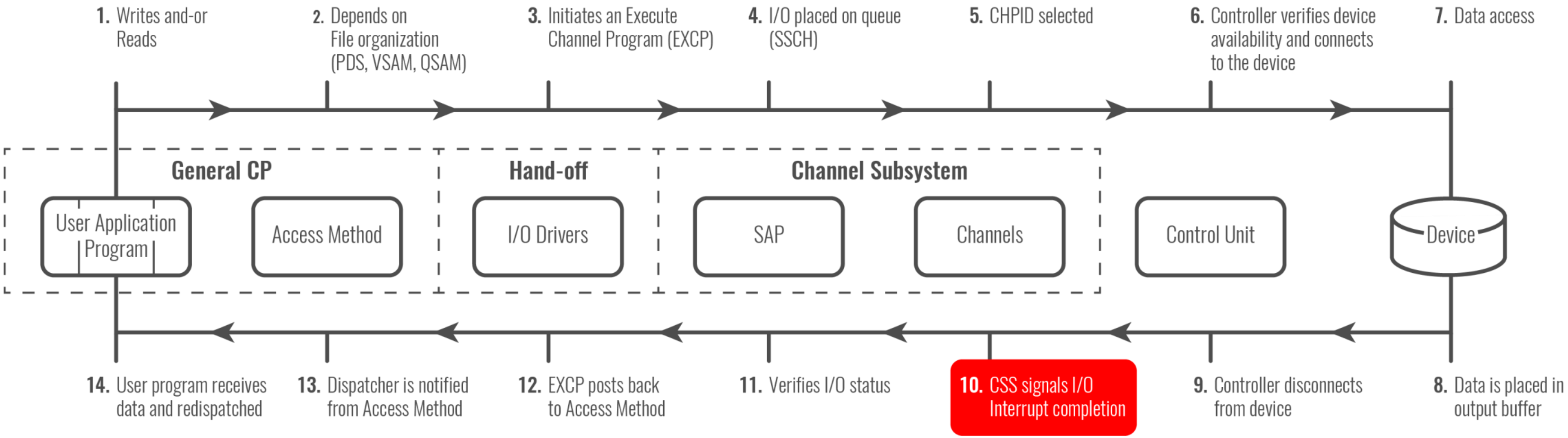
- Although the concept of in-memory processing has been around for a long time, the falling price of RAM and growing use cases have led to a new focus on in-memory techniques and processing
- Total cost of ownership can be lowered if you can reduce your hardware footprint using in-memory techniques
- Operating costs may also be cut by reducing maintenance needs
- Cloud options may allow you to move from fixed to variable expenses
- In-memory technology can bolster performance and possibly even change business processes

Disk Access is Much Slower Than Memory Access

- 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



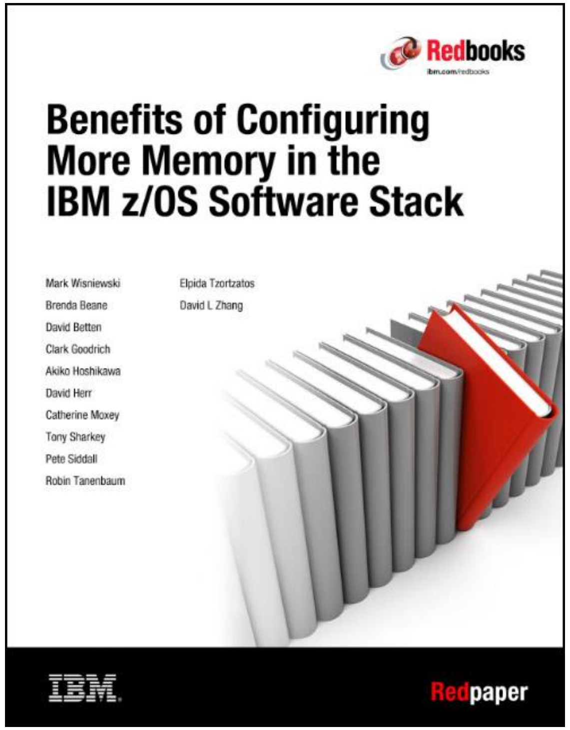
What is Involved in an I/O Operation?



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

Benefits of Memory

- CPU efficiency is improved with large memory when paging is avoided
- Batch workload processing time can be reduced
- For OLTP workloads, large memory provides substantial latency reduction, which leads to significant response time reductions and increased transaction rates



Source: *Benefits of Configuring More Memory in the IBM z/OS Software Stack,*
IBM RedPaper, REDP-5238-01, January 2017

In-Memory Use Cases

- In-memory techniques can optimize processes where large amounts of data, complex operations, and business challenges require real-time support
- Look for areas where instantaneous information can improve decision quality; in-memory processing can improve the speed of decision-making
- Analytics is likely to drive in-memory but its usefulness is not limited to analytical processing. Consider also transactions, long-running batch, and data warehousing
  - Requires modifying existing processes to take advantage of in-memory which can be time-consuming

IMDBMS:
In-Memory Database Management System

What is an In-Memory DBMS (IMDBMS)?

- An in-memory database (IMDB) is a database management system that **primarily depends on main memory** for storing data
- IMDBs are **quicker**
- IMDB eradicates disk access

Source: *Technopedia,*
https://www.techopedia.com/definition/28541/in-memory-database

IMDBMS: Benefits and Disadvantages

Benefits of In-Memory DBMS

- Performance
- Remove overhead related to translation and caching of data
- Use significantly less CPU
- This can deliver faster transaction processing

Disadvantages of In-Memory DBMS

- Although memory cost is dropping it is still more expensive than disk
- Lack of IT expertise
- Limitations on database size due to amount of memory available

Source: *How to determine if an in-memory DBMS is right for your company,* TechTarget 2017

Examples of In-Memory DBMS Offerings

- Aerospike
  - Flash-optimized open source NoSQL DBMS
- Altibase
  - Proprietary, general purpose IMDBMS with full ACID
- MemSQL Enterprise
  - Distributed in-memory SQL IMDBMS with full ACID
- Oracle TimesTen
  - In-memory relational database
- SAP HANA
  - In-memory, column-oriented RDBMS from SAP
- VoltDB
  - Michael Stonebraker's IMDBMS offering

Additional examples: *https://en.wikipedia.org/wiki/List_of_in-memory_databases*

What About Db2's In-Memory Capabilities?

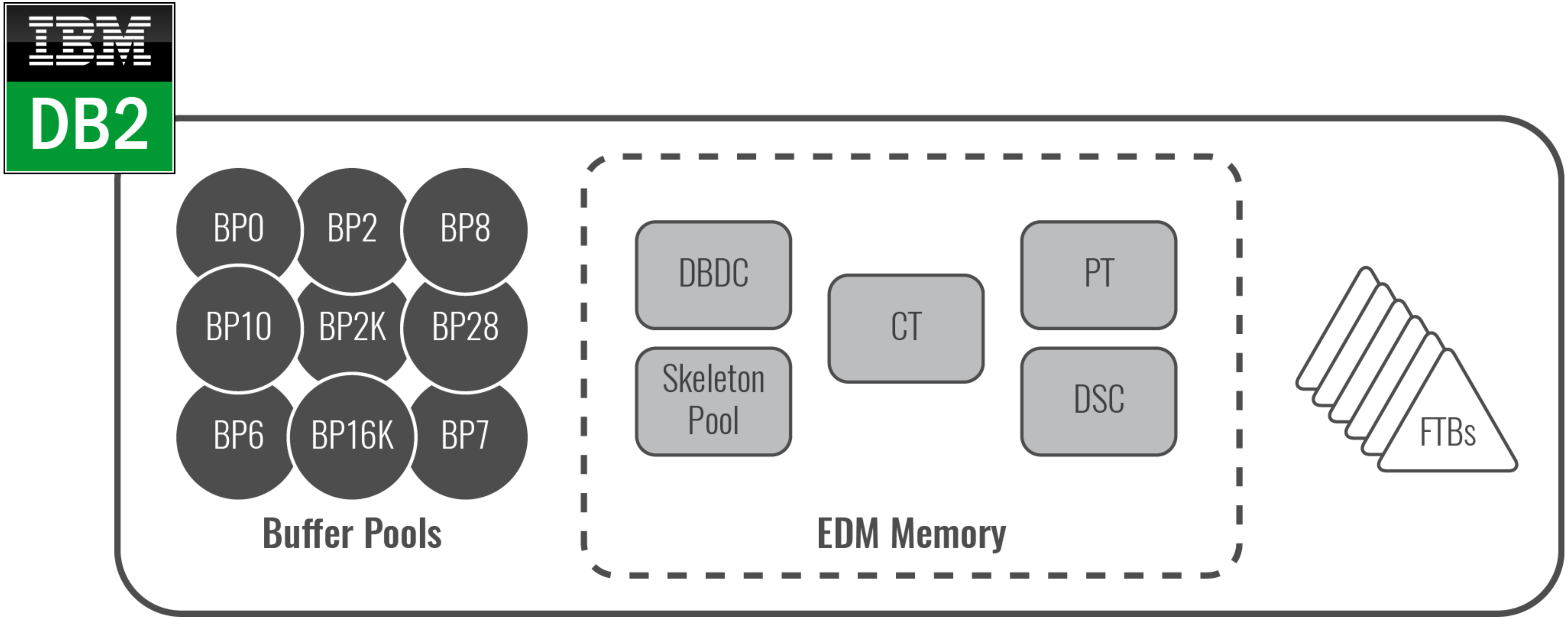
Db2 12 for z/OS

- There are many new features in Db2 12 for z/OS that exploit in-memory techniques
- So much so that analysts at Gartner have called Db2 for z/OS an “in-memory” DBMS
- We will examine these new features in detail:
  - Index Fast Traversal Blocks (FTBs)
  - New Fast Insert Algorithm
  - Contiguous Buffer Pools
  - In-Memory Sort Processing

Index FTBs

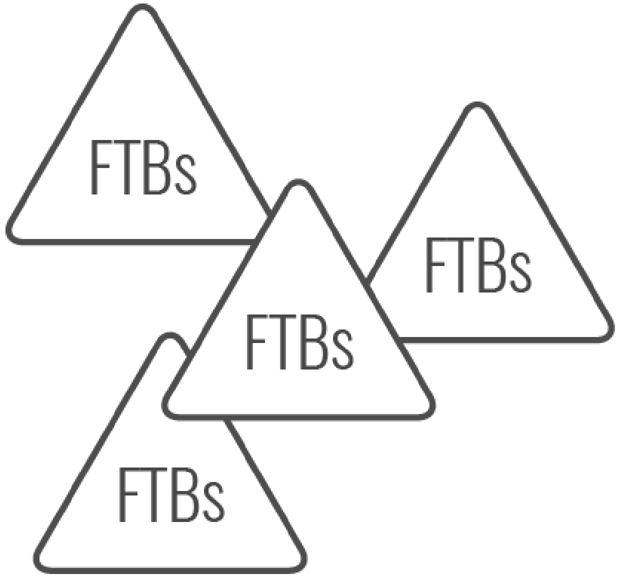
- Fast Traverse Blocks (FTBs)
  - Unique indexes can be stored in-memory
  - The key size must be 64 bytes or less
  - Stores only the high-level pages, not leaf pages
  - Unique indexes with INCLUDE columns are also supported in the FTB
  - Using FTBs for index traversal is much faster than doing traditional page-oriented page traversal for indexes that are cached in buffer pools

Where are FTBs Stored?



FTB Candidates

- Any index that is used predominantly for read access by way of key lookups
  - Also INSERT, UPDATE, and DELETE
- The best candidates for using FTB are:
  - Indexes that support heavy read access,
  - Indexes on tables with a random insert or delete pattern
  - Indexes with high PCTFREE



FTB Performance Measurements

1. Random index access by using single thread random select/insert/delete
    - PBG table space with 1 unique index, key size < 64 bytes, 5 levels
    - Class 2 CPU time decreases, between 8.5% and 22.4%
2. Sequential index access
    - PBG table space with 1 unique clustering index with a 56-byte key
    - CPU time change was insignificant (between +2% and -2%)
3. IBM Brokerage Workload
    - FTB set to AUTO
    - 12 fewer GETPAGEs per COMMIT

Source: *IBM Db2 12 for z/OS Performance Topics (SG24-8404)*

FTB Summary

- FTBs enable more index data to be stored in memory
- FTBs can improve performance of queries that rely on unique indexes
- The greater the number of levels in the index, the greater the expected CPU savings will be
  - Initial measurements as published by IBM in the Db2 12 for z/OS Technical Overview indicate CPU savings varies from about 8% for a two-level index to 23% for a five-level index

New, Fast Insert Algorithm

Fast Insert Algorithm (aka Insert Algorithm 2)

- New INSERT algorithm for journaling workload
  - Data is unclustered, just added to the end of the space
  - Not for standard, try-to-keep-things-clustered workloads
- Requires MEMBER CLUSTER and UTS
  - Only available for Universal table spaces
  - MEMBER CLUSTER minimizes data sharing overhead for an INSERT-heavy Db2 table (space map management)

New, Fast Insert Algorithm

- How it works
  - An in-memory structure called an Insert Pipe is used to control INSERTs across data sharing members
  - Insert Algorithm 2 uses an asynchronous background system task
  - The Insert Pipe is filled asynchronously

New, Fast Insert Algorithm: Insert Algorithm 2

- Performance
  - Can improve INSERT throughput, especially when data is not indexed
  - Can also lower logging activities and reduce class 2 elapsed time and class 2 CPU time
- Benchmarking tests by IBM showed a high potential for performance improvement for the right use cases
  - Workloads that are constrained by lock/latch contentions on the space map pages and data pages are likely to benefit more from it

Use Cases for Fast Insert

- High rate of concurrent INSERTs into a journal or audit table
  - When rows cannot be inserted quickly enough using the standard INSERT algorithm, performance suffers:
  - Insert Algorithm 2 use cases include tracking data for regulatory compliance, writing out access details, etc.

Insert Algorithm 2: Performance Measurement

Test1: Insert with no indexes defined

- Two-way data sharing environment with group buffer pool-dependent objects
- Two tests were run:
  - The insert rate showed an 18% improvement
  - The class 2 elapsed time per transaction reduced by 54%
  - The Db2 class 2 CPU time per transaction decreased by approximately 15%

Source: *IBM Db2 12 for z/OS Performance Topics (SG24-8404)*

Insert Algorithm 2 Performance Measurement (cont.)

Test 2: Insert with indexes defined

- Same setup but test were run with one, two, and three indexes defined
- The tables were clustered, so all the rows are inserted in the order of their sequential keys.
- The insert rate improved by 26%

Test 3: Random Insert and Delete

- Same setup but with random inserts (not journaling) and deletes
- No significant difference

Source: *IBM Db2 12 for z/OS Performance Topics (SG24-8404)*

Contiguous Buffer Pools

- Contiguous Buffer Pools
  - In-memory pools are treated as a single block of storage
  - They do not require chain maintenance
  - Ideal for code tables and frequently used smaller tables
    - Stable data
  - Set up using the PGSTEAL parameter of the buffer pool

Using Contiguous Buffer Pools

- Which tables are good candidates for contiguous Buffer Pools?
  - The table or index should be able to fit entirely within the buffer pool
  - Objects should be referenced frequently with a high number of GETPAGEs
  - Identifying high GETPAGEs:
    - New RTS column GETPAGES in RTS tables
  - GETPAGE intensity is important, too

Contiguous Buffer Pools Performance Measurements

- OLTP workload
  - First test done with buffer pools having a high GETPAGE count configured to use Contiguous Buffer Pools
  - Second test done with the buffer pool setting changed to default settings, but using the same VPSIZE
  - Class 2 elapsed time reduced 7%
  - Db2 class 2 CPU time decreased by 8%

In-Memory Sort Processing

- In-memory sort processing
  - Increased max number of nodes available for sort tree
  - By increasing in-memory sorting you can avoid writing intermediate SORTWORK files to disk
  - Limited number of nodes could also effectively cap the sort pool size
- These enhancements can require more memory, but can result in a reduced CPU

Sort Performance Measurements

- 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

Source: *IBM Db2 12 for z/OS Performance Topics (SG24-8404)*

Sort Performance Measurements (cont.)

- 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

Source: *IBM Db2 12 for z/OS Performance Topics (SG24-8404)*

Db2 12 In-Memory Synopsis

- Db2 12 provides significant new in-memory capabilities:
  - Index Fast Traversal Blocks (FTBs)
  - New Fast Insert Algorithm
  - Contiguous Buffer Pools
  - In-Memory Sort Processing
- After hearing about these features, Gartner analysts would refer to Db2 for z/OS as an IMDBMS

Other In-Memory Techniques

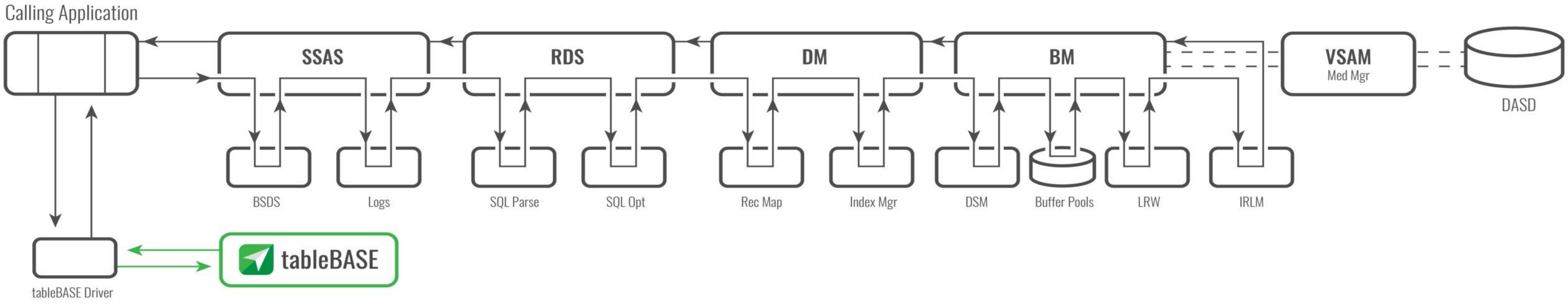
- There are numerous techniques you can use to expand your usage of memory
  - Working storage memory
  - Use of z/OS storage (ECSA)
  - Dataspaces
  - Above the bar storage
  - Use a vendor product that handles it for you

High-Performance In-Memory Technology

- What is high-performance In-memory technology?
  - An in-memory accelerator for mainframe applications
  - Dramatic improvements for existing applications
  - Doesn’t replace your existing database – it complements it
  - Example: *DataKinetics tableBASE*

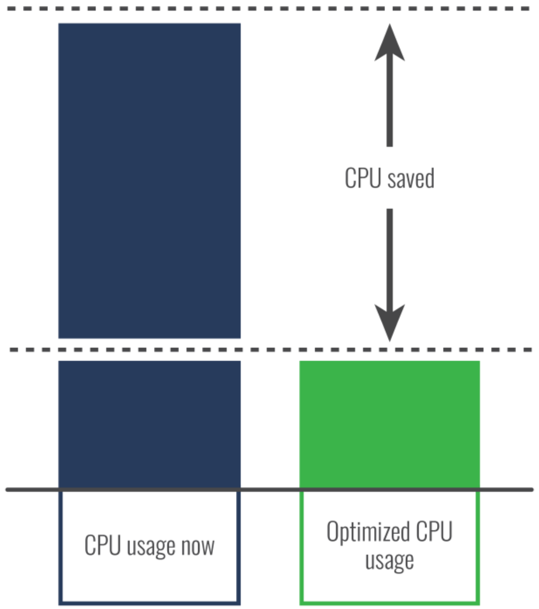
tableBASE In-Memory Technology

- How does it work?
  - Uses a much shorter code path to access data
  - Top path is a typical DBMS code path
    (typ. 10,000 to 100,000 machine cycles)
  - Bottom path is the high-performance in-memory code path, 20x faster
    (typ. 400-500 machine cycles)

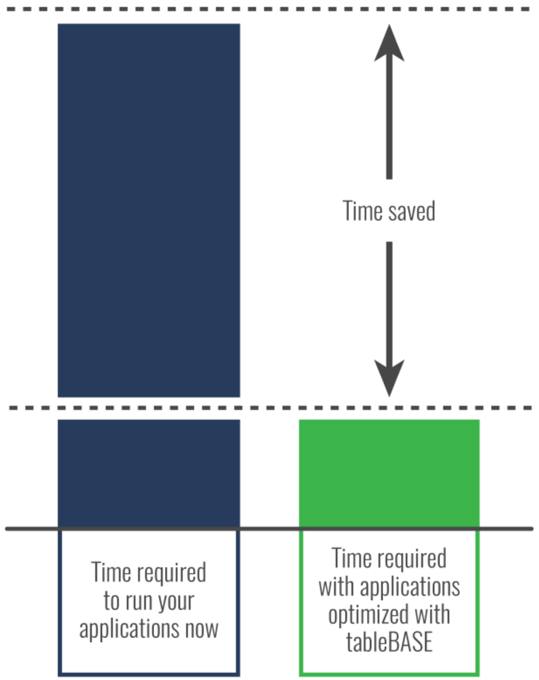


High-performance In-memory Results

ONLINE APPLICATION CPU USAGE



BATCH PROCESSING TIME



Elapsed Time and CPU Reduction Example

GUIDE SHARE EUROPE UK REGION

SDSF output showing job status – original (BEFORE) and afterwards, using tableBASE (AFTER):

BEFORE
CPU 4.74
Elapsed Time 49.6

AFTER
CPU 0.21
Elapsed Time 0.70

BEFORE

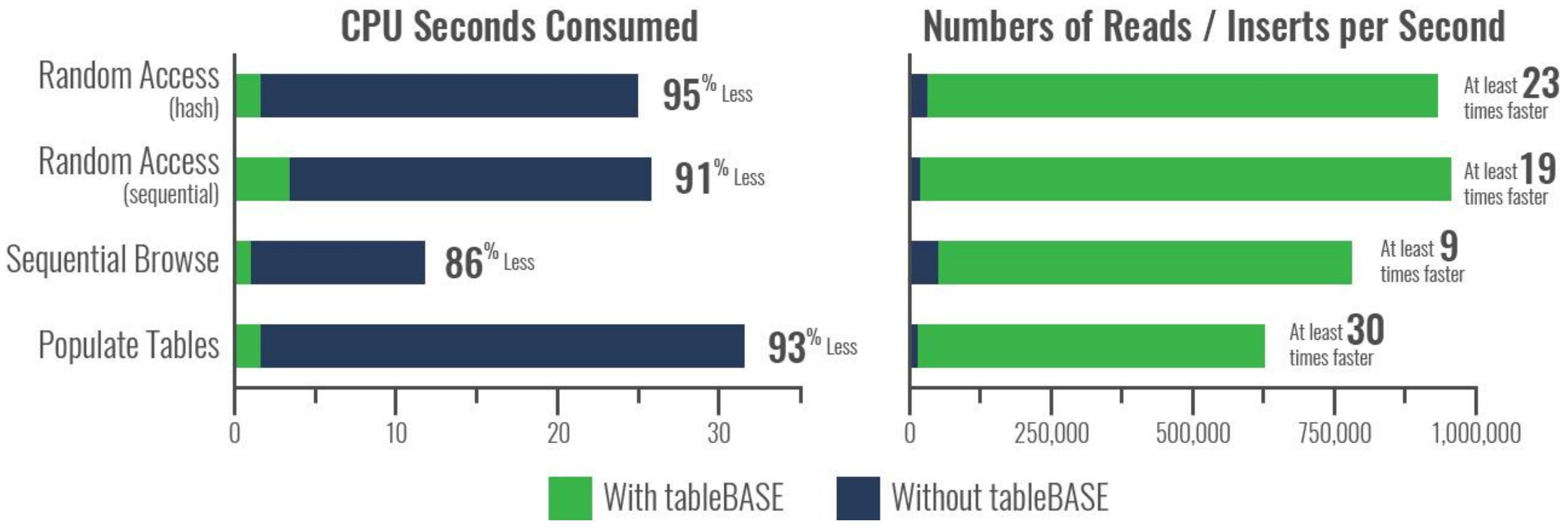
```
11.59.49  J0027773  ----  THURSDAY,    20 APR 2017  ----
11.59.49  J0027773  ICH70001I  UCTMMBD    LAST ACCESS AT 11:55:31 ON THURSDAY, APRIL 20, 2017
11.59.49  J0027773  $HASP373  OGJEDOM2  STARTED - WLM INIT - SRVCLASS BATCH_A - SYS MEXD
11.59.49  J0027773  IEF403I  OGJEDOM2 - STARTED - TIME=11.59.49
11.59.50  J0027773  -                                          --TIMING (MINS.)--          -----PAGING COUNTS-----
11.59.50  J0027773  -STEPNAME  PROCSTEP    RC  EXCP  CONN  TCB  SRB  CLOCK   SERV  WORKLOAD  PAGE  SWAP  VIO  SWAPS
11.59.50  J0027773  -NONCAT2  CONTROLR     00   649   196  .00  .00     .0   7595  BATCH        0     0    0      0
11.59.50  J0027773  -OGJDOM2  OGCDOM01     00  417K 40456 4.74  .04   49.5 35054K  BATCH        0     0    0      0
12.49.25  J0027773  IEF4041  OGJEDOM2 - ENDED - TIME=12.49.25
12.49.25  J0027773  -OGJEDOM2 ENDED.  NAME-JCL SORT FILE     TOTAL TCB CPU TIME=    4.74    TOTAL ELAPSED TIME=    49.6
12.49.25  J0027773 $HASP395  OGJEDOM2 ENDED
```

AFTER

```
11.43.25  J0026723  -OGJEDOM2  OGCDOM01    00  123K  9430  .21  .00     .7 18057K  BATCH        0     0    0      0
11.59.49  J0026723  IEF4041  OGJEDOM2 - ENDED - TIME=11.43.25
11.59.49  J0026723  -OGJEDOM2 ENDED.  NAME-JCL SORT FOLE     TOTAL TCB CPU TIME=     .21    TOTAL ELSAPSED TIME=     .7
11.59.49  J0026723  $HASP395  OGJEDOM2 ENDED
```

How Fast? 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



Summary

- Data growth continues unabated – using memory can make a difference
- Cost of memory is decreasing it more cost-effective
- IMDBMS are gaining popularity – both new and old
- Db2 for z/OS is gaining significant in-memory capabilities
- There are other 3rd-party solutions that can be added to further take advantage of in-memory performance gains

Q&A

Q&A

We want your feedback!

- Please submit your feedback online at ….
  - ➢http://conferences.gse.org.uk/2018/feedback/lf
- Paper feedback forms are also available from the Chair person
- This session is LF