

Total Recall: Exploiting In-Memory Features in DB2 12 for z/OS



Julian Stuhler
DB2Night Show
October 2018

Acknowledgements

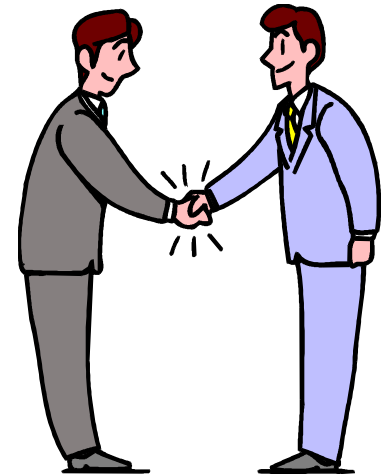
- Nina Bronnikova (IBM)
- Peter Hartmann (IBM)
- Jeff Josten (IBM)
- Akiko Hoshikawa (IBM)
- Cristian Molaro (Molaro Consulting)

Agenda

- Introduction
- Background: The Big Memory Era
- In-Memory Features in DB2 12 for z/OS
- Practical Guidelines
- Conclusions

Introduction

- DB2 consultant with Triton Consulting, based in the UK
- 30 years DB2 experience
 - Database Administration
 - Systems Programming
 - Application Development
- IBM Gold Consultant
- IBM Champion
- IBM White Papers, Redbooks, Flashbooks, etc
- IDUG Best Speaker and Past President



Agenda

- Introduction
- Background: The Big Memory Era
- In-Memory Features in DB2 12 for z/OS
- Practical Guidelines
- Conclusions

Background: The Big Memory Era

Machine Type	Max Memory per CEC	Max Memory per LPAR (z/OS)	Approx Cost per GB ¹	Reduction v Previous Generation
z10	1.5TB	1TB	\$,6,000	-
z196	3TB	1TB	\$1,500	75%
zEC12	3TB	1TB	\$1,500	-
z13	10TB	4TB²	\$300⁴	80%
z14	32TB	4TB^{2,3}	-	-

DB2 Release	Max Total BP Size ⁵
V9	1TB
V10	1TB
V11	1TB
V12	16TB

1. Approximate US “street price” at time of each release
2. Requires z/OS V2.2, or z/OS V2.1 with some PTFs
3. z14 LPAR architectural limit is 16TB, some OSES already able to exploit this fully (e.g. z/VSE)

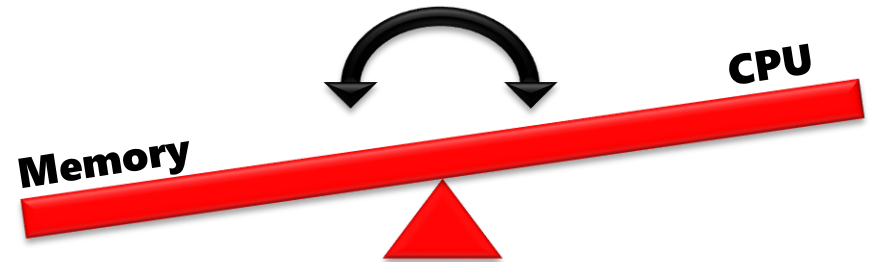
4. Assuming customer triples previously installed memory at upgrade
5. Subject to max. of 2x LPAR storage, not recommended!

Big Memory Exploitation

- From DB2 10 onwards virtual storage is typically no longer a major constraint
- Supersizing real storage allocation can be a very cost effective way of improving performance
 - One time purchase cost, unlike CPU upgrade you do not pay additional IBM or ISV MLC fees
 - DB2 is able to use as much memory as you're willing / able to provide!
- However:
 - Limited resource, was expensive to buy
 - Sometimes difficult to convince z/OS sysprogs to give up!
- z13 fundamentally changed the game with significantly increased real storage capacity at a fraction of the previous price, z14 continues to push the limits
- DB2 for z/OS is aggressively exploiting this trend

Big Memory Exploitation: The “Campbell Feast”

- Trade increased real storage usage for decreased CPU / elapsed time (and stability)
 - DB2
 - ▶ Local and group bufferpools
 - ▶ Dynamic Statement Cache
 - ▶ Log output buffers
 - ▶ Etc.
 - ▶ DB2 12 offers new possibilities!
 - Other
 - ▶ DFSORT
 - ▶ z/OS dump space

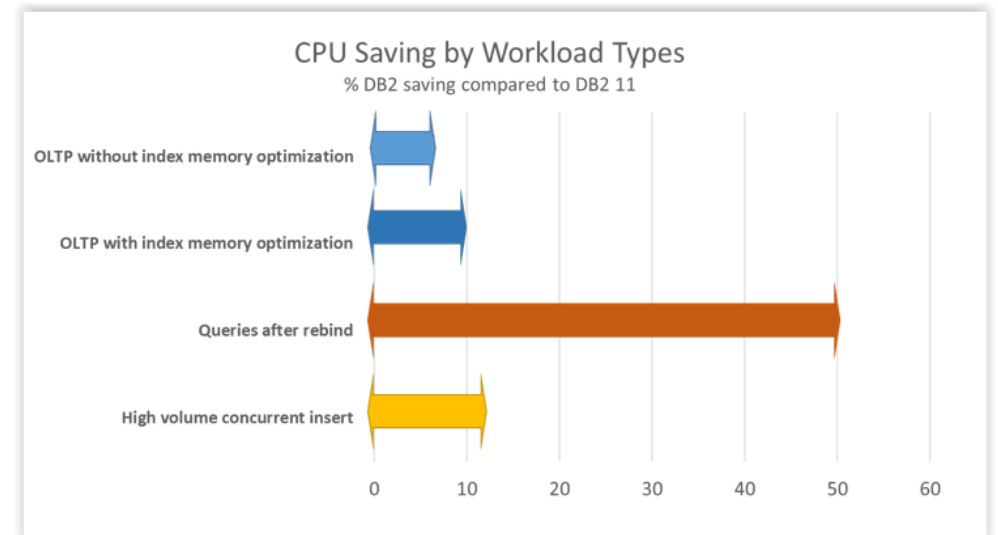


Agenda

- Introduction
- Background: The Big Memory Era
- In-Memory Features in DB2 12 for z/OS
- Practical Guidelines
- Conclusions

DB2 12 for z/OS

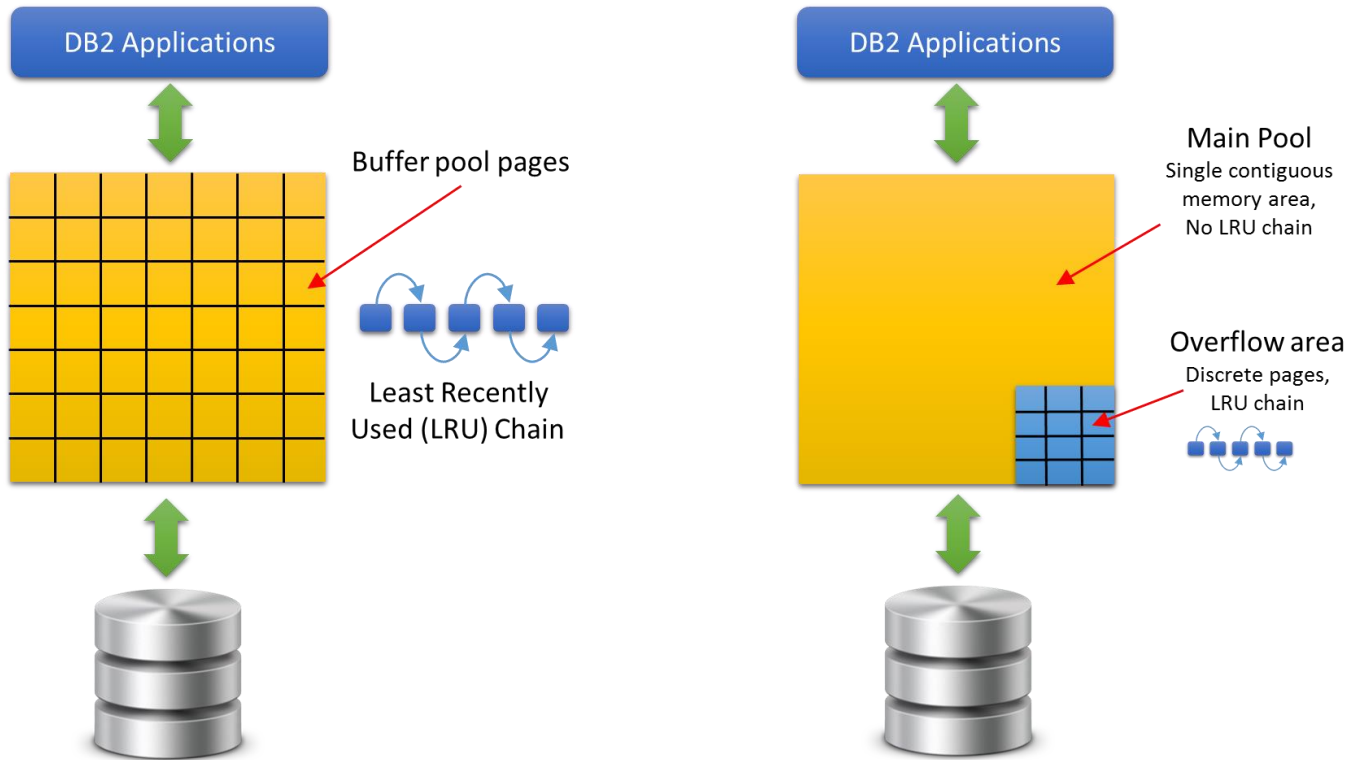
- DB2 12 Early Support Program announced in October 2015, and started in May 2016
- General Availability for new release on 21st October 2016
- IBM has steadily expanded the in-memory capabilities of DB2 in recent releases
 - DB2 12 for z/OS is now officially recognised by Gartner as a bona fide “In Memory Database”
- Wealth of important new features, but many of the performance benefits relate to in-memory capabilities and increased exploitation of real storage
 - IBM quoting up to 6-7% CPU reduction for OLTP without Fast Traverse Blocks, increases to up to 8-9% with FTB enabled
 - Further savings possible with contiguous bufferpools and other DB2 12 features



Contiguous Buffer Pools – Overview

- PGSTEAL(NONE) improved in DB2 12 to avoid LRU and hash chain management overheads
- Pages stored contiguously in memory in same order as on disk
- Up to 8% CPU reduction measured for OLTP in IBM testing
 - Direct access to data, so reduced GETPAGE and RELEASE PAGE overhead
 - Hash and LRU chains not maintained
- Separate overflow area for pages that won't fit in pool
 - Included in VPSIZE (need to add allowance for overflow to total size of objects in pool)
 - Page stealing can still occur for overflow pages (LRU basis)
 - Overflow area not backed by real storage until needed (hopefully never)

Contiguous Buffer Pools – Overview



Contiguous Buffer Pools – Monitoring

- -DISPLAY BUFFERPOOL shows size of overflow area in DSNB402I

```
DSNB402I - BUFFER POOL SIZE = 10000 BUFFERS  AUTOSIZE = NO  VPSIZE MINIMUM = 0  VPSIZE MAXIMUM = 0  
          ALLOCATED = 10000  TO BE DELETED = 0  IN-USE/UPDATED = 8746  OVERFLOW ALLOC = 1000
```

- -DISPLAY BUFFERPOOL (DETAIL) shows additional overflow information in DSNB416I

```
DSNB416I - OVERFLOW RANDOM GETPAGE      = 0  
          OVERFLOW SYNC READ I/O (R) = 0  
          OVERFLOW SEQ.  GETPAGE        = 0  
          OVERFLOW SYNC READ I/O (S) = 0
```

- IFCID002 (statistics BP activity) extended to include same counters as shown in DSNB416I
- DSNB604I message is issued if pages are read into overflow area

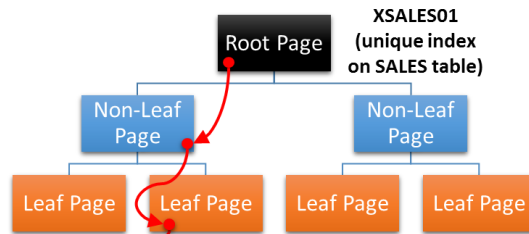
```
DSNB604I  PAGES WERE READ INTO BP4 BUFFER POOL OVERFLOW AREA FOR DATASET = DB2P.DSNDBC.  
          DPAY0001.SSAL0034.I0001.A001 OVERFLOW AREA SIZE = 6400
```

Fast Traverse Blocks – Overview

- Memory-optimized structure for fast random index lookups
- DB2 autonomically determines which indexes would benefit from FTB (see later)
- UNIQUE indexes only (INCLUDE supported), key size 64 bytes or less
- Resides in special memory area outside of the buffer pools
 - Separate FTB allocation for each member in a data sharing group
 - Plan for increased real storage for this as part of your V12 migration
- Enabled from V12R1M100 onwards
 - V12R1M500 needed for GBP-dependent indexes to be eligible, due to new FTB p-lock
- New INDEX_MEMORY_CONTROL ZPARM controls size of FTB memory area
 - AUTO – DB2 sets upper limit to largest of: 10MB or 20% of **total** allocated BP storage (max 200 GB)
 - DISABLE – Disable FTB feature (pre-DB2 12 behaviour)
 - nnnn – fixed upper limit, 10 – 200,000 MB
- Benefits increase with number of index levels
 - Up to 23% CPU reduction in IBM testing (5-level index)

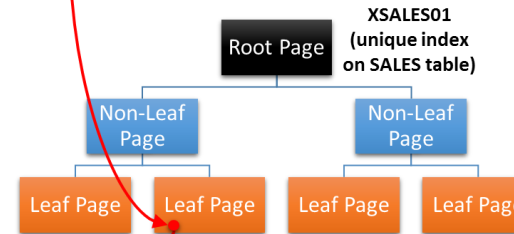
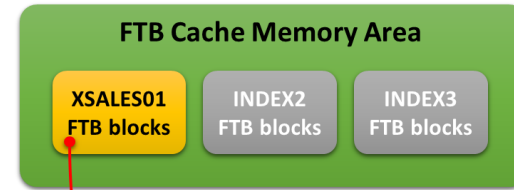
Fast Traverse Blocks – Overview

```
SELECT SALES_DATE, SALE_VALUE ...
FROM SALES
WHERE SALES_NO = 84628
```



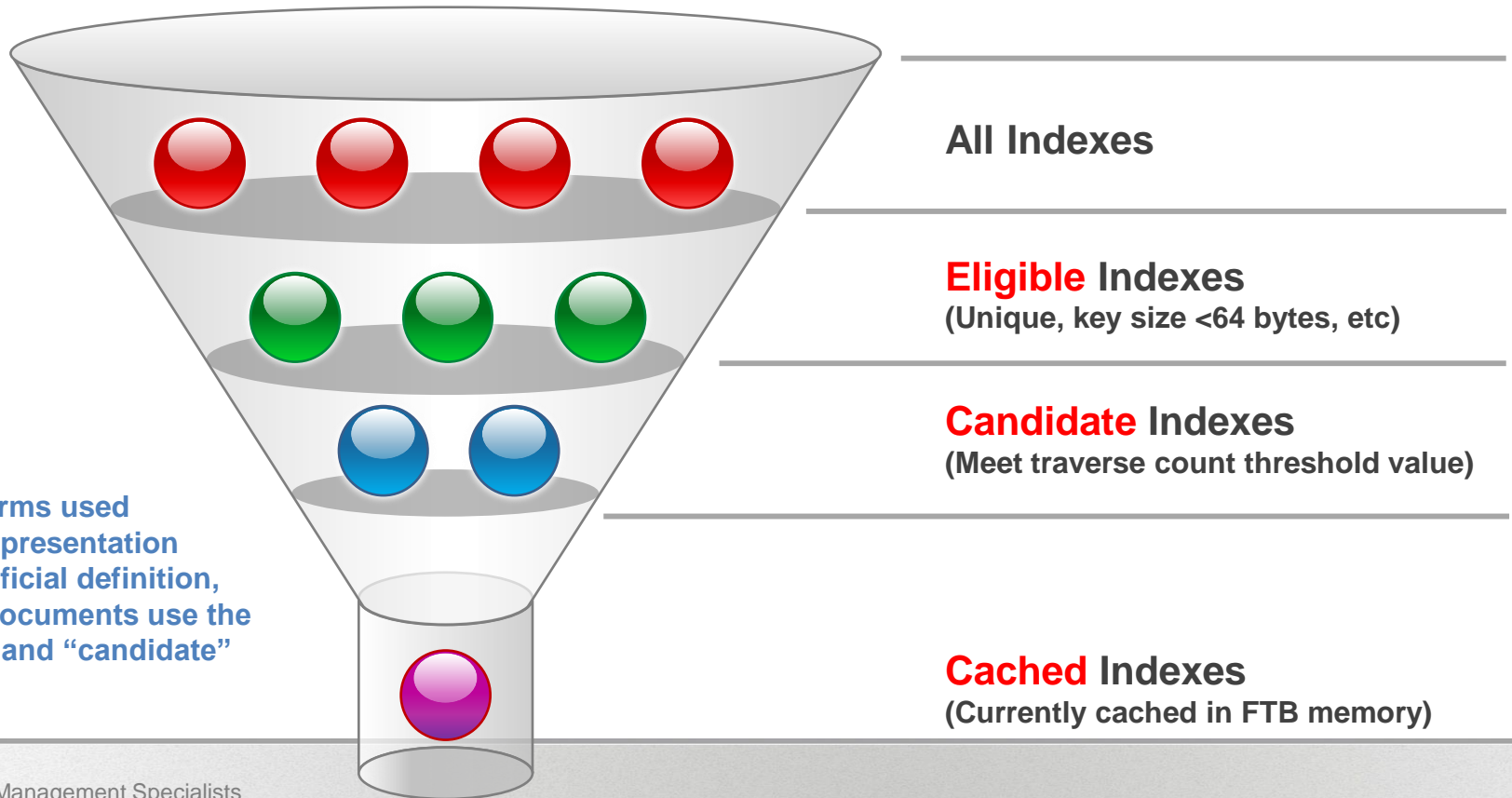
SALES_NO	SALES_DATE	SALE_VALUE	...
...
84627	04/07/2013	43.75	...
84628	10/07/2013	4453.00	...
84629	10/07/2013	43.54	...
84630	11/07/2013	765.12	...
...

```
SELECT SALES_DATE, SALE_VALUE ...
FROM SALES
WHERE SALES_NO = 84628
```



SALES_NO	SALES_DATE	SALE_VALUE	...
...
84627	04/07/2013	43.75	...
84628	10/07/2013	4453.00	...
84629	10/07/2013	43.54	...
84630	11/07/2013	765.12	...
...

Fast Traverse Blocks – Terminology



Please note:

- These are the terms used throughout this presentation
- This is not an official definition, many of IBM's documents use the terms "eligible" and "candidate" interchangeably

Fast Traverse Blocks – Eligibility

- To be eligible, indexes must:
 - Be open
 - Be UNIQUE (INCLUDE supported)
 - Have a key size 64 bytes or less
 - Not have multiple versions
 - ▶ `SYSINDEXES / SYSINDEXPART: OLDEST_VERSION = CURRENT_VERSION`
 - Not be defined on a catalog table
 - Not have a `TIMESTAMP WITH TIME ZONE` column in the key

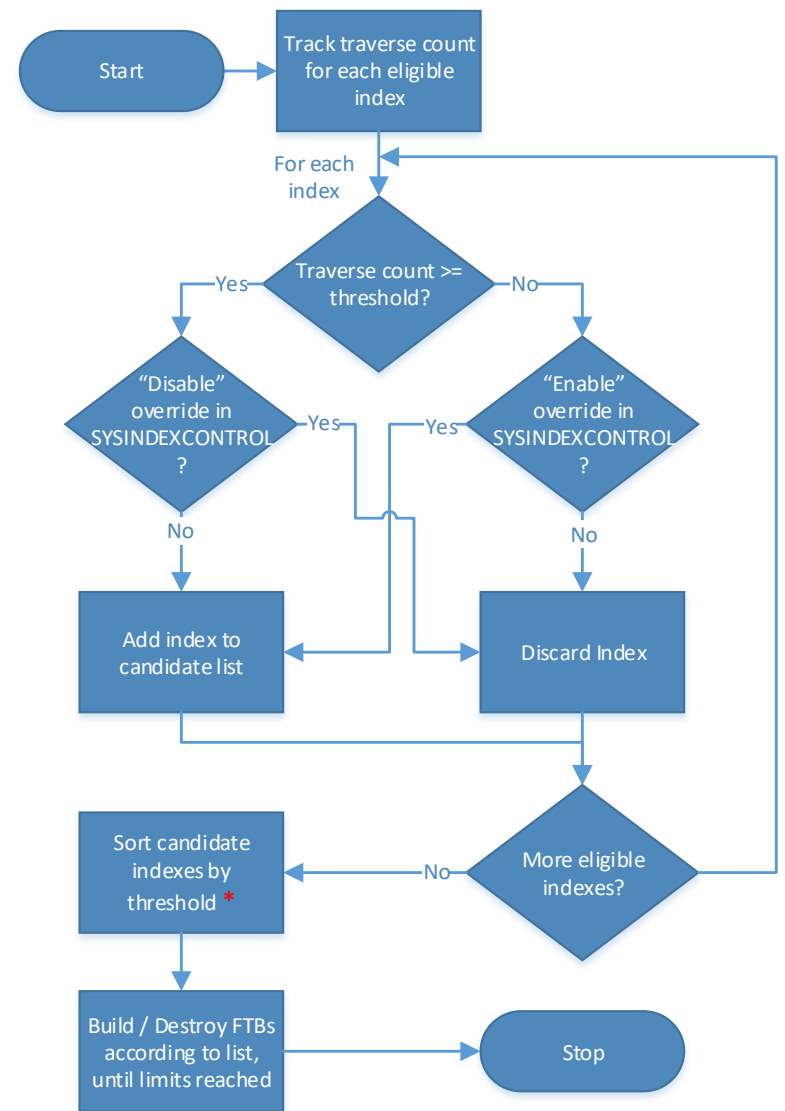
Fast Traverse Blocks – Manually Influencing Candidates

- **SYSIBM.SYSINDEXCONTROL**
 - Specifies time windows to influence the use of FTB for an index
 - Each row specifies time window for enabling or disabling the use of FTB for a specific index object
 - Up to 10 minutes before changes to table take effect
- **Note:** modified threshold requirements must be met, even for actions “A” and “F”!
- **Caution:** duplicates allowed, only first row read for a given index is used!

Column name	Description
SSID	The name of a DB2® subsystem.
PARTITION	Partition number
IXNAME	The name of the index.
IXCREATOR	The schema of the index.
TYPE	The purpose for which memory is being used: <ul style="list-style-type: none"> • F – A structure for FTB
ACTION	The action that is being performed: <ul style="list-style-type: none"> • F – Force FTB creation • D – Disable FTB creation • A – Automatic FTB creation
MONTH_WEEK	The meaning of the value in the DAY column. <ul style="list-style-type: none"> • M – A day of the month • W – A day of the week
MONTH	Month during which the time window applies. Valid values are 1 (January) to 12 (December)
DAY	Day of the month or day of the week for which the time window applies. <ul style="list-style-type: none"> • If MONTH_WEEK='M', valid values are 1 (first day of the month) to the number of the last day of the month, or NULL. • If MONTH_WEEK='W', valid values are 1 (Monday) to 7 (Sunday), or NULL.
FROM_TIME	The time of day at which the time window begins.
TO_TIME	The time of day at which the time window ends.

Fast Traverse Blocks – Selecting Candidates

- FTB daemon regularly monitors index usage to determine FTB candidates (every 2 mins)
 - Runs in DBM1 address space, 100% zIIP eligible
 - Indexes can drop in/out of eligibility according to workload
 - Daemon attempts to avoid thrashing while minimising time that low-benefit FTBs continue to exist
 - Object level control / override possible through catalog table SYSIBM.SYSINDEXCONTROL
- FTB “traverse count” is main selecting factor
 - Moving average tracked for each eligible index/partition within a given DB2 member
 - Incremented by random access traverse
 - Decremented by sequential access or non-leaf page split
 - Triggering threshold
 - ▶ Reported in IFCID002 QISTTRAVMIN
 - ▶ Set by hidden ZPARM, might be made adaptive in the future



* Adjustments made for SYSINDEXCONTROL entries, indexes with existing FTB's positively weighted

Fast Traverse Blocks – Monitoring (System Level)



- DSNIO70I provides information on overall FTB usage
 - Changed in PI72330 to reduce number of metrics reported and frequency (now only issued when number of cached indexes changes)

 **No. of index objects
cached in FTB pool**
Cached

```
00- 19.37.11 STC00072 DSNIO70I -DB2P FAST INDEX TRAVERSAL STATUS: 20 MB, - USED BY: 16 OBJECTS,  
CANDIDATE OBJECTS: 16
```

 **No. of candidate
index objects**
Candidate
(that meet threshold requirements)

**Memory allocated
to FTB pool**

Fast Traverse Blocks – Monitoring (System Level)

- Data Management section of IFCID002 (statistics class 1) expanded by PI72330 to include system-level FTB diagnostic info

IFCID002 Field	Description
QISTTRAVMIN	FTB threshold: minimum number of index traversals
QISTFTBCANT	Total number of indexes which meet FTB criteria
QISTFTBCAN	Total number of indexes which meet FTB criteria and the traverse count is above the threshold
QISTFTBSIZE	Total memory allocation for all FTBs for this member (in MB)
QISTFTBNUMP	Number of indexes for which FTB existed in the previous run of in-memory optimization
QISTFTBNUMC	Number of indexes for which FTB exists in the current run of in-memory optimization



Eligible



Candidate



Cached


Fast Traverse Blocks – Monitoring (Object Level)

- -DISPLAY STATS (INDEXMEMORYUSAGE) / (IMU)

```

DSNT783I -
DBID  PSID  DBNAME    CREATOR      INDEXNAME     LEVEL  PART   SIZE(KB)
-----  -----  -
0278  0005  DB1       SYSADM       I1            0005   00001  00018339
0017  0005  DB3       SYSADM       IDX3          0003   00006  00000061
      -THRU                                00010
***** DISPLAY OF STATS COMPLETE *****
DSN9022I - DSNTDSTS 'DISPLAY STATS' NORMAL COMPLETION
  
```

**FTB memory
allocated to
each index / part
(largest -> smallest)**



Fast Traverse Blocks – Monitoring (Object Level)

- New IFCID389 (statistics class 8) lists all cached indexes
 - DBID, PSID, Partition number, number of index levels, amount of storage allocated to FTB (equivalent to DISPLAY STATS IMU)
 - Contains data for all current FTBs in the system
 - Written every 2 minutes
- New IFCID 477 (performance) written on create/free of FTBs for a given index
 - DBID, PSID, Partition number, number of index levels, amount of storage allocated to FTB, IFCID Flags and which action occurred
- PI73762 corrects issue with these records being truncated

Other Memory-Related V12 Items

- Async lock structure duplexing
- Buffer Pool Simulation
- Prefetch enhancements
- Part of data set control block moves above the bar
- Increased use of 1MB pageable large page frames for DB2 working storage
- LOB Compression
- Sort Enhancements

Agenda

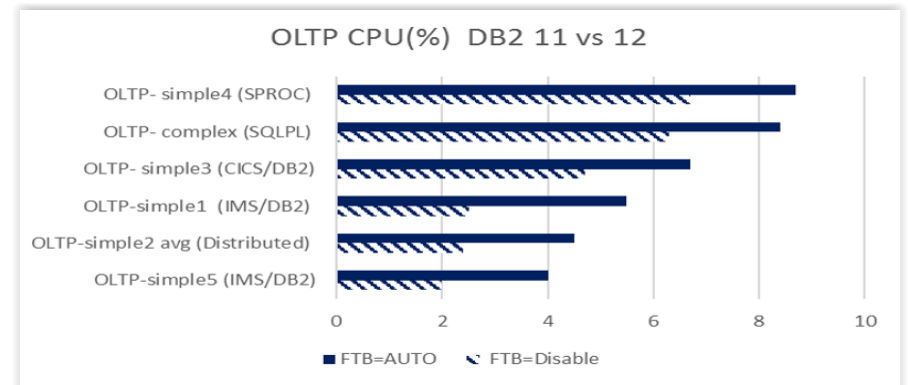
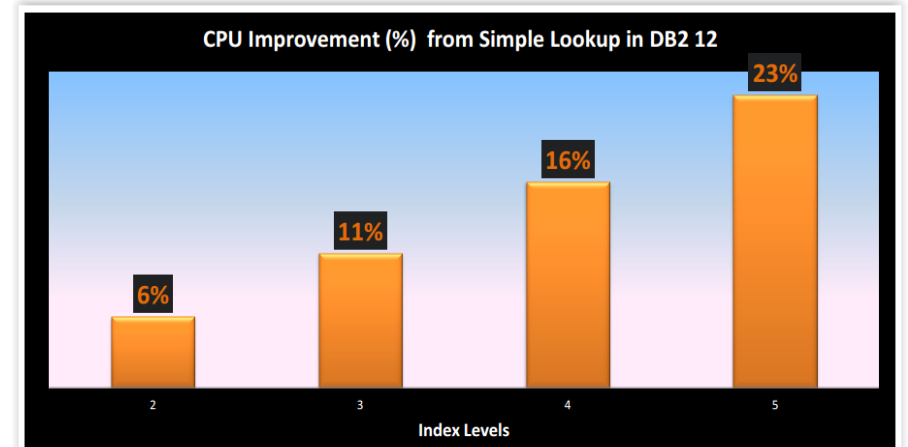
- Introduction
- Background: The Big Memory Era
- In-Memory Features in DB2 12 for z/OS
- Practical Guidelines
- Conclusions

Practical Guidelines

- Performance Expectations
- Contiguous BPs
 - Identifying Suitable Candidates
 - Allowing for the Overflow Area
- FTBs
 - Tuning Thoughts
 - Why aren't FTB's being used?

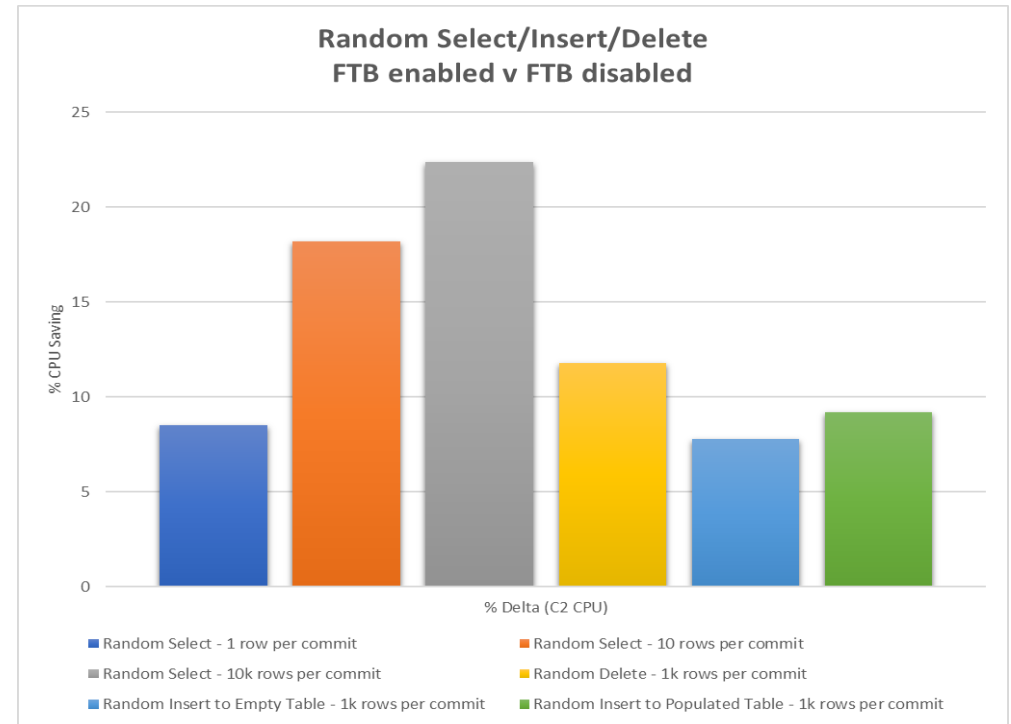
FTB Performance Expectations –General

- Most traditional OLTP workloads should benefit from FTBs, depending on
 - Access pattern (should be mostly random)
 - Depth of indexes (more levels will provide more benefit)
 - Memory allocated to the FTB pool
- IBM Lab / customer CPU savings for FTB are typically in the 2-3% range, but can be as high as 10-20% for some workloads



FTB Performance Expectations – DB2 12 Performance Topics Redbook

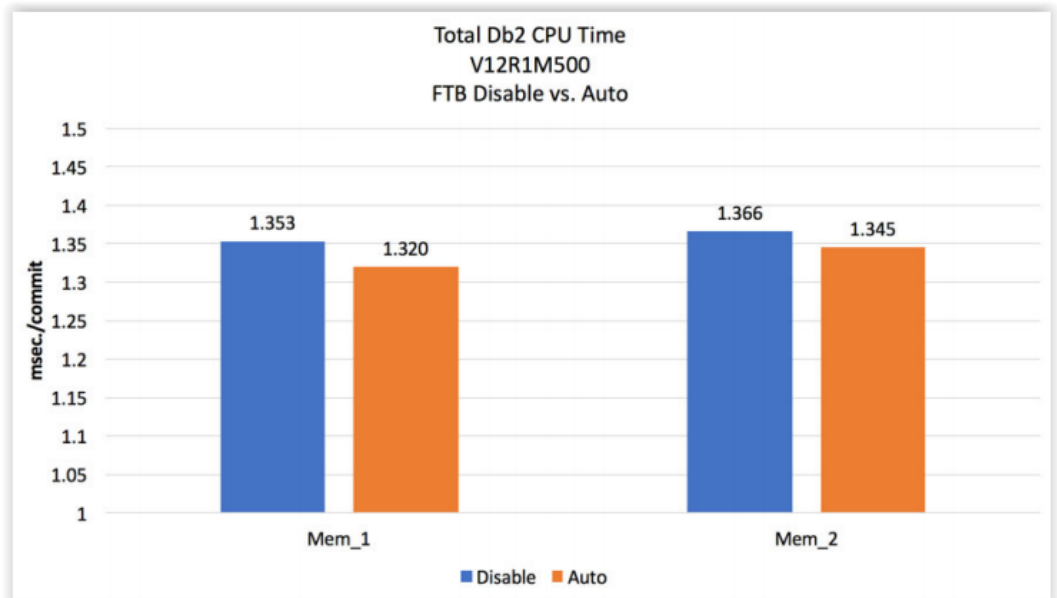
- Random index access workload
 - PBG table with one unique 5-level index and index-only access
 - Up to 22.4% C2 CPU savings in a non data sharing environment



Source: DB2 12 Performance Topics Redbook

FTB Performance Expectations – DB2 12 Performance Topics Redbook

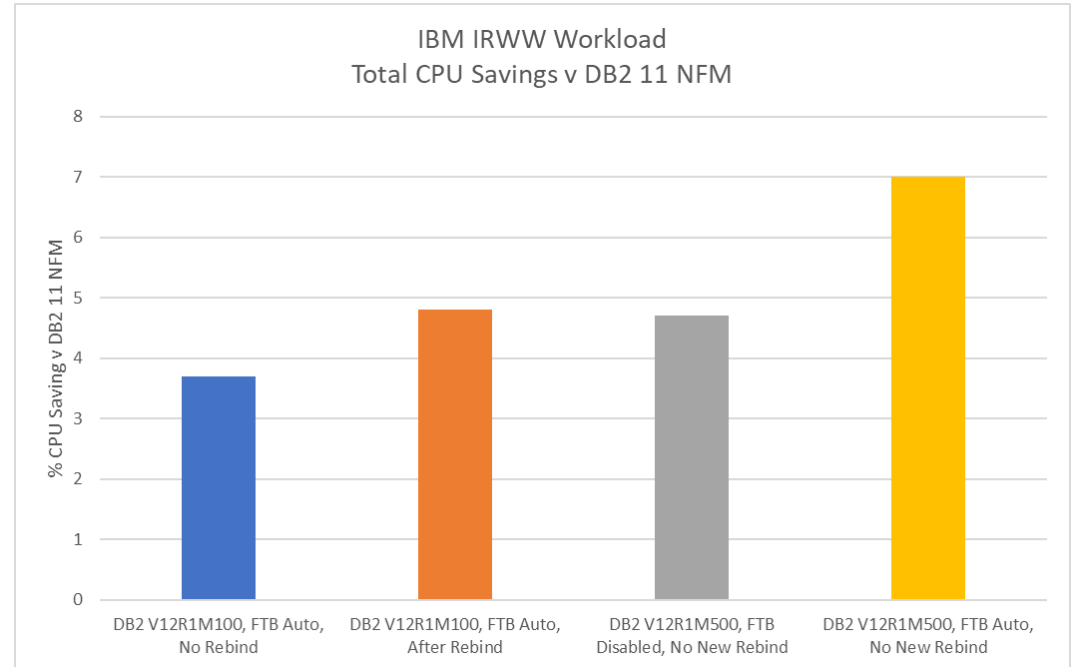
- IBM Brokerage Benchmark workload
 - Approx. 10k index objects in OLTP workload with 2-way data sharing
 - Average reduction of 25 getpages per TX and overall CPU reduction of 2% at V12R1M500



Source: DB2 12 Performance Topics Redbook

FTB Performance Expectations – DB2 12 Performance Topics Redbook

- IBM IRWW workload
 - 2-way data sharing, heavy OLTP transaction mix, with emphasis on random index access, ideal for FTBs
 - 7% reduction in average total CPU and 47% reduction in getpages at V12R1M500 compared to V11 NFM baseline (of which 2.3% reduction in average total CPU due to FTB feature)



Source: DB2 12 Performance Topics Redbook

FTB Performance Expectations – Data Sharing

- New kind of FTB p-locks and IRLM notify used during index structure modification (e.g. index leaf page split / consolidation)
 - Overhead of this can be noticeable within a heavy insert environment
- Somewhat unlikely as DB2 will not favour indexes with lots of structure modification, but if this happens consider manual override with `SYSIBM.SYSINDEXCONTROL`

Contiguous Buffer Pools Performance Expectations – DB2 12 Performance Topics Redbook

- OLTP Brokerage workload
 - BPs with high getpage count, sized so that all objects can fit in pool
 - No sync I/O for either test
 - PGSTEAL(NONE) showed 8% reduction in C2 CPU, and 7% reduction in elapsed, compared to PGSTEAL(LRU)

Source: DB2 12 Performance Topics Redbook

Contiguous Buffer Pools – Identifying Suitable Candidates

- Guidelines remain the same as for PGSTEAL(NONE) pools in V10 and V11
- Objects should be
 - Frequently accessed (read and / or write)
 - Relatively small (but bigger objects can increasingly be accommodated)
 - Stable in size
 - E.g. code / lookup tables, critical indexes not eligible to FTB
- DB2 12 Real Time Statistics includes GETPAGE counters and temporal support, which can help to identify suitable candidates
 - Process for using these is documented in new DB2 12 Performance Topics Redbook, using object size and “GETPAGE intensity”

Contiguous Buffer Pools – Allowing for the Overflow Area

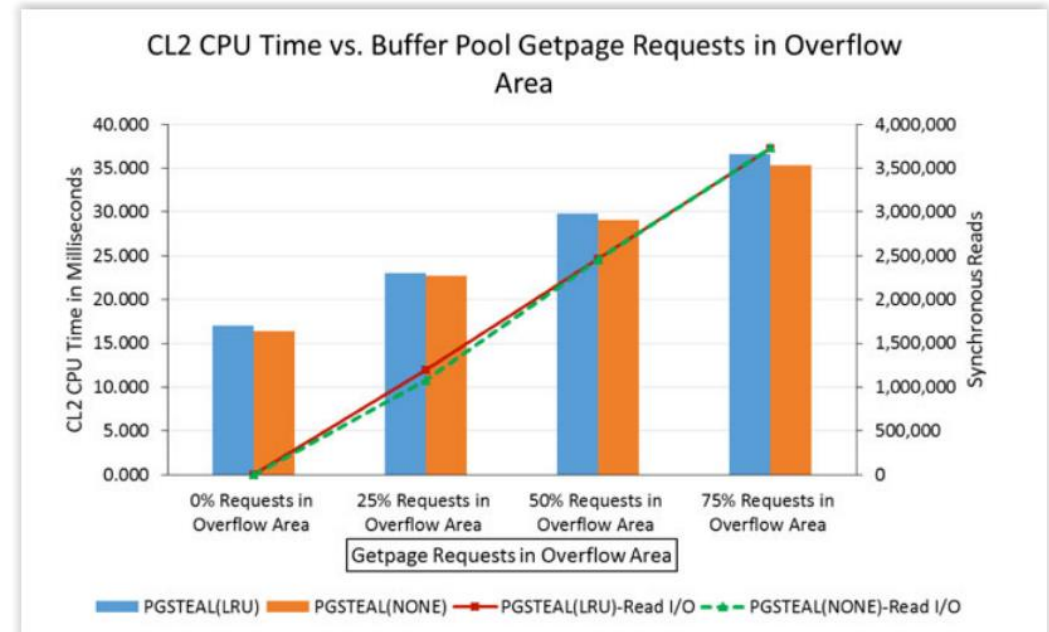
- 10% of buffer pool VPSIZE is set aside for the overflow area
 - Subject to minimum of 50 buffers and maximum of 6,400 buffers
- Objective is to size all objects to fit in main area, so make sure VPSIZE is increased accordingly
 - For most pools, this will mean just adding the 6,400 buffers back on to the calculated total
- Monitor for DSNB604I message, indicating that overflow area is in use
 - Only way to identify which specific object is in overflow

VPSIZE (Pages)	Main Area (Pages)	Overflow Area (Pages)
100	50	50
200	150	50
1,000	900	100
10,000	9,000	1,000
50,000	45,000	5,000
100,000	93,600	6,400
200,000	193,600	6,400

Example of requested/actual pool sizes with PGSTEAL(NONE) in V12

Contiguous Buffer Pools – Impact of Overflow

- V12 Performance Topics Redbook team conducted some controlled testing
 - Batch program, 100% of data in table accessed randomly in uniform distribution pattern
 - Varying VPSIZE values used to trigger overflow area (6,400 pages in all tests)
 - With 75% of requests going to overflow area, C2 CPU time is approximately doubled compared to properly sized BP



Source: DB2 12 Performance Topics Redbook

FTBs – Tuning Thoughts

- Somewhat limited real-world tuning knowledge available
- Sizing FTB Memory
 - If INDEX_MEMORY_CONTROL <> AUTO, consider increasing size of FTB pool if number of index candidates > number of indexes cached
 - ▶ DSNI070I: CANDIDATE OBJECTS > USED BY
 - ▶ IFCID002: QISTFTBCAN > QISTFTBNUMC
 - As always with memory tuning, law of diminishing returns will apply as you increase FTB pool
 - Assess whether memory could be better used elsewhere (buffer pools, DSC, etc)
- PCTFREE impact
 - Consider increasing PCTFREE and/or increasing REORG frequency to encourage FTB usage for more frequently updated indexes
- SYSINDEXCONTROL Overrides
 - Use sparingly: all opportunities to override DB2 autonomics can come back to haunt/bite you!

FTBs – Why Aren't FTB's Being Used?

- FTB use for a given index / partition is under DB2 control, many reasons why DB2 may chose not to cache an index (or to stop caching an index), including
 - Traverse count does not meet the traverse threshold within the interval
 - ▶ Access isn't random enough to reach traverse threshold
 - ▶ There is enough sequential access / page split activity to discourage FTB creation
 - DB2 has hit the FTB memory limit (INDEX_MEMORY_CONTROL) or max numbers of FTBs (internal limit of 10,000 indexes per member)
 - Paging – DB2 disables FTB creation if system paging is detected
 - DB2 12 Function Level < 500 and index becomes GBP dependent
 - Indexspace stopped via -STOP DB(...) SPA(...) command
 - Any ALTER INDEX statement (structural or not – possible future enhancement to ease this)
 - Mass delete
 - LOAD / REORG/ REBUILD utility activity

Agenda

- Introduction
- Background: The Big Memory Era
- In-Memory Features in DB2 12 for z/OS
- Practical Guidelines
- **Conclusions**

Conclusions

- “Big Memory” is a major strategic direction for both System Z and DB2 for z/OS, and the DB2 product is once again at the forefront of exploiting new System Z capabilities
- The CPU / performance savings offered by Contiguous Bufferpools and Fast Traverse Blocks are expected to be a significant part of the DB2 12 for z/OS value proposition
 - Approx. 2-3% CPU savings for many common OLTP workloads
 - Savings are “out-of-the-box”— both features available in BNFA with no application changes required
- Major new features
 - As with any other new feature, thorough testing is essential
 - Don’t be put off by number of FTB-related APARs, several issues found during ESP but nothing major is currently outstanding
 - Make sure you’re up to date on V12 maintenance 😊

Triton Consulting
51-59 Rose Lane
Norwich NR1 1BR
United Kingdom

email: julian.stuhler@triton.co.uk

web: www.triton.co.uk

Tel: +44 (0) 870 2411550

Fax: +44 (0) 870 2411549