SWG BetaWorks

DB2 9 for z/OS

Technical Education Series

“Scalability”

Important Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.

IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

- CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
- ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

Agenda

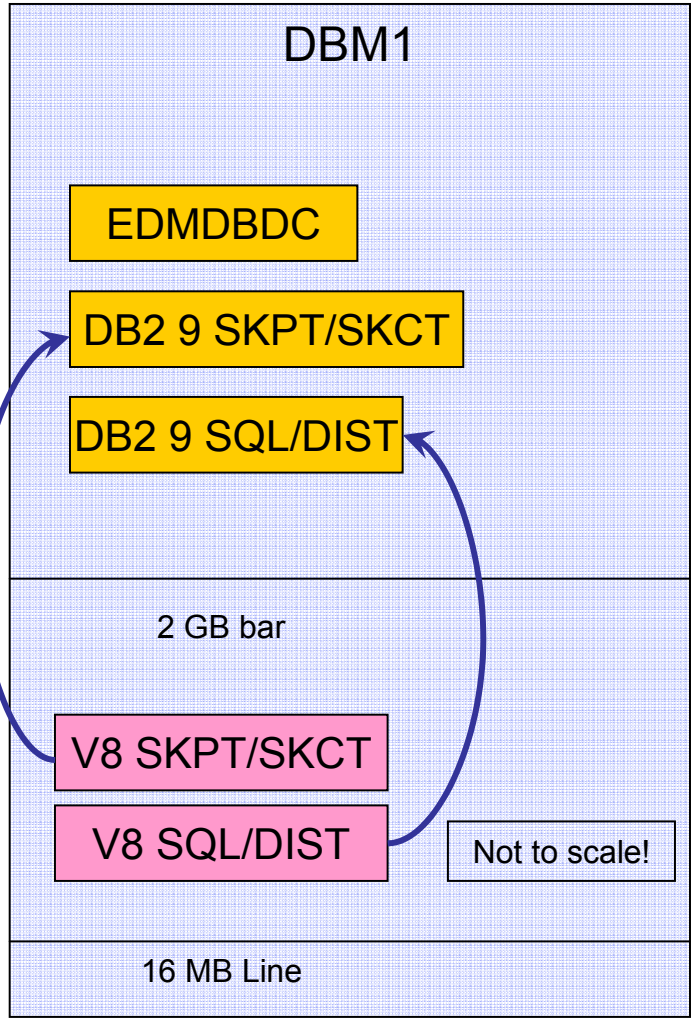
- **Virtual Storage Constraint Relief**
- **64-bit DDF**
- **WLM assisted BPOOL management**
- **Logging Enhancements**
- **Exploit WLM changes to fix latch contention**
- **Index compression**
- **Relief for sequential key insert**

SWG BetaWorks

Virtual Storage Constraint Relief

Virtual Storage Constraint Relief

- Virtual Storage Constraint is still an important issue for many DB2 customers. The following changes provide some relief:
- EDMPOOL Changes:
  - V8 – DBD storage moved above 2GB bar.
  - DB2 9 – SKCT, SKPT storage moved above 2GB bar.
- Other changes:
  - Some storage acquired for distributed applications moved above 2GB bar.
  - Some storage acquired for dynamic SQL statement execution moved above 2GB bar.



SWG BetaWorks

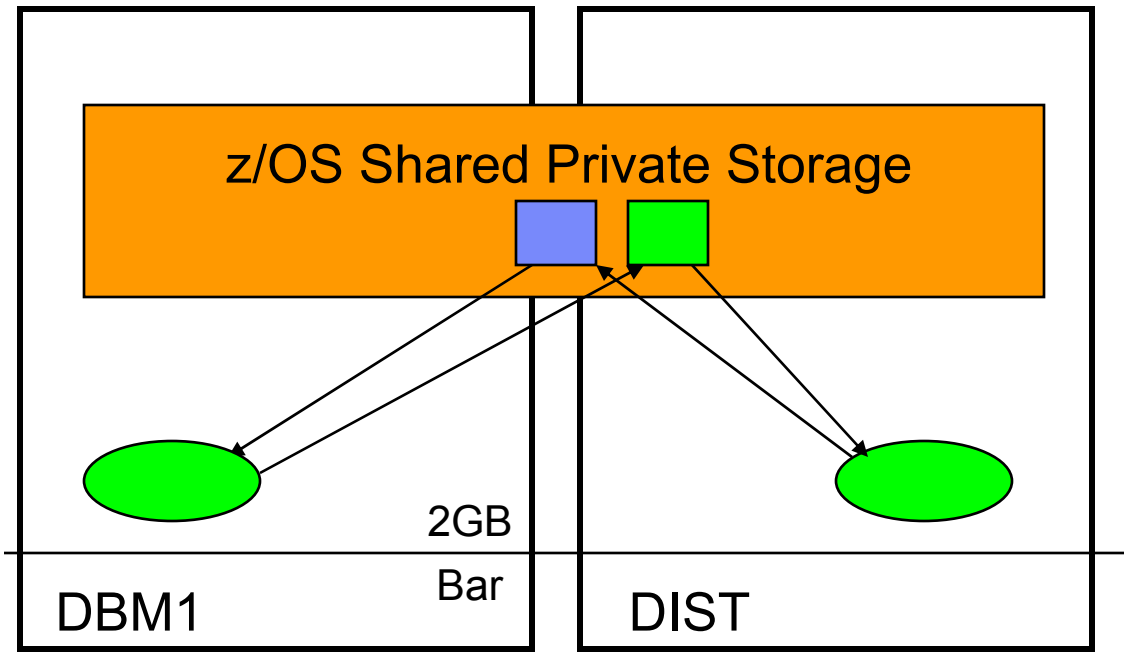
64-bit DDF – Shared Private with DBM1

64-bit DDF and DBM1: z/OS Shared Storage

- This is a performance enhancement for distributed server processing.
- The plan: use z/OS Shared Memory Facility to reduce data moves between DBM1 and DDF.
- The Shared Memory Facility was introduced in z/OS 1.5.
  - Shared memory: a new type of virtual storage allowing multiple address spaces to easily address storage.
  - Similar to ECSA – always addressable, no AR or XM moves needed.
  - Different to ECSA – not available to all address spaces on the system, only those registered with z/OS as being able to share this storage.
- Before DB2 9: when DDF invokes DB2 to process a request, data are copied from the DDF address space to DBM1 at the beginning of the request, and are copied back from DBM1 to DDF at the completion of the request.

DB2 9 Improvements for VStor Constraint Relief

- **DDF address space runs in 64-bit addressing mode**
  - Shared 64-bit memory object avoids xmem moves between DBM1 and DDF and improves performance
  - Constraint relief




```
RESPONSE=MV50
 IAR019I  23.20.10 DISPLAY VIRTSTOR 472
  SOURCE = DEFAULT
  TOTAL SHARED = 522240G
  SHARED RANGE = 2048G-524288G
  SHARED ALLOCATED = 131072M
```

- **Reduced amount of data formatting and data movement**
- **Reduced virtual storage**
  - It exists once, instead of in each address space
- **Has storage key & fetch protection**
- **Defaults to 2TB size**
- **DB2 requires a minimum of 128GB configured or DB2 9 will not run**
  - Even if not running DIST
- **Set by HVSHARE in Parmlib**
- **DISPLAY VIRTSTORE,HVSHARE**

DB2 Changes for 64-bit DDF and DBM1-Shared Memory

- DB2 9 for z/OS: many of these interface areas are now in shared memory, so no cross memory move will be necessary for those blocks.
- DDF is changed to run in 64-bit mode – will access Shared Memory areas.
- A new shared memory object (Virtual Shared Object or VSO) is created at DB2 initialization.
- As MSTR, DBM1 and DIST go through their local storage initialization, they are registered to use this VSO.
- As each address space is terminated during shutdown, they request that their interest in the VSO be deleted.
- The shared memory object is freed at DB2 termination.
- Expected SQL statement beneficiaries:
  - INSERT
  - UPDATE ... WHERE
  - DELETE ... WHERE
  - SELECT ... WHERE, singleton and cursor
  - CALL with parameters

Changes to Messages and IFCIDs

- DSNY011I and DSNY012I updated to indicate that z/OS V1.7 or above with the shared memory feature of z/OS is required.
  - DSNY011I DSNYASCP ******** Z/ARCHITECTURE WITH 64-BIT ADDRESSING AND SHARED STORAGE REQUIRED
    - System Action: DB2 startup abends with reason code 00E8005A.
  - DSNY012I DSNYASCP ******** THIS RELEASE OF DB2 REQUIRES Z/OS V1R7 OR ABOVE
    - System Action: DB2 startup abends with reason code 00E80058.
- IFCIDs 217 and 225 updated with fields for shared storage
  - DSNWMGS changes:

```
0217 QE0217SG TOTAL GETMAINED SHARED 64-BIT STORAGE FOR ASID
...
0225 QW0225SF TOTAL FIXED SHARED 64-BIT STORAGE FOR ASID
0225 QW0225SG TOTAL GETMAINED SHARED 64-BIT STORAGE FOR ASID
0225 QW0225SV TOTAL VARIABLE SHARED 64-BIT STORAGE FOR ASID
```

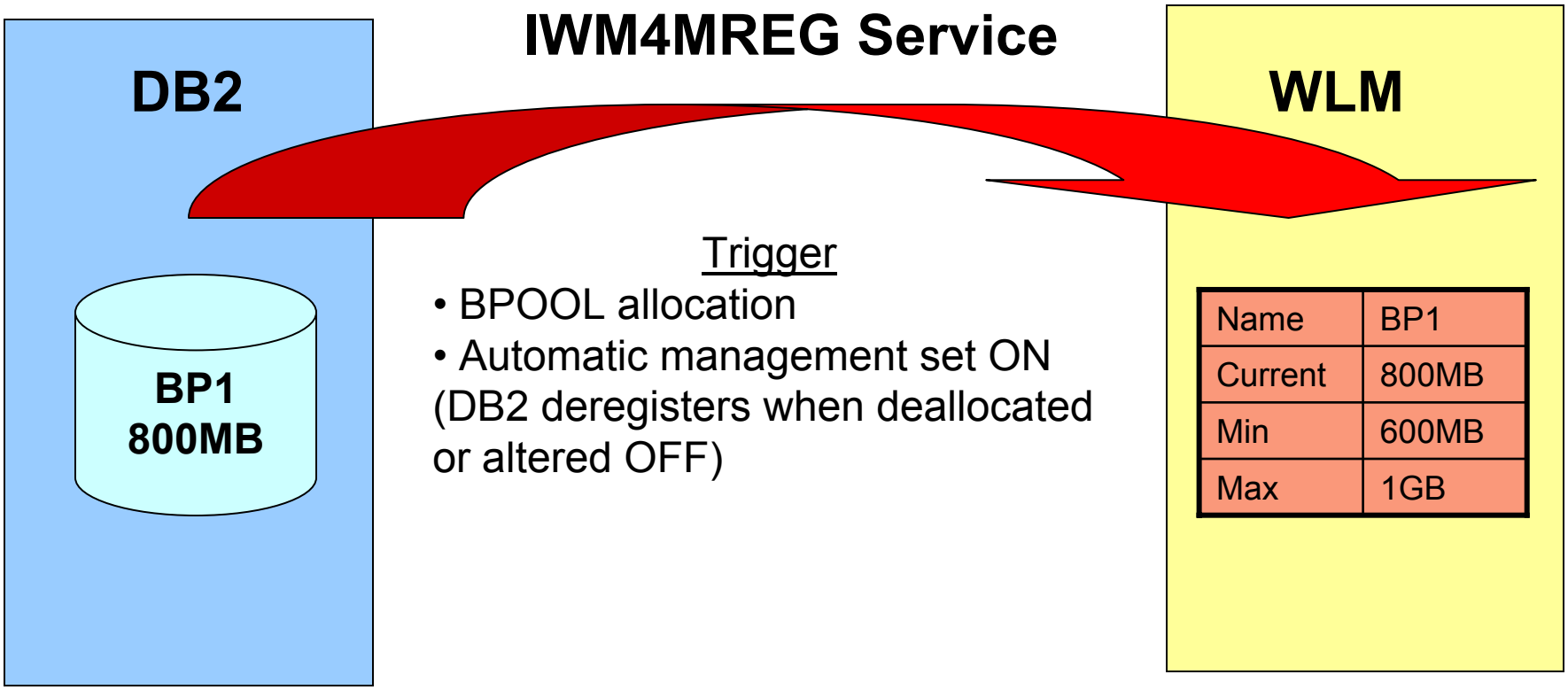
SWG BetaWorks

WLM Assisted BPOOL Management

Automatic buffer pool management

- Only the size attribute of the buffer pool.
- Can be enabled or disabled at the individual buffer pool level.
- Automatic management entails the following:
  - DB2 Registers the BPOOL with WLM
  - DB2 provides sizing information to WLM
  - DB2 communicates to WLM each time allied agents encounter delays
  - DB2 periodically reports BPOOL size and random read hit ratios to WLM

DB2 Registers BPOOL to WLM

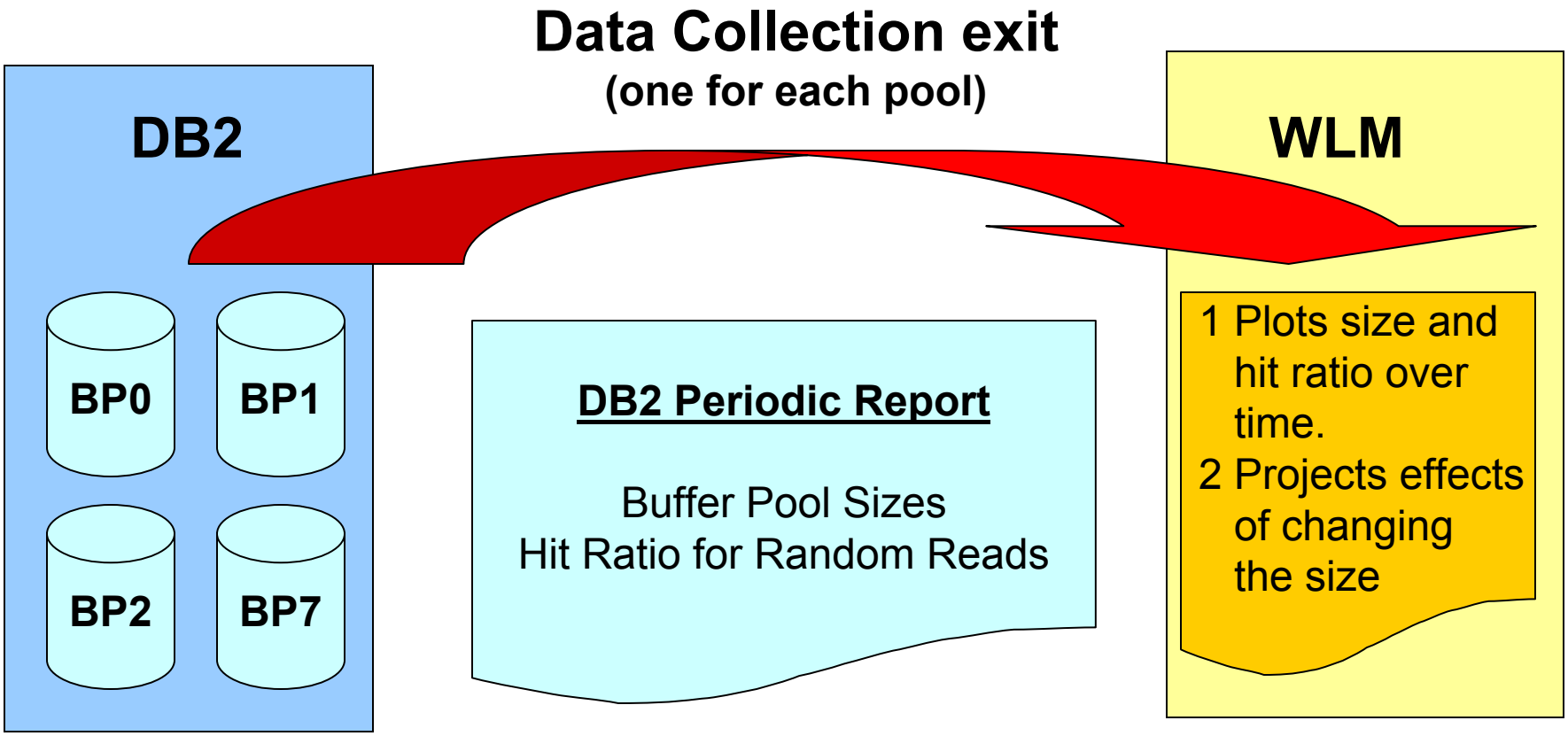


DB2 communication to WLM

Each time an allied agent encounters a delay caused by a random Get Page having to wait for read I/O.

- The following cases are not communicated to WLM:
  - Prefetch I/O
  - Wait for I/O on a sequential GetPage
  - Group buffer pool reads

Periodic reporting



WLM Information processing

- WLM uses the information to determine the best course of action if work is not achieving its goals.
- If WLM determines that waiting on buffer pool I/O is the predominant delay
  - It will assess whether increasing the buffer pool size could help to achieve the performance goals.
  - WLM will determine whether sufficient available frames exist on the system.
    - If so, then WLM notifies DB2's "resource adjustment" exit of the recommendation to increase the buffer pool size.
    - If not, then WLM will again consult its histogram data to assess whether taking storage from another buffer pool would result in an overall performance benefit.
    - If such a buffer pool is found, then WLM notifies that DB2 subsystem to reduce the buffer pool size by the prescribed amount.
  - Once this is accomplished, the first buffer pool can then be increased.

Buffer Pool adjusting

- If the buffer pool is adjusted, the result will be just as though an ALTER BUFFERPOOL VPSIZE command had been issued
  - The new size is stored by DB2 in the BSDS
- If the buffer pool is deallocated (e.g. because DB2 is being stopped) it will subsequently be reallocated at its most recently allocated size.

  Example
  - If BPOOL is adjusted from 800 MB to 900 MB
  - Then DB2 is stopped and restarted
  - BPOOL will be subsequently allocated at 900 MB
- Because DB2 registers the buffer pool using the allocated size, in this scenario the buffer pool would be registered at 900 MB with a min and max of +/- 25% of 900 MB.

What if the BPOOL is manually altered?

- If a buffer pool's size is manually altered (via the ALTER BUFFERPOOL VPSIZE command)
  - It is deregistered, and then registered at the new size.

- Example
  - BPOOL registered at 800 MB
  - Altered to a size of 1000 MB
  - Then after the alteration has completed, DB2 deregisters and re-registers the buffer pool at 1000 MB with a new min of 750 MB and a new max of 1250 MB.

AUTOSIZE option

- DB2 will increase or decrease the size of a given buffer pool by up to 25% of the originally allocated size.
- By default, automatic buffer pool adjustment is turned OFF.
- It can be activated via a new AUTOSIZE(YES) option on the ALTER BUFFERPOOL command.
- Once activated, it can be deactivated by:

  ALTER BUFFERPOOL(bpname) AUTOSIZE(NO).

- The AUTOSIZE attribute is added to the:

  DISPLAY BUFFERPOOL output.

New messages

- DSNB544I AUTOSIZE FOR bpname HAS BEEN SET TO nasize
  - A response to an ALTER BUFFERPOOL command, indicating that the requested change to the AUTOSIZE attribute has been accepted.

- DSNB555I WLM RECOMMENDATION TO ADJUST SIZE FOR BUFFER POOL bpname HAS COMPLETED

  OLD SIZE = csize BUFFERS

  NEW SIZE = nsize BUFFERS

  - Message issued when WLM notifies DB2 to adjust the size of a BPOOL.
  - Recommendation is made based on:
    - WLM's dynamic monitoring of the effects of buffer pool I/O on the achievement of workload goals
    - Amount of available storage on the system.

Changed message DSNB402I

BUFFERPOOL SIZE = vpsize	**BUFFERS AUTOSIZE = auto**
ALLOCATED = vpalc	TO BE DELETED = vptbd
IN-USE/UPDATED = vpcba	**BUFFERS ACTIVE = vpact**

- Message displayed by the DISPLAY BUFFERPOOL command:
  - vpsize - The user-specified buffer pool size
  - auto - the buffer pool AUTOSIZE attribute that is applicable to the current allocation of the buffer pool.
    - YES - Buffer pool uses WLM to automatically adjust the size of the buffer pool
    - NO - Buffer pool does not use WLM services for automatic buffer pool sizing
  - vpalc - Number of allocated buffers in an active buffer pool.
  - vptbd - Number of buffers to be deleted in an active buffer pool because of pool contraction.
  - vpcba - Number of currently active (not stealable) buffers in the buffer pool.
  - vpact - Number of allocated buffers which contain data.

Migration considerations

- Functionality is available in DB2 9 for z/OS compatibility mode or new function mode.
- There are no fallback or coexistence considerations.
  - If a buffer pool is defined with AUTOSIZE(YES) while in DB2 9, then the user falls back to V8, the AUTOSIZE(YES) option will be honoured upon remigration to DB2 9.
- The AUTOSIZE option is ignored while running in V8.
- IFCID 0201 ("alter buffer pool") additions / changes in support of function

SWG BetaWorks

Log Manager Enhancements

LRSN Increment Wait Reduction

- In data sharing mode prior to DB2 9 for z/OS, the LRSN value of each log record on a given member was unique.
  - DB2 will re-drive pulling of STCK for LRSN, if two successive log records for the same member have the same value.
- The LRSN value increments every 16 microseconds.
- On modern processors, this can cause significant delays for workloads that have a high rate of log record creation.
- In DB2 9 for z/OS NFM and ENFM, the wait for LRSN increment is eliminated for most log writes.
  - In DB2 9, LRSN values only have to be unique within a given page
    - Compensating for any DB2 9 CPU performance regression.
- Big performance boost from change to logging in data sharing especially when running on very fast processors: z900→z990→z9→z10.

Conversion of archive log processing from BDAM to BSAM

- DB2 V8 and prior releases use the BDAM to access DASD archive logs.
  - Allows rapid access to the target CI
  - At the expense of support for striped datasets and DFSMS compression (i.e. Extended Format or EF data sets).
- In particular, the inability to use striped datasets poses a problem in that, as DB2 supports striped active logs, creating non-striped archive logs can create a bottleneck if active logs are being cycled through quickly.
- DB2 9 for z/OS converts the current BDAM logic to corresponding BSAM logic which allows DASD archive logs to exist on EF data sets.
- This change requires DB2 9 for z/OS in new function mode or ENFM.
  - Hence no need for toleration code in DB2 V8.
- DB2 will not create EF archive logs when in DB2 9 for z/OS compatibility mode.

Conversion from AMODE(24) to AMODE(31)

- The DB2 V8 (and earlier) archive log processing code that interfaces with DFSMS frequently runs in 24-bit mode.
  - Forces the buffers used during read and write to reside below the 16MB line, which significantly limits the size of these buffers.
- Converting the code to AMODE(31), moves the buffers above the 16MB line, allowing them to be much larger.
- In addition, double buffering will be used when possible on log reads (filling the next buffer while processing the current one).
- Changes should result in improved archive log read and write performance.
- This change will be in effect for all DB2 9 for z/OS modes.

Support for >64K track DASD archive logs

- Since the inception of the xSAM access methods, there has been a limit of 64K tracks per DASD volume on the size of sequential and partitioned datasets.
- z/OS 1.7 lifts this limit with the introduction of DSNTYPE=LARGE support.
- DB2 currently supports active logs as large as 4GB.
- For 3390 geometry DASD, this requires 87,382 tracks and for 3380 geometry the requirement is 104,858 tracks. Both of these are above the 64K track limit, forcing installations that use 4GB logs to put archive logs on tape.
- By exploiting the new z/OS support for large datasets, installations will be allowed to have 4GB DASD archive logs.
- This function is only available in new function mode or ENFM
  - Hence no need for toleration code in DB2 V8.

Additional points to note

- DSN1LOGP output may include multiple log records with the same LRSN value on a single DB2 data sharing member.
- DSNJU004 output may list consecutive active or archive log data sets where the end LRSN value of the first is the same as the beginning LRSN value of the second.
- Both of the above conditions can occur in DB2 9 for z/OS NFM, and after falling back from new function mode to compatibility mode or ENFM.
- After converting to new function mode, if you decide to have archive logs striped and/or compressed, any recovery actions involving those archive logs must be performed on a DB2 9 for z/OS system.

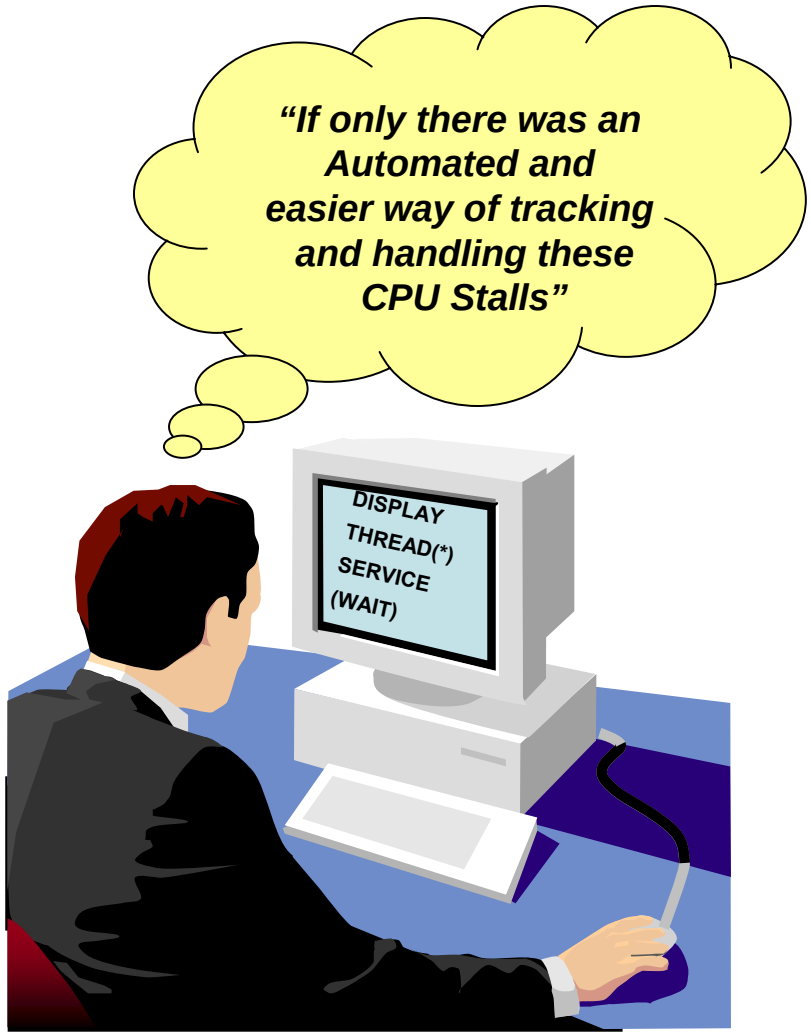
SWG BetaWorks

Exploit WLM changes to fix latch contention

Background

- **Two common issues facing DB2:**
  - CPU stalls
  - DBM1 storage constraints below the bar
- **Some manual identification techniques available today:**
  - Commands
  - Monitoring programs
- **DB2 does not however currently support automated methods.**
- **Many customers overlook such monitoring altogether.**
  - Often leading to unhealthy systems and outages.

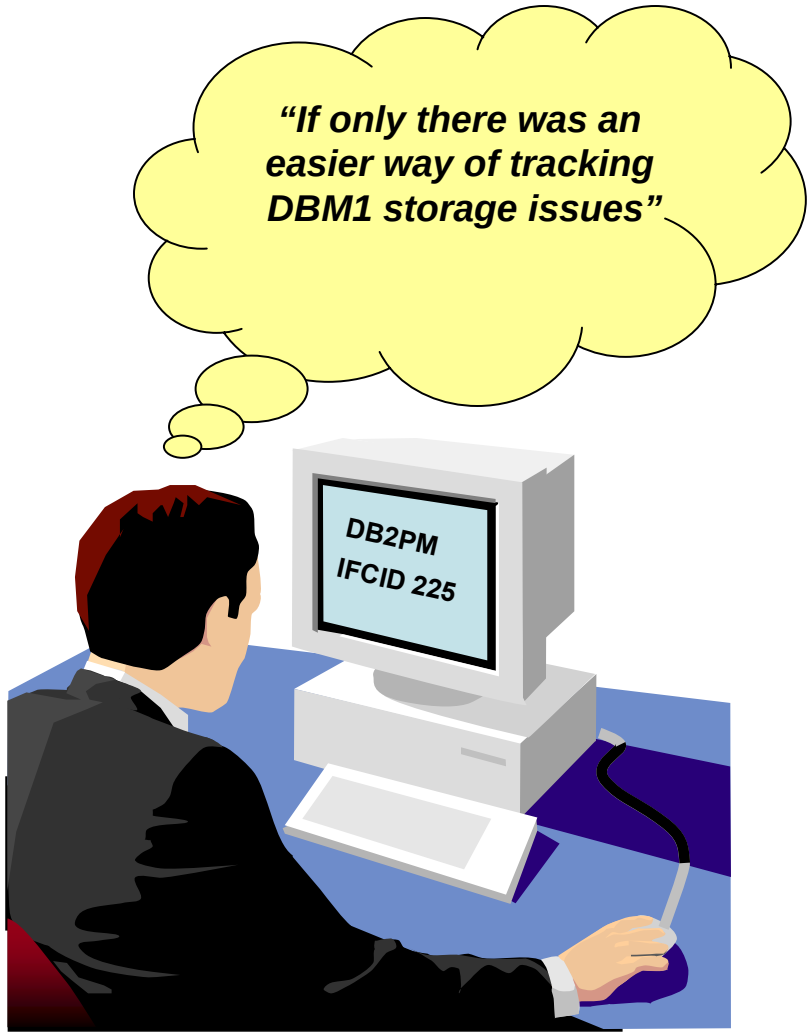
Manually handling CPU Stalls



Today DB2 has introduced a serviceability command to help identify and rectify some CPU stalls

DISPLAY THREAD(*) SERVICE(WAIT)

Manually handling DBM1 storage issues



Use a monitor / reporting program to analyze IFCID 225 records

- DBM1 storage constraint issues are typically identified via the IFCID 225 record analyzed by a DB2 monitor program.
- John Campbell's presentation / audio with accompanying spreadsheet

http://www.ibm.com/support/docview.wss?rs=64&context=SSEPEK&uid=swg27006985

The DB2 9 enhancement – “An Internal Monitor”

- An internal monitor.
  - Automate problem identification and correction.
- Will identify CPU stalls that result in “Latch Contention”.
  - Attempt to clear via a temporary WLM priority boost.
- Will monitor DBM1 storage below the bar for critical storage increases.
  - When thresholds reached Console messages will report:
    - DBM1 storage consumption.
    - The agents consuming the most storage.
- The “monitor” (more autonomic than just a monitor as will attempt to fix some problems)
  - Should lower the Cost of Ownership for DB2 and help ensure healthy DB2 systems.

DISPLAY THREAD SERVICE extension

- Extended to include a STORAGE option (Abbreviated to STG).

  DISPLAY THREAD(*) SERVICE(STORAGE)

- When the keyword TYPE specifies SYSTEM or ACTIVE threads, a V492 serviceability message will be included in the DISPLAY output.

```
DSNV401I  @ISC9 DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  @ISC9 ACTIVE THREADS -
NAME      ST A  REQ  ID              AUTHID   PLAN      ASID    TOKEN
RRSAF      T        8  ISC9ADMT0066  SYSDSP   ?RRSAF    00B4        2
V492-LONG 140 K VLONG 28 K 64 1028 K
RRSAF      T      363  ISC9ADMT0001  SYSDSP   ?RRSAF    00B4        3
V492-LONG 140 K VLONG 28 K 64 1028 K
TSO        T  *     3  COOKI         COOKI              00B5       16
DB2CALL    T        4  COOKI         COOKI    KO2PLAN   00B5       24
V492-LONG 64 K VLONG 28 K 64 0 K
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  @ISC9 DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
```

Internal monitor tasks

- **The primary monitor task runs in DB2′s MSTR address space.**
  - It is recommended that MSTR run at SYSSTC dispatching priority.
- **Monitor tasks will also run for the DBM1 and DIST (If started)**
  - Priority will be MSTR with interval of 1 minute
  - DBM1, with interval of 2 minutes (Inactive unless MSTR monitor task stopped)
    - An active monitor and some failover monitors to ensure our monitor doesn’t get stalled.

Active monitor processing

- A CPU stall scan
  - Looking for stalled agents (system, DBAT and Allied)
  - Grant a temporary priority boost via WLM services, to the latch holder..
- Storage consumption analysis
  - When DBM1 below the bar storage crosses thresholds of 88, 92, 96, or 98 percent of the available storage, a DSNV508I message will be issued.
- **System health status will be reported to WLM.**
- **All monitor tasks are stopped during DB2 shutdown.**

Internal Monitor messages

- DSNV507I **- active, inactive> MONITOR, INTERVALS=, STG=, BOOSTS=**
- DSNV508I **- DB2 DBM1 BELOW-THE-BAR STORAGE**
- DSNV509I **- DB2 <monname> INTERNAL MONITOR STOPPING**
- DSNV510I **– BEGIN DISPLAY OF LARGEST STORAGE CONSUMERS IN DBM1**
- DSNV511I **- END DISPLAY OF LARGEST STORAGE CONSUMERS IN DBM1**
- DSNV512I **- AGENT <rank>**
  - greatest consumer of 31-bit storage below the bar in DBM1 address space

Summary

- The MSTR address space should have SYSSTC priority.
- The DB2 health monitor starts up automatically.
- Monitors for processor stalls that result in latch contention, and attempts to clear the stalls through a temporary WLM priority boost.
- Monitors use of storage below the 2-GB bar and issues console messages to indicate the state of that storage.

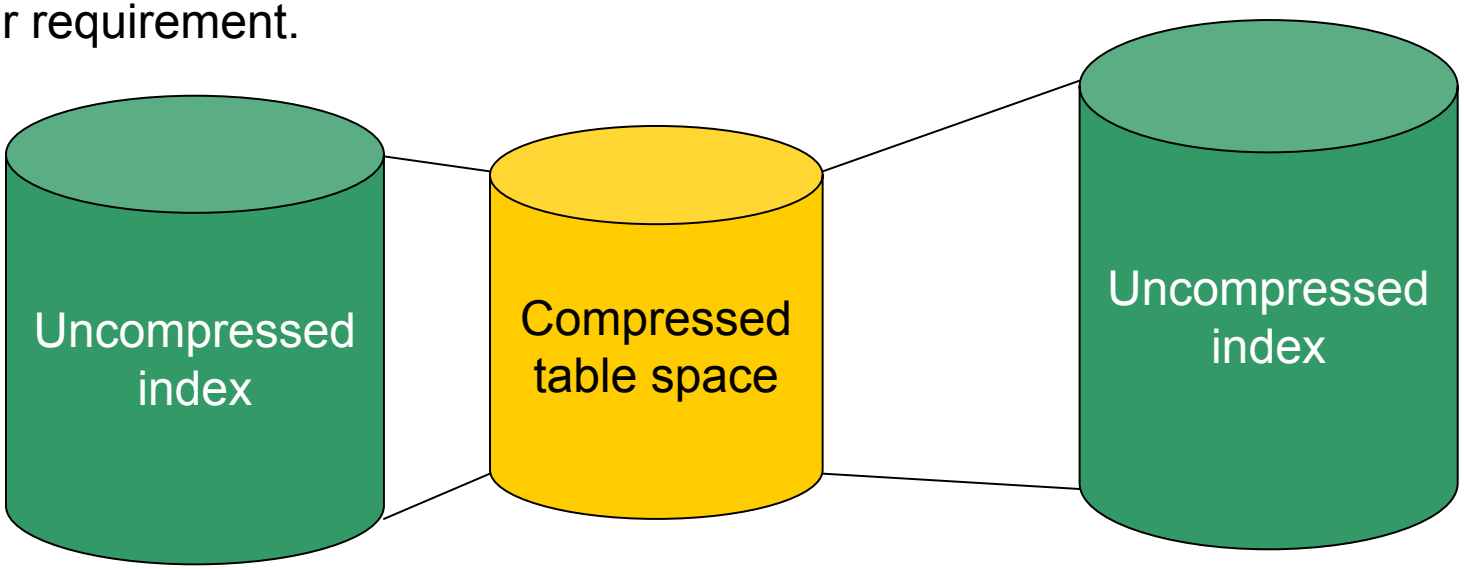
- DB2 now automates monitoring for and correction of processor stalls and monitoring of storage use and reporting of storage shortages.
- Should fix some problems, help provide early warning of others and in the worst case more diagnostic information.

SWG BetaWorks

Index Compression

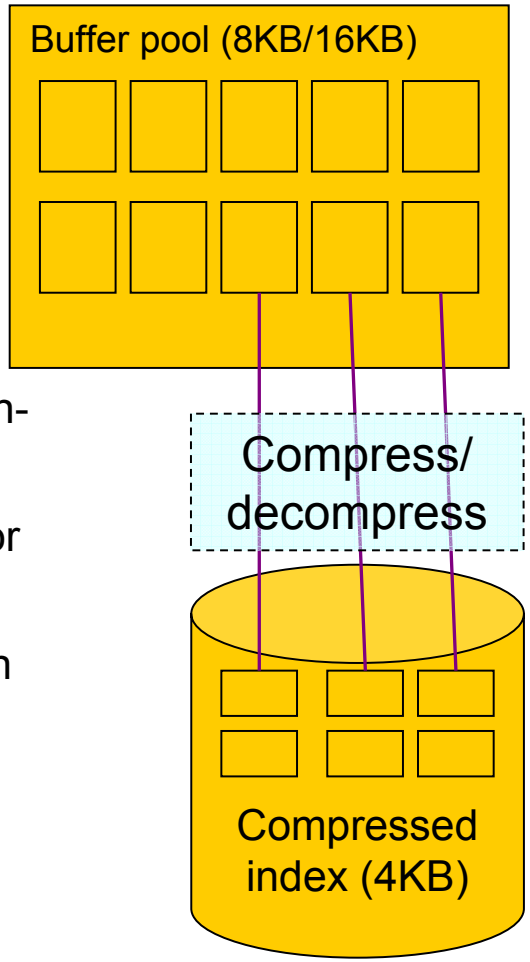
Why Index Compression?

- Support for compressed rows within a table, but not for indexes.
- Data Warehouse applications can create a large number of very large indexes (e.g. Star Schema)
- Index spaces can be much larger than the table they are based upon
- Shrinking index storage can contribute to lowering the TCO for DB2.
- Possible performance benefits
- Customer requirement.



How Index Compression is Achieved

- "Dictionary-less" compression
  - No dictionary – a compression algorithm is used.
  - Software compression
  - Compression on disk but not in bufferpool
    - Prefix compression
    - RID compression
  - An index page on disk will contain both compressed and non-compressed data
- Index pages on disk as 4KB pages, but in buffer pool as 8KB or 16KB pages.
- Physical 4KB page expanded to 8KB or 16KB page to reside in buffer pool
  - After read from disk, a "decompress" routine expands the page.
  - Prior to writing a to disk, a "compress" routine shrinks the page.
- Index pages are only updated and logged in uncompressed format



DDL, Catalog and Utility Changes

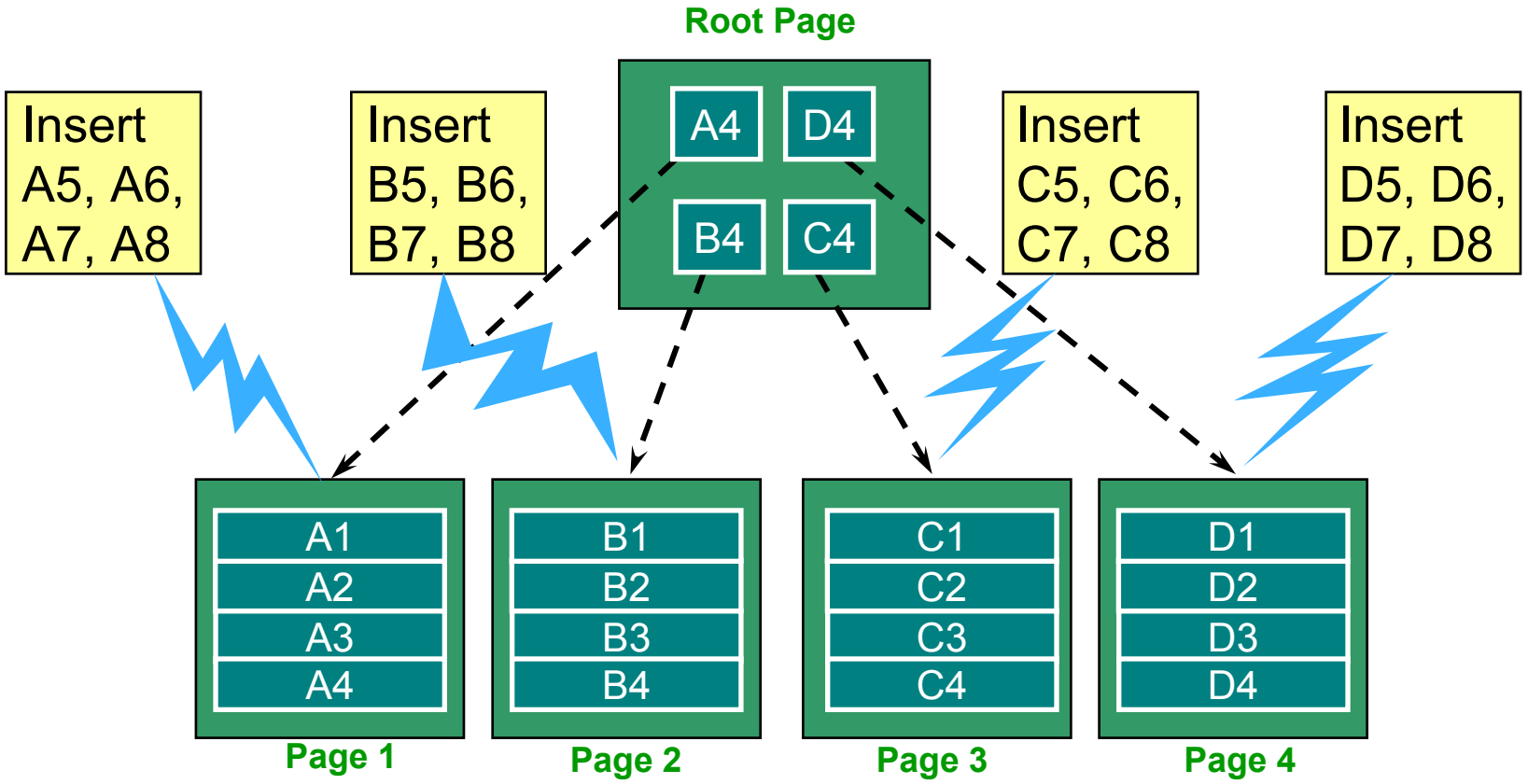
- New keyword COMPRESS YES | NO for CREATE/ALTER INDEX
- ALTER INDEX COMPRESS YES
  - Marks the index as REORG Pending
- Index compression requires an 8K or 16K buffer pool page size.
- New column “COMPRESS” added to SYSIBM.SYSINDEXES:
  - Indicates if index compression is active (‘N’ or ‘Y’; blank (' ') indicates the index was created prior to DB2 9 for z/OS NFM).
- Utility changes
  - DSN1PRNT, DSN1COPY now print uncompressed index page contents.
  - DSN1COMP
    - Before DB2 9 for z/OS, estimated space savings for compressed table spaces
    - Can now estimate space savings for indexes and recommend 8KB or 16KB page size.

SWG BetaWorks

Relief for Sequential Key Insert

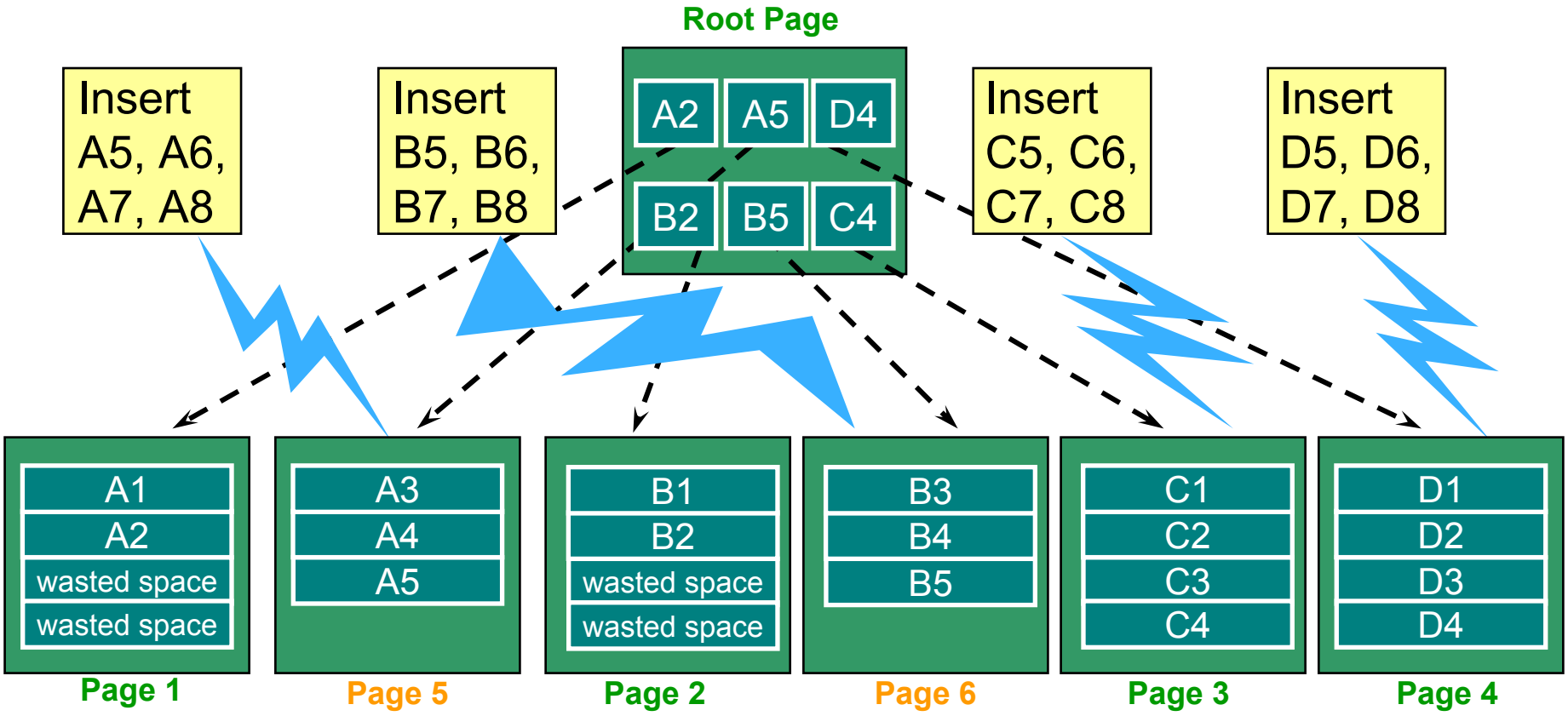
Example Sequential Key Insert Before DB2 9 – 1

- **In this example, an index is updated by parallel insert processes.**



Example Sequential Key Insert Before DB2 9 – 2

- **The effect of page splits – after inserts of keys A5 and B5**

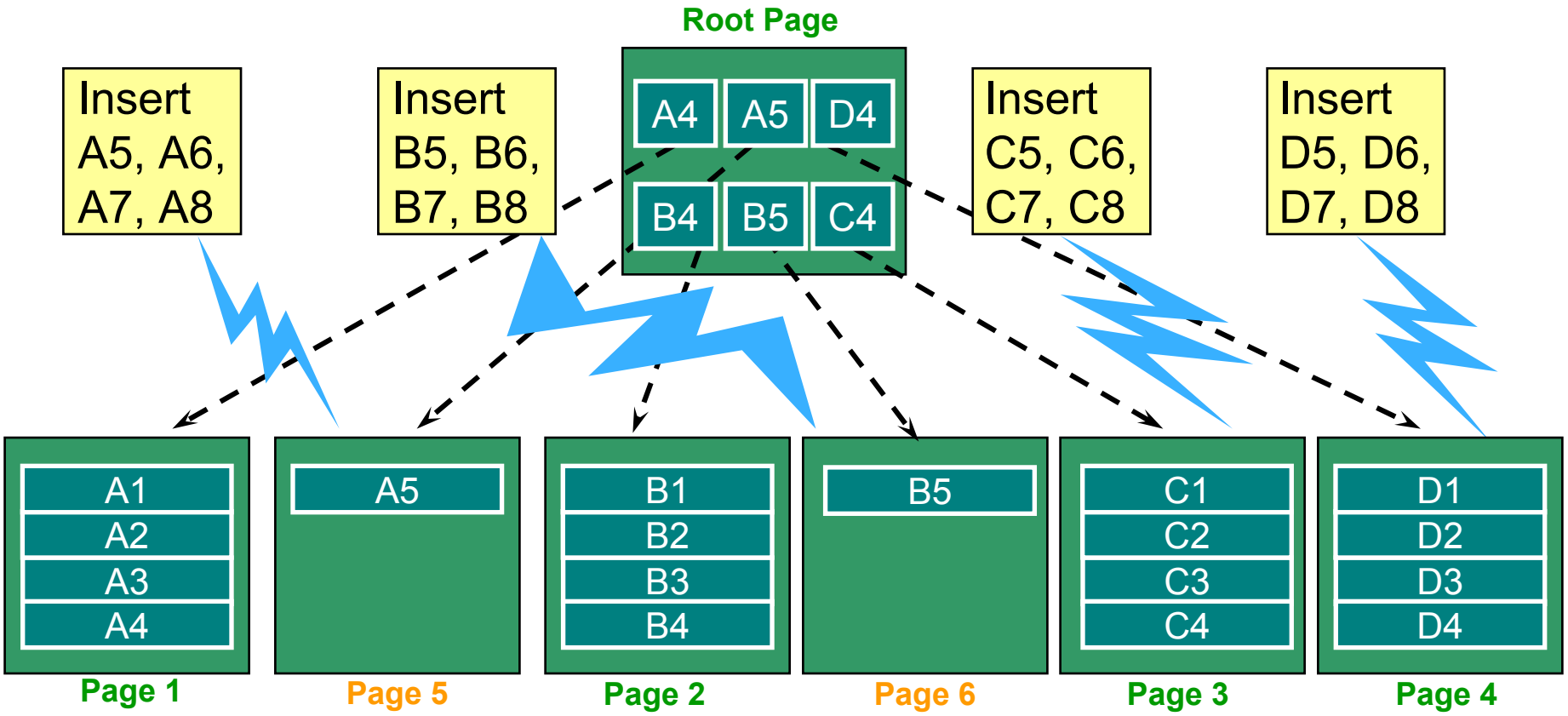


Relief for Sequential Key Insert

- Addressing the problem in two ways:
  1. Asymmetric splitting for index pages introduced to accommodate varying patterns of inserts into an index.
  2. Larger than 4KB page sizes allowed for indexes.

Sequential Key Insert With Asymmetric Page Splits

- **Asymmetric index page splits lead to more efficient space usage and reduces index tree contention.**



Page Sizes Larger than 4KB Allowed For Indexes

- **Before Version 9, only a 4KB buffer pool can be specified on the CREATE INDEX statement.**
- **Version 9 allows indexes to have pages that are greater than 4KB in size**
  - 8KB, 16KB, and 32KB buffer pools allowed: CREATE/ALTER INDEX and DATABASE.
- **Only available in DB2 9 for z/OS NFM**
- **Index page size considerations:**
  - A larger page size can improve performance for indexes with sequential insert and fetch patterns.
  - It can cause degraded performance for random index access, as larger amounts of data have to be read in from DASD.
  - A smaller page size can (depending on key size) provide better performance for indexes with random access patterns.
- **Operational support:**
  - DSN1PRNT, DSN1COPY and RECOVER support.

"Thank You for listening"

If you have any questions on this DB2 9 for z/OS session, then please send them to the BetaWorks team at:

Ian_Cook@uk.ibm.com
FLETCHPL@uk.ibm.com