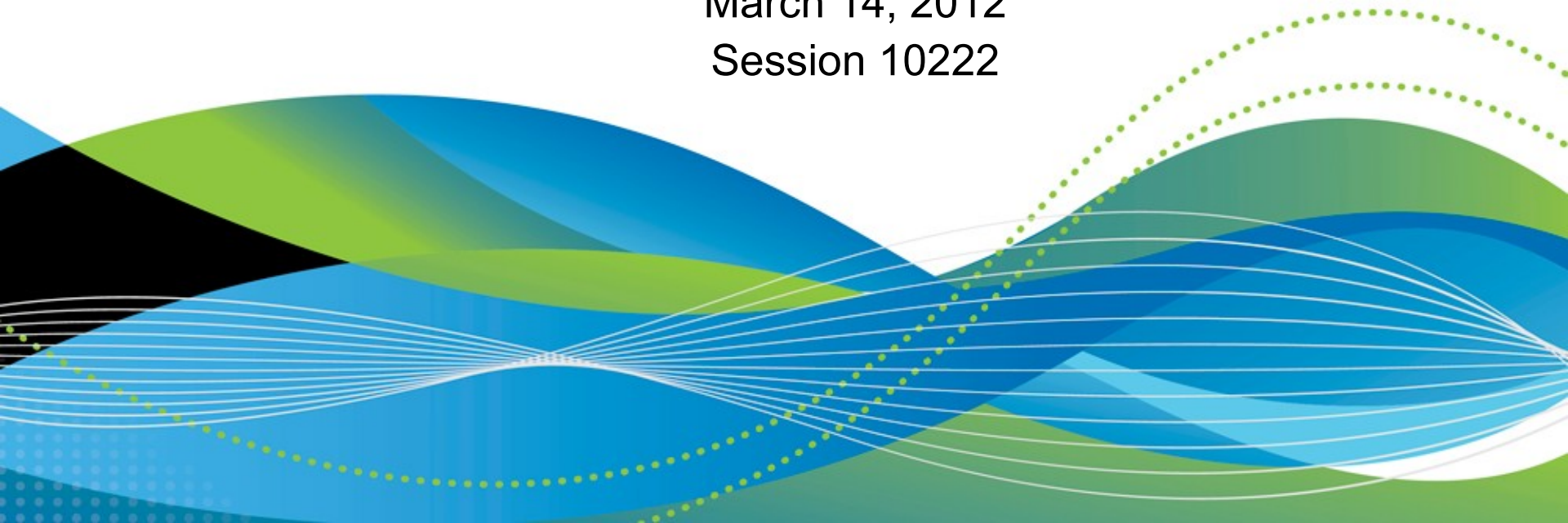
#SHAREorg

SHARE
Technology · Connections · Results

# DB2 for z/OS Data Sharing – Then and Now

Robert Catterall
IBM

March 14, 2012
Session 10222



# Agenda

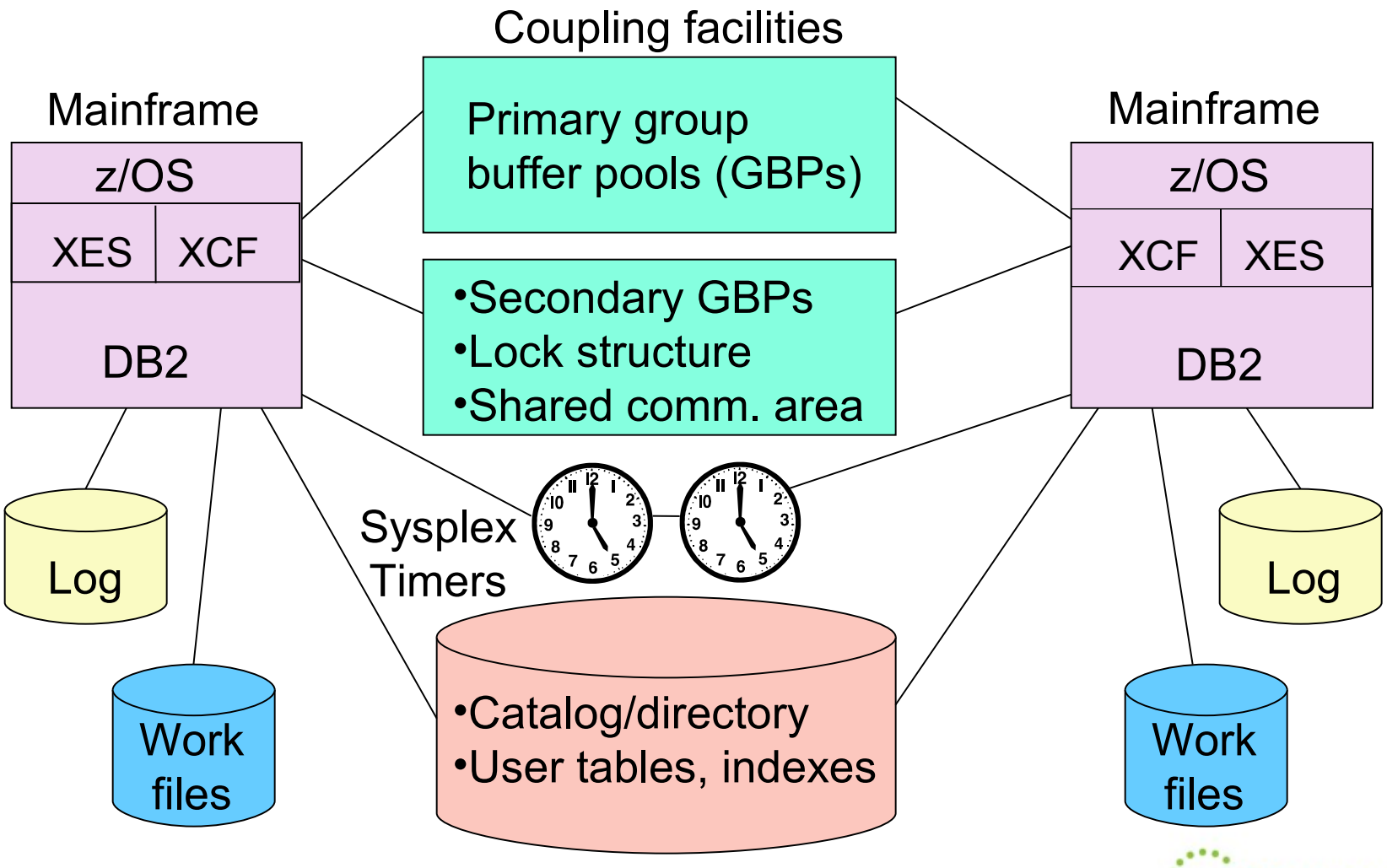
- A quick overview of DB2 data sharing
- Motivation for deployment – then and now
- DB2 data sharing / Parallel Sysplex configuration – then and now
- Data sharing locking – then and now
- Data sharing performance – then and now

# A quick overview of DB2 data sharing

# The basics of data sharing

- Multiple DB2 subsystems share read/write access to a database
  - The different subsystems are *members* of a *data sharing group*
    - Architecture allows for up to 32 members in one group (the biggest group I know of has 16 members – some might be larger than that)
  - DB2 data sharing runs on a Parallel Sysplex mainframe cluster
- Data sharing was introduced with DB2 V4 (mid-1990s)
  - Very robust technology, proven in all kinds of industries, all over the world
  - The most highly scalable, highly available data-serving platform on the planet

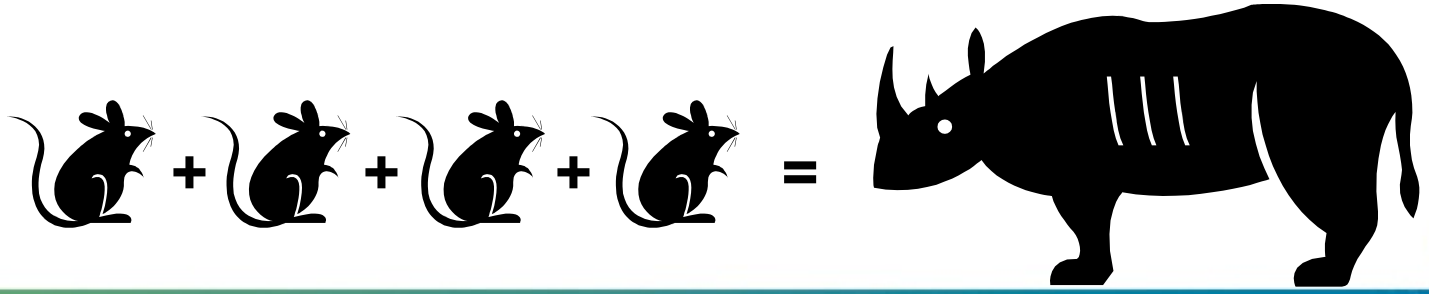
# The big picture



# Motivation for deployment – then and now

# Then: scalability

- IBM had just introduced the first CMOS-based mainframes, replacing systems based on more expensive (and faster) bipolar chip sets
  - Initial models had 5 MIPS per engine, with a max of (I think) 10 engines on one server
  - To migrate large workloads off bipolar systems (ES/9000) to CMOS systems (9672), you had to lash several of the latter together
    - Parallel Sysplex was the means of doing that, and data sharing enabled the multiple mainframes in a cluster to operate (and appear to application programs) as a single-image DB2 system

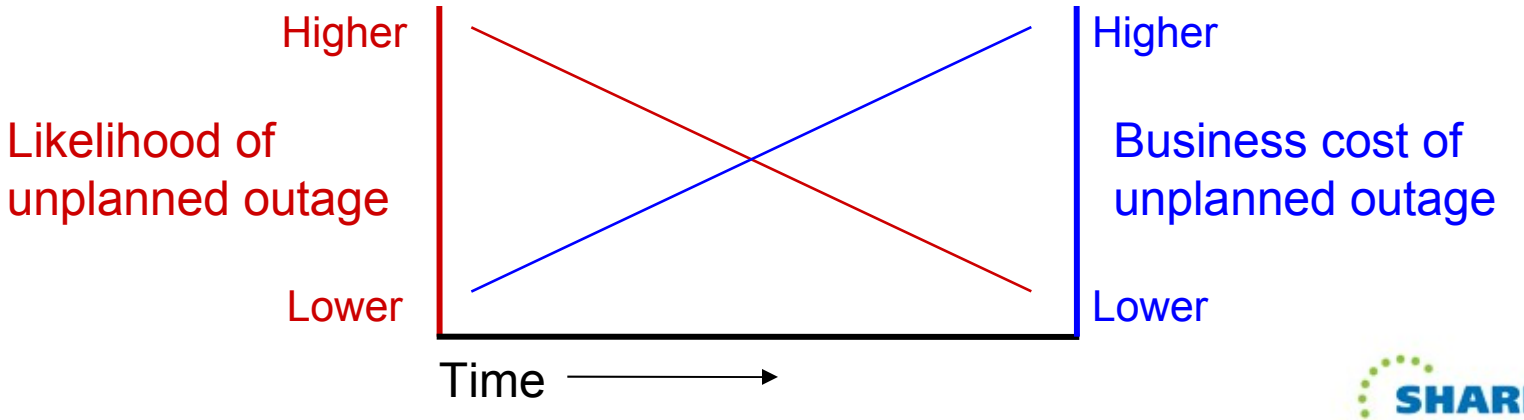


# Now: availability is #1 motivation

- Scalability is still a motivator, but given advances in IBM mainframe technology, there are fewer application workloads that won't fit on one System z server
  - Now: more engines per mainframe (up to 96 for zEnterprise)
  - More MIPS per engine (around 1000 for zEnterprise)
- Used to be a lot of Parallel Sysplexes with > 2 mainframes
  - Now, lots of 2-mainframe clusters (though number of DB2 data sharing members can be significantly larger than number of System z servers in the Parallel Sysplex)

# Availability: unplanned outages (1)

- As System z hardware, software continue to become more and more reliable, these failures are becoming less common
  - HOWEVER, as importance of 24x7 operations becomes ever more critical, business cost of downtime continues to go up
    - Plants are idled
    - Products don't ship
    - Customers are lost ("competition is only a click away")



# Availability: unplanned outages (2)

- Parallel Sysplex / DB2 data sharing advantage: reduce scope of unplanned outages:
  - Probably only a small portion of database will be unavailable if a DB2 member (or a z/OS LPAR or a System z server) fails
    - The pages (or rows) that were X-locked by processes running on a DB2 member at the time of the member's failure
  - Unavailable pages/rows freed up when failed member restarted, and DB2 restart is FASTER in a data sharing environment than in a standalone DB2 environment (often less than 2 minutes)
    - Faster restart: changed pages are externalized (to group buffer pools in the coupling facilities) at commit time (versus being written to disk when buffer pool deferred write threshold hit or at DB2 checkpoint)
      - *Result: roll-forward part of restart processing is accelerated)*

(restart usually automated via z/OS Automatic Restart Manager policy)

# Availability: planned outages

- With unplanned outages becoming more rare, focus has shifted to avoidance of planned outages
  - Usually scheduled for hardware or software maintenance
  - With a DB2 data sharing group, almost any maintenance activity can be performed without the need for a maintenance window
  - Example: upgrade DB2 maintenance:
    1. Apply fixes to DB2 load library
    2. Quiesce one member of the DB2 data sharing group (work continues to flow to other members)
    3. Stop and restart the quiesced member to activate the DB2 maintenance, and resume flow of work to that member
    4. Repeat steps 2, 3 for other members until maintenance updated for all
  - Same basic “round-robin” approach can be used for server and z/OS maintenance, and for DB2 version migration

# Outage-less DB2 migration (1)

- Old conventional wisdom: do not run CATMAINT (which makes catalog and directory changes needed for new DB2 release) concurrently with application workload
  - For standalone DB2 systems, this was not such a big deal, as you have to stop and restart DB2 anyway to activate new release
  - For data sharing systems, stop and restart doesn't require workload outage, as members can be stopped/started in round-robin fashion, and different DB2 versions can coexist in same data sharing group
    - But you still need to stop workload for CATMAINT, right?
    - WRONG: CATMAINT (and CATENFM) can run concurrently with applications

# Outage-less DB2 migration (2)

- If you run CATMAINT concurrently with a DB2-accessing application workload (this applies to CATENFM, too)…
  - Possible that some programs that access DB2 catalog/directory objects might fail with a timeout or a “resource unavailable” code
  - Also possible that CATMAINT itself might fail due to contention with application programs
    - If that happens, it’s NOT a disaster
    - Terminate job with -TERM UTILITY, re-execute from the beginning (actually, resubmit job DSNTIJTC, which executes CATMAINT)
  - To minimize contention between CATMAINT, application programs:
    - Run CATMAINT during a period of relatively low application activity
    - Avoid executing DDL statements while CATMAINT is running
    - Avoid package bind and rebind activity while CATMAINT is running

# Data sharing and availability and $$

- Some say, “Data sharing is too expensive – we can’t afford it”
  - Their assumption: you need multiple mainframe server “boxes” in order to implement Parallel Sysplex and data sharing
  - In fact: while having 3 “boxes” optimizes availability by eliminating single points of failure, you CAN get MAJOR availability benefits by implementing a “1-box” data sharing group
    - 2 z/OS LPARs, 2 DB2 members, 2 internal CFs in 1 System z server
    - Failure of whole box will fail entire group, but box failure is very rare
    - Still get benefit of software maintenance without maintenance window
    - Still get benefit of reduced scope of DB2 or z/OS LPAR failure (failure impact: some data unavailable until retained locks cleared)
    - If you do lose entire box, restart processing will take longer to complete versus restart of a single DB2 subsystem, due to recovery of objects in group buffer pool recover pending status (GRECP)

# Availability and DVIPAs (1)

- Terminology:
  - VIPA (Virtual IP Address) – a means of disassociating an IP address on a z/OS system from a physical adapter
  - DVIPA (Dynamic VIPA) – a VIPA that can move from one TCP/IP stack in a Sysplex to another
  - Distributed DVIPA – a special type of DVIPA that can distribute connections within a Sysplex
    - This is the DVIPA of the Sysplex Distributor
    - It's also the DVIPA for the data sharing group
  - Sysplex Distributor – a z/OS component that leverages DVIPA and WLM to maximize server availability in a client/server environment

# Availability and DVIPAs (2)

- If data sharing group used for DRDA client/server computing:
  - Assign a DVIPA to each DB2 member
    - That way, if member fails and is restarted on another z/OS LPAR in the Parallel Sysplex, requesters utilizing DRDA 2-phase commit protocol will be able to find it (important for resolving in-doubt DBATs)
    - NOTE: prior to DB2 10, “restart light” (free up retained locks, then shut down) does NOT resolve in-doubt DBATs, because DDF isn’t started for restart light (so, normal restart needed to resolve in-doubt DBATs)
    - DB2 10: DDF restart light, to enable resolution of in-doubt DBATs
  - Assign a DVIPA (“distributed DVIPA”) to the Sysplex Distributor
    - That way, an initial client request to connect to the data sharing group will succeed, as long as at least one DB2 member is active
    - After that initial connection request, distribution of subsequent requests from the client is managed by DB2 members and WLM

# DB2 data sharing / Parallel Sysplex configuration – then and now

# Then: coupling facility structure sizes

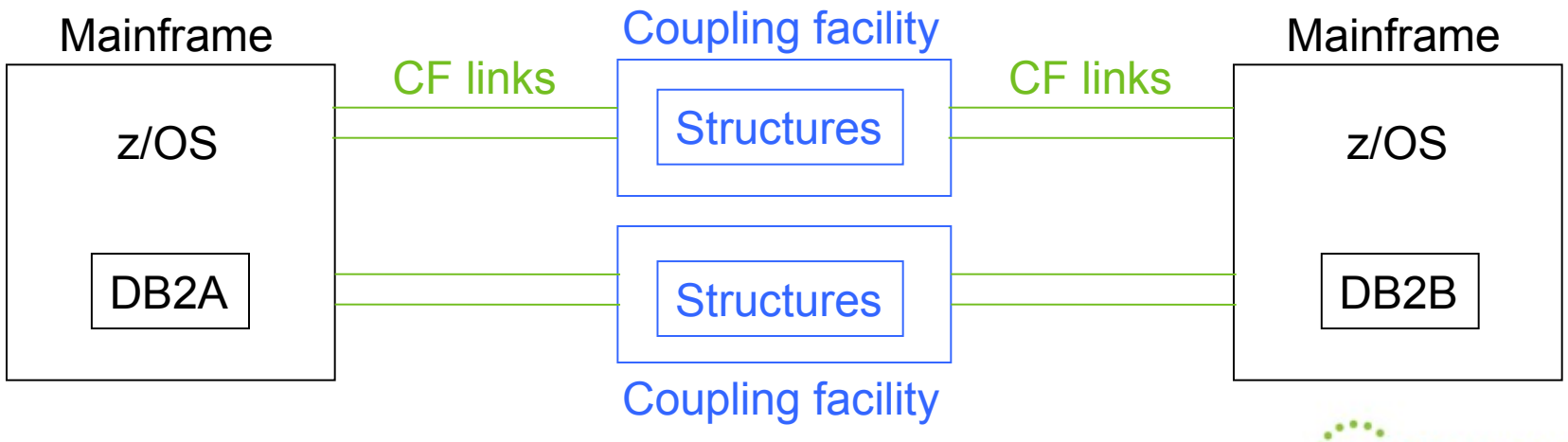
- A “then and now” look at DB2-related CF structures is relevant to group buffer pools (these are usually much larger than lock structure and shared communications area)
- Then: coupling facility control code (like OS/390) operated in 31-bit addressing mode
- Max size of a group buffer pool was 10 GB (2 GB for directory entries, 8 GB for data entries)

# Now: coupling facility structure sizes

- Starting with coupling facility control code level 12 (current level is 17) CFCC had 64-bit addressing capability
- 64 bits enables addressing of exabytes of memory, but the size of a coupling facility structure is limited to 99,999,999 KB (just under 100 GB)
  - This is a limit of the coupling facility resource manager (CFRM), through which CF structures are defined
  - Still, that's way bigger than before, and way bigger than any coupling facility structure I've seen

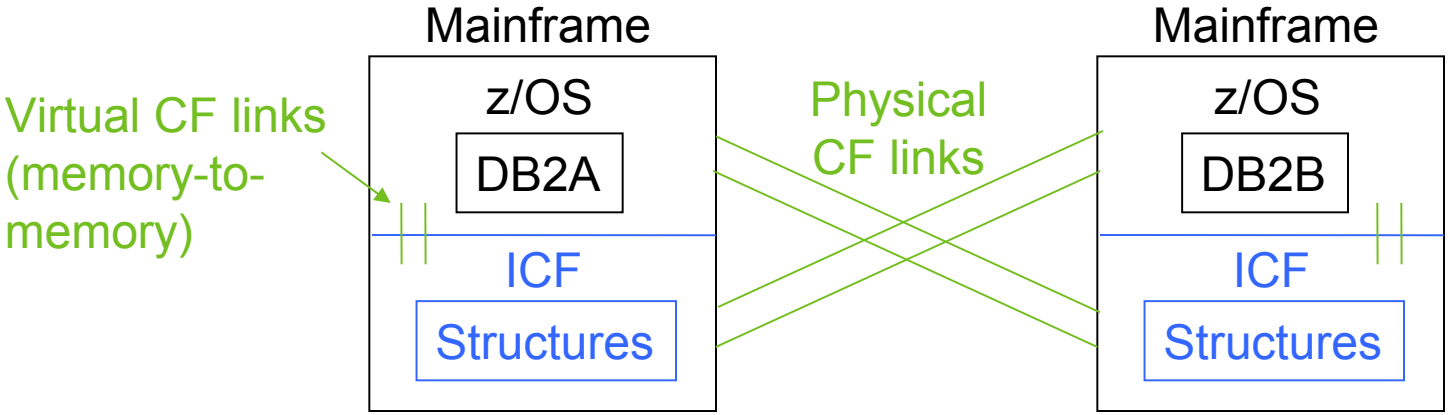
# Then: external coupling facilities

- Physically separate boxes that ran only coupling facility control code (CFCC)
  - Initially, that was your only choice
  - Same microprocessors as found in the mainframe servers
  - Attached to the mainframe servers via coupling facility links



# Now: internal coupling facilities

- Just another LPAR on a mainframe in the Parallel Sysplex
  - Recall that external CFs used regular System z microprocessors
  - Even on external CF, coupling facility control code ran in LPAR mode
- Primary motivation: cost (less expensive than external CF)
  - Secondary benefit: memory-to-memory data transfer with z/OS LPAR on same mainframe box reduces service times, boosts performance

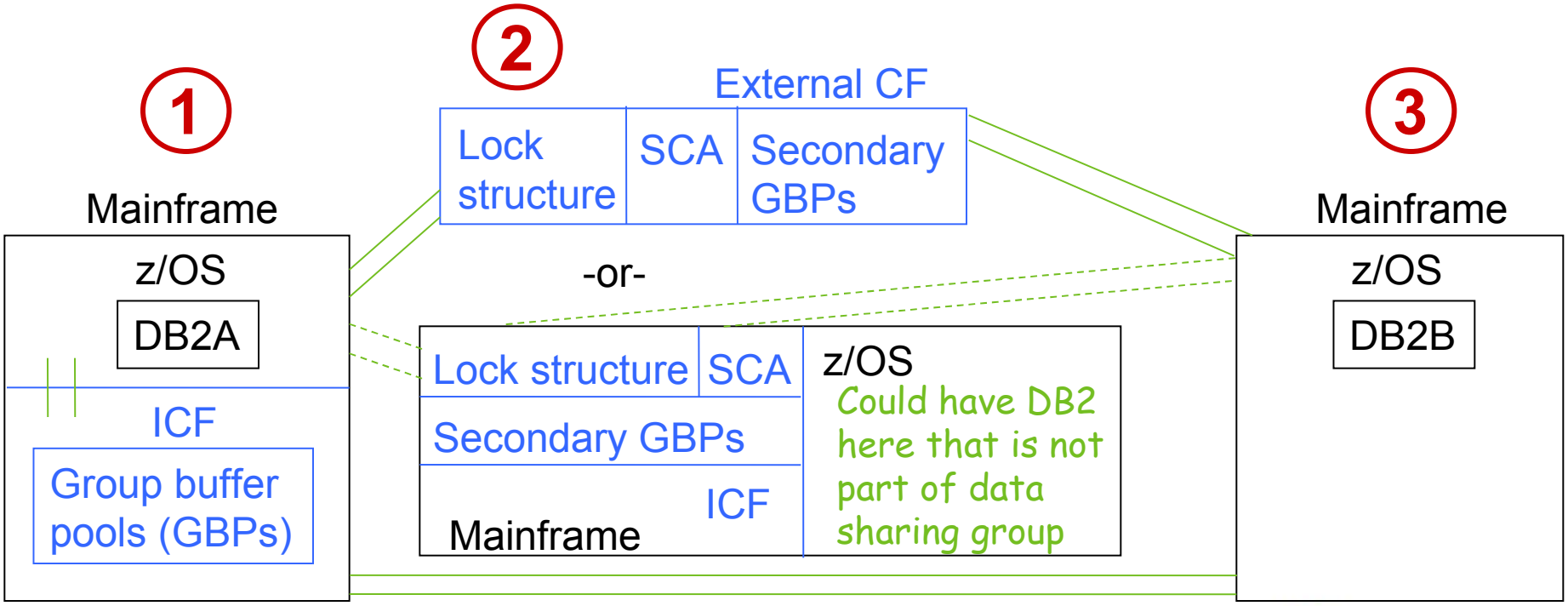


# ICF issue: “double failure” scenario

- If members lose connectivity to lock structure or shared communications area (SCA), structure has to be rebuilt
- Successful rebuild requires information from all members of the data sharing group
- If one mainframe box has an ICF with the lock structure and SCA, and also has a z/OS LPAR with a DB2 member that uses those structures, and that box goes down…
  - You’ve simultaneously lost lock structure and SCA and a member of the associated DB2 data sharing group
  - Information from the failed DB2 member that is needed for lock structure / SCA rebuild is not available, so rebuild fails
  - Because data sharing requires lock structure and SCA, the group fails

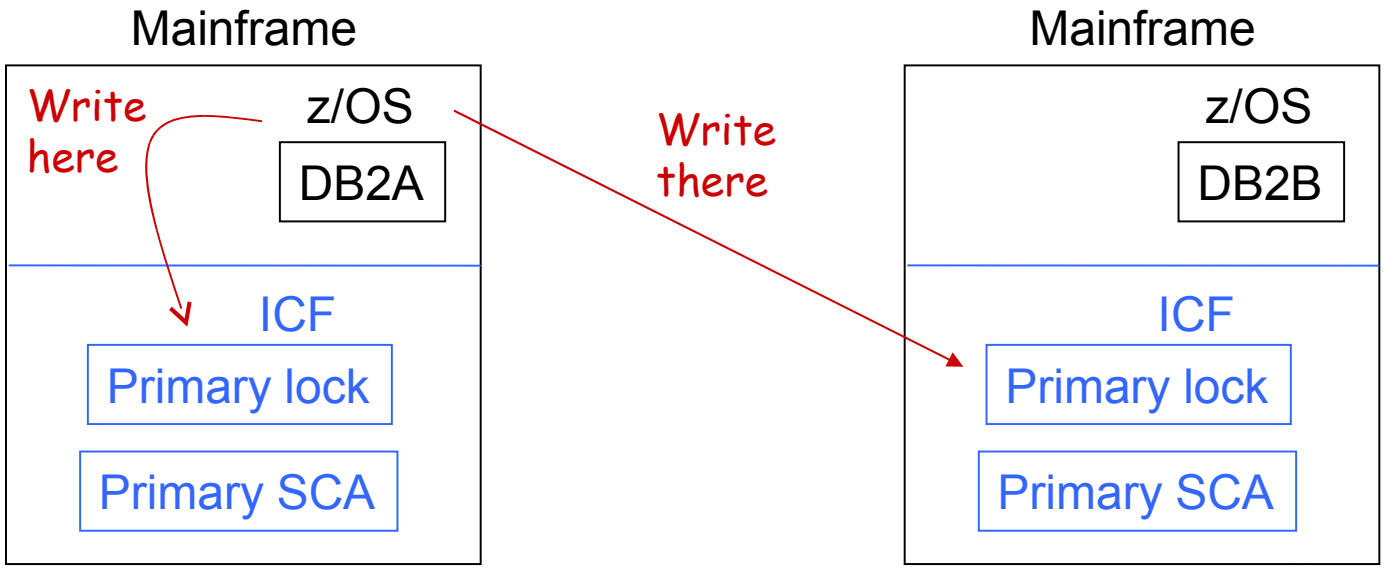
# Then: what to do about double failure

- One option: have at least 3 physical server boxes
  - Put lock structure and SCA (and secondary GBPs) in external CF, or in an ICF on a mainframe on which you DO NOT run a member of the associated data sharing group



# Then: what to do about double failure

- Another option:
  - Duplex the lock structure and the SCA
    - Every write to either one goes synchronously to both primary and secondary structures in two different CFs



# Now: what to do about double failure

- Increasingly, organizations are not worrying about it
- These organizations don't want to pay the cost of double failure protection
  - Don't want to pay the cost of an external CF, may not have an "extra" mainframe in which they can put an ICF and in which they do not run a member of the DB2 data sharing group
  - Don't want to pay cost of lock structure and SCA duplexing (higher CPU overhead for DB2 data sharing, due to many more synchronous CF requests and much higher service times)
    - Average service time for lock structure requests can be 2 to 4 times higher when lock structure is duplexed versus not duplexed

# Not worrying about double failure

- One reason cost of double failure protection looks high to many organizations: risk of scenario occurring is exceedingly low
  - You'd have to lose entire mainframe box (not "just" a DB2 or z/OS or ICF), and it would have to be a particular box (the one with the ICF holding the lock structure and SCA)
- And, if the double failure scenario actually occurs?
  - Data sharing group fails, and you initiate a group restart that should complete in minutes
    - Assuming no data sets in group buffer pool recover pending status (GRECP), group restart should take about 50% longer than restart of a single DB2 member (should be no GRECP if duplexing GBPs)
  - No loss of committed DB2 data changes

# As for group buffer pool duplexing…

- KEEP DOING THAT!
- CPU overhead cost of GBP duplexing is MUCH lower than that of lock structure / SCA duplexing:
  - Volume of requests to secondary GBPs is far lower than volume of requests directed to primary GBPs (only changed pages – no page registration activity)
  - Requests to secondary GBPs are asynchronous (so mainframe engine driving a request doesn't "dwell" until CF responds)
- Benefit of GBP duplexing is significant
  - Without it, if members lose connectivity to GBPs, you could have thousands of DB2 data sets in GRECP status
    - Getting those data sets out of GRECP could take a while

# Then: client/server configuration

- DRDA requesters connected to a DB2 data sharing group as a whole or to a single member subsystem
- Some people wanted DRDA clients to be able to communicate with a subset of a group's members
  - More than one, for better availability (if DB2A and DB2B are in subset, connection successful if at least one is active)
  - Not all, because of desire to keep DDF traffic (for certain applications, at least) off of some members
    - Some DDF applications are very dynamic, and some group members may be tuned "just so" for high-volume OLTP and/or batch work
- With DB2 lacking this capability, people had to make the "one or all" DDF choice (if they didn't have the option of setting ZPARM parameter MAXDBAT to 0 for a member)

# Now: client/server configuration

- DB2 V8: data sharing member subsetting (location aliases)
  - A way to define a new location name, for an existing data sharing group, that maps to a subset of the group's members
  - When DRDA requesters use that location alias, only members in the associated subset process the requests
  - Implementation: update BSDS using DSNJU003 (change log inventory)
    - On member DBP1 of group LOCDBGP:

```
DDF LOCATION=LOCDBGP,PORT=1237,RESPORT=1238,ALIAS=DBPA:8002
```

    - On member DBP2 of group LOCDBGP:

```
DDF LOCATION=LOCDBGP,PORT=1237,RESPORT=1239,ALIAS=DBPA:8002
```

    - -DISPLAY DDF on member DBP1 (portion of output):
    - DSNL087I ALIAS PORT
    - DSNL088I DBPA 8002

Or, use –MODIFY DDF command in DB2 10 environment!

# Data sharing locking – then and now

# Then and now: biggest change

- Locking protocol 2
- First, a little background:
  - What we just call “locks” in a standalone DB2 environment are called logical locks (or L-locks) in a data sharing system
  - L-locks are divided into two categories:
    - Parent (generally speaking, table space- or partition-level locks)
    - Child (these are page- or row-level locks)
  - The most common parent L-locks are intent locks (IX for data-changing processes, and IS for read-only processes)
    - The system lock manager (a z/OS component that handles global locking) knows two lock state: S and X
    - How do IS and IX parent L-locks get propagated to the lock structure?

# Then: parent L-lock propagation

| Parent L-lock type | Propagated to lock structure as: |
|---|---|
| X | X |
| IS | S |
| IX | X |
| S | S |

Sources of XES contention

- Problem: if process on DB2A has IS lock on table space XYZ, and process on DB2B has IX lock on same table space, system lock manager will think that there is global lock contention
  - There isn't, because actual lock states are IS and IX, and these states are compatible; however, system lock manager knows only S and X
  - This type of "false positive" global lock contention is called XES contention – it gets resolved (with IRLM's help), but it drives up CPU cost of data sharing

# Now: parent L-lock propagation

| Parent L-lock type | Propagated to lock structure as: |
|---|---|
| X | X |
| IS | S |
| IX | S |
| S | X |

Sources of XES contention

- Thanks to the change implemented via locking protocol 2 (DB2 V8 NFM), IX table space lock will be propagated as S lock, and S table space lock will be propagated as an X lock
  - No more perceived contention when processes on two different DB2 members have IS and IX (or IX and IX) locks on same table space
    - When the two lock states are IS and S (or S and S), “false positive” will still occur, but S locks on table spaces are quite rare, so overall effect of locking protocol 2 tends to be a significant reduction in data sharing lock contention

# Then: row-level locking

- There used to be a widely-held belief that row-level locking could not be used in a DB2 data sharing environment
  - Concern had to do with effect on CPU cost of data sharing

# Now: row-level locking

- Unfortunately, there is still a widely-held belief that row-level locking cannot be used in a DB2 data sharing environment
  - This is NOT true
  - Use of row-level locking does increase the volume of what are called page physical locks (or page P-locks)
    - You'll have page P-lock activity anyway, for space map pages and index pages (pages that are not L-locked) – row-level locking adds to this activity
  - A small increase in data sharing overhead is probably preferable to a lot of lock timeouts and deadlocks
- My advice: in a data-sharing system, use row-level locking where you need it, but only where you need it
  - Probably need it for just a few table spaces, if you need it at all

# Data sharing performance – then and now

# Then: higher data sharing overhead

- Largely determined by volume of coupling facility requests and average service time for those requests
  - Especially synchronous requests, as mainframe engine driving such a request will “dwell” until receiving response from CF
    - Most group buffer pool requests, and almost all lock structure requests, are synchronous (volume of requests to SCA usually low)
  - In an environment characterized by a high degree of “inter-DB2 write/write interest” (meaning, concurrent data-change activity on multiple members targeting common database objects):
    - Overhead of DB2 data sharing expected to be between 10 and 20%
    - Meaning: increase in CPU cost of executing an SQL statement in a data sharing system as compared to the cost of executing the same statement in a standalone DB2 environment

# Now: lower data sharing overhead

- Overhead in a high inter-DB2 write/write interest environment generally around 10%
- Reasons for reduced overhead:
  - Fewer coupling facility requests (various DB2, z/OS, and CFCC enhancements, such as improved efficiency of index page split processing)
  - MUCH faster servicing of synchronous CF requests:
    - Late 1990s: < 150 microseconds for lock structure, < 250 microseconds for GBP
    - Now: 9 microseconds for lock structure, 20 microseconds for GBP
  - Actually HAD to get these requests serviced faster, because faster mainframe engines meant increased cost of “dwell” time

# What hasn't changed

- For my money, DB2 for z/OS data sharing on the Parallel Sysplex mainframe cluster is the most highly available, highly scalable data-serving platform on the market
  - Proven over 16 years in ultra-demanding application environments, across industries, all over the world
  - No longer "exotic" technology – if you aren't using DB2 data sharing at your organization, perhaps you should

# Thanks for your time!

Robert Catterall
rfcatter@us.ibm.com