Introduction to Coupling Facility Requests and Structure (for Performance)

z/OS Performance
Education, Software, and
Managed Service Providers

Instructor: Peter Enrico

Email: Peter.Enrico@EPStrategies.com

Enterprise Performance Strategies, Inc.
3457-53rd Avenue North, #145
Bradenton, FL 34210
http://www.epstrategies.com
http://www.pivotor.com

Voice: 813-435-2297
Mobile: 941-685-6789

Current 2011 Class Schedule

- WLM Performance and Re-evaluating of Goals
  - Instructor: Peter Enrico
  - June 6 - 10, 2011 Columbus, Ohio USA
  - September 12 – 16, 2011 Baltimore, Maryland, USA
- Essential z/OS Performance Tuning
  - Instructor: Peter Enrico and Tom Beretvas
  - May 9 - 13, 2011 St. Louis, Missouri, USA
- Parallel Sysplex and z/OS Performance Tuning
  - Instructor: Peter Enrico
  - May 16 - 20, 2011 Omaha, Nebraska USA
  - September 19 – 23, 2011 Dallas, Texas, USA
- z/OS Capacity Planning and Performance Analysis
  - Instructor: Ray Wicks
  - August 15 – 17, 2011 Columbus, Ohio, USA

Peter Enrico Speaking Schedule

- Sysplex: Introduction of Coupling Facility Requests and Structures for Performance
  - Tuesday, March 1, 2011: 11:00 AM-12:00 PM
- Sysplex: Key Coupling Facility Measurements - Cache Structures
  - Tuesday, March 1, 2011: 1:30 PM-2:30 PM
- Exploring the SMF 113 Processor Cache Counters and LSPRs
  - Thursday, March 3, 2011: 9:30 AM-10:30 AM
- z/OS Ask the Experts Panel & MVS Program Closing
  Thursday, March 3, 2011: 6:00 PM-7:00 PM

Contact, Copyright, and Trademark Notices

Questions?

Send email to Peter at Peter.Enrico@EPStrategies.com, or visit our website at http://www.epstrategies.com.

Copyright Notice:

Trademarks:

Abstract

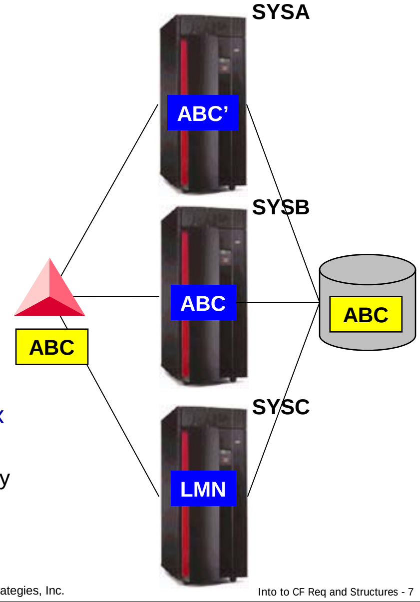
- **Abstract**
  - During this performance presentation Peter Enrico will provide an overview to the Coupling Facility request types and List, Lock, and Cache structures. Emphasis will be placed on the performance sensitive points for each.

Section Overview

- Some information on couple datasets
- Coupling facility request types
  - Asynchronous requests
  - Synchronous requests
- Coupling facility structures
  - List structures
    - Un-serialized
    - Serialized
  - Lock structures
  - Cache structures
- How data sharing works with lock structures
  - What is data sharing
  - Data sharing logic using the coupling facility

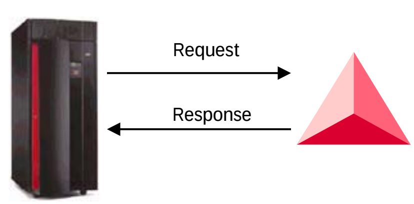
Data Sharing - Parallel Sysplex Model

- Database is not partitioned
  - All systems have read/write access to all data
- Data is accessed directly, but coupling facility is used to communicate and coordinate any updates
  - Transaction and Database management software needed as well
- Response time and throughput of data access near linear
- Availability designed into parallel Sysplex
  - Workload balancing and recovery built in
  - Can easily be configured with redundancy for recovery purposes



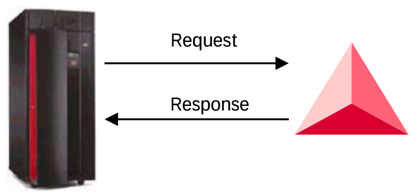
Coupling Facility Request Types

- Performance is heavily dependant on a number of variables:
  - Speed of requesting CPU
    - Larger processor will 'waits faster' for a response
  - Busy conditions (Subchannel, path)
  - Time it takes to transmit data to the CF
    - CF link performance
    - Speed of data over link
    - Distance – Geographically Dispersed Parallel Sysplex?
  - Speed of CF processor
    - Shared LPAR versus dedicated LPAR
    - Shared CF versus Dedicated CF



Coupling Facility Request Types

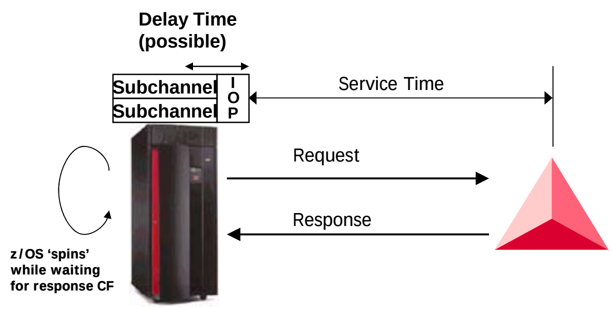
- Requests sent to a CF may encounter busy conditions
  - Subchannel busy
  - Path busy
- If busy condition is encountered
  - Some can be queued
  - Other request must complete synchronously (i.e. lock request)
- CF accesses can be classified into two categories
  - Synchronous requests
  - Asynchronous requests



CF Synchronous Request Processing

- Requesting processor spins waiting for CF request to complete
- Two types of sync requests
  - Those that must continuously run as synchronous
    - Lock requests - XES spins
  - Those that start out as sync
    - But converted to async if doing so helps performance
    - Sync cache/list requests - XES changes to async

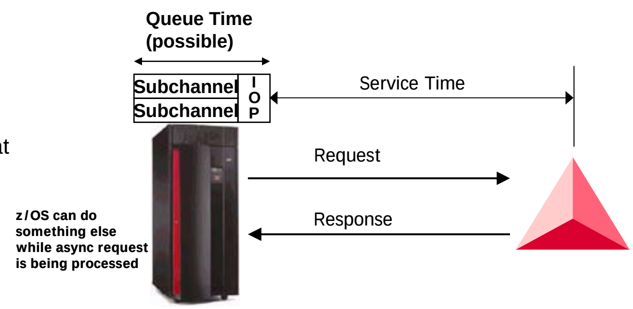
Impacts response time of sender, and performance of sending system



CF Asynchronous Request Processing

- Requesting processor can do other processing while the CF request is queued
  - Immediate reply is not required
    - Async cache/list requests - XES queues
  - Serialized access is not required
  - Some async requests may have started out as sync requests
    - Sync cache/list requests - XES changes to async

Performance impact much less severe that sync requests



Best Sync and Async Service Times Expected

- Speed of sending processor and CF factor in greatly into service times
  - Note: All times in microseconds

	9672-R64 to C04	2064-114 to 1xx	2084-3xx to 3xx	2094-109 to 2xx
Sync Request	100 - 175	15 - 30	10 - 25	7 - 20
Async Request	500 - 1500	150 - 450	100 - 350	80 - 300

- The ranges are service times take into account the transfer of data
  - Low end of ranges – when there is no transfer of data (such as a GRSLOCK request)
  - High end of ranges – when there is the largest transfer of data allowed
    - Sync request – 4K is largest transfer of data allowed
    - Async request – 64K is largest transfer of data allowed
  - Ranges also assume the fast CF link technology is available on the processor and that the Sysplex is well tuned

Example: Host CPU Seconds Due to Synch Spins

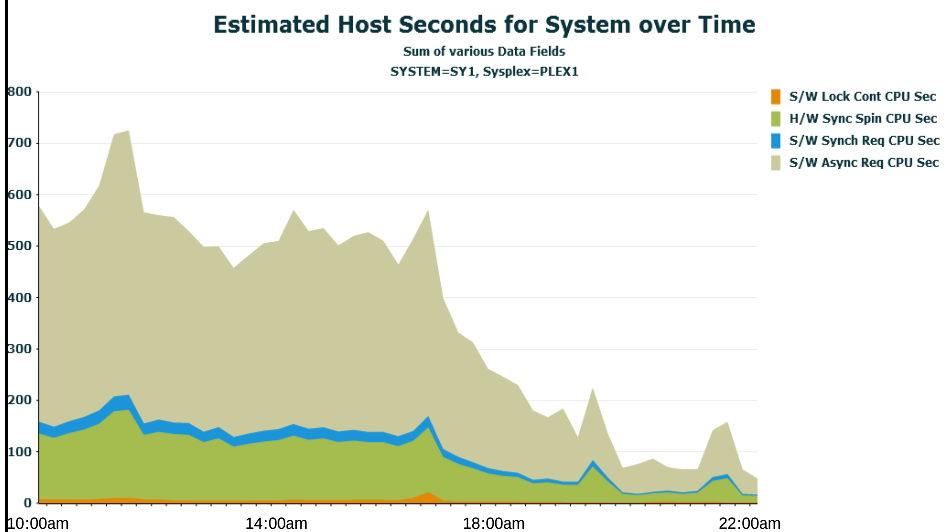
- CPU seconds consumed due to sync immediate Spin
  - Sync Immediate requests cause processor issuing the request to 'spin'
  - How many CPU seconds did the sending LPAR spend spinning?
    - Logical processor unavailable to other work running in the same LPAR
    - Physical processor that logical processor is dispatched to is unavailable to other LPARs

$$\text{CPU Seconds Spinning} = \frac{(\#\,\text{REQ Sync})*(\text{Sync Service Time})}{1{,}000{,}000}$$

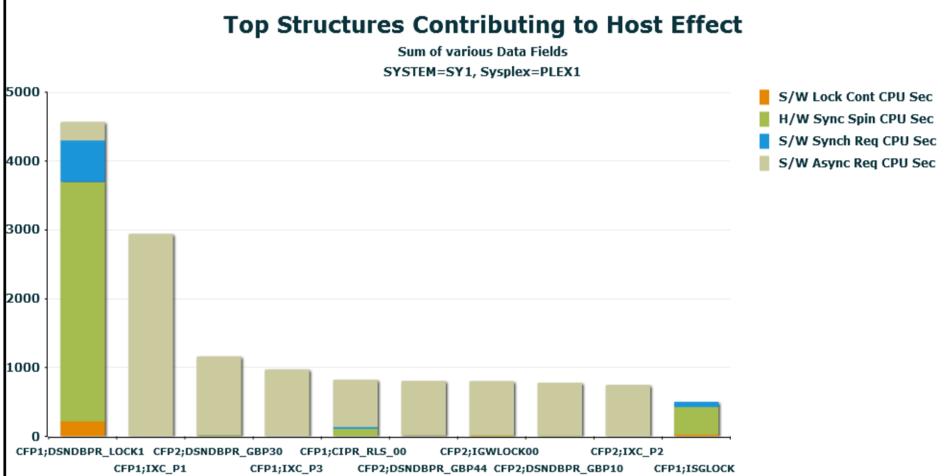
  - Note: Below example: ((9,047,000 sync req) * (14.6 mics per req)) / 1,000,000) = 132.1 CPU spin Seconds

SYSTEM NAME	# REQ TOTAL AVG/SEC	-- CF LINKS -- TYPE	GEN	USE	PTH BUSY	----------- REQUESTS -----------	# REQ	-SERVICE TIME(MIC)- AVG	STD_DEV	------------------ DELAYED REQUESTS ----	# REQ	% OF REQ	------ AVG TIME(M /DEL	STD_DEV
SYSD	29203K	CBP	2	2	0	SYNC	9047K	14.6	10.9	LIST/CACHE	9714	0.0	574.3	5937
	16224	SUBCH	14	14		ASYNC	20201K	47.4	215.1	LOCK	587	0.0	9.9	19.0
						CHANGED	6321	INCLUDED IN ASYNC		TOTAL	10K	0.0		
						UNSUCC	0	0.0	0.0					

Example: Host Effect Example – Over Time (assuming 701)



Example: Host Effect Example – 12 Hours



Sync CF Request Heuristic Algorithm

- ‘self learning’ function available
- Fact: Long running Sync CF requests use more CPU on the sender
  - During sync requests, the requesting z/OS system / LPAR spin waiting for the CF response
- CF response time for sync requests are monitored
  - List, lock, and cache requests times compared to threshold value
  - All long (only long) requests are converted to async
    - No mater the reason
  - Threshold used determined by certain factors
    - Whether a simplex or duplex environment
    - Lock and non-lock requests are normalized by processor types

Heuristic Algorithm cont…

- CF sync request heuristic algorithm attempts to limit the impact of
  - Long link distances
    - Short link distances (such as 5 meters) will see little affect based on distance impact
    - Longer link distances (such as 10km) will see reductions in costs of data sharing
  - Technology mismatches
    - Such as an old/slow CF attached to a new/fast z/OS processor
  - High CF utilizations
- With long links, most sync requests are converted to async requests
  - In the RMF Coupling Facility reports / measurements
    - Requests are counted as async requests
    - But not counted as changed requests
- Since it is heuristic
  - The decision threshold is continuously reevaluated by allow every nth sync request to be issued as unchanged and then z/OS compares the response time of request with the current threshold values
    - So in coupling facility measurements you will still see sync elongated requests

When Sync Requests Converted to Async

- Some delayed sync requests can be converted to async

  Conversion logic is as follows:

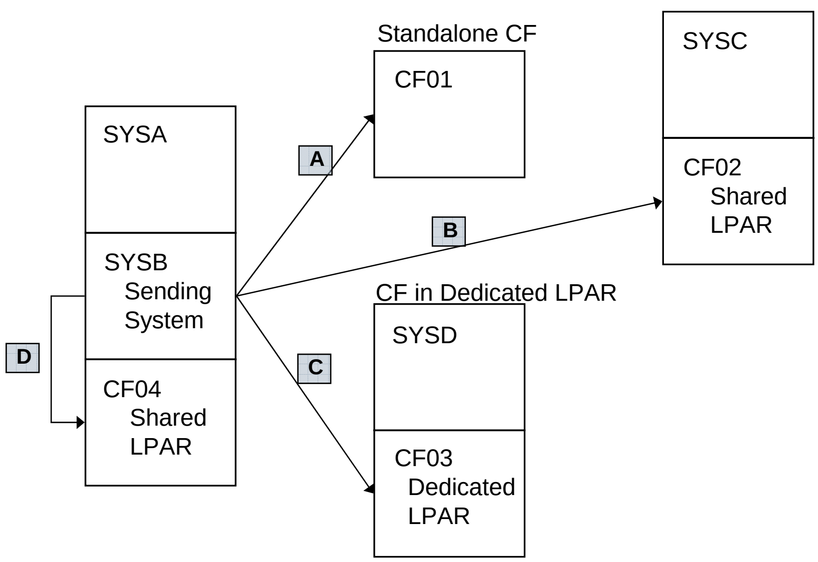
  A. Receiving CF is a standalone CF
     - Sync request is converted via heuristic conversion algorithm

  B. Receiving CF is shared LPAR on different CEC
     - Sync request is converted via heuristic conversion algorithm

  C. Receiving CF is dedicated LPAR or ICF
     - Sync request is converted via heuristic conversion algorithm

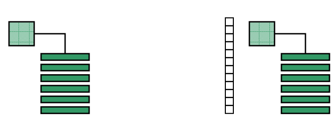
  D. Receiving CF is shared LPAR on same CEC
     - Sync request is converted via heuristic conversion algorithm

Examples of Sync Requests Converted to Async

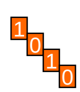


Introduction to Coupling Facility Structures

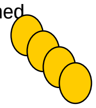
- Data is organized in the coupling facility in one of four different structure types
  - List structures – Simple (un-serialized) and Serialized
    - When data needs to be organized in to lists, queues, stacks



  - Lock structures
    - When serialization is required



  - Cache structures
    - When data needs to cached



Multi-System Management Oriented Exploiters

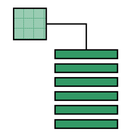
- Enhanced Catalog Sharing (ECS)
- XCF signaling
  - So anything using XCF messaging
- JES2 Checkpoint data set
- Operlog – Shared operations log stream
- Logrec – Shared Logrec log stream
- RACF
- GRS Star topology
- VTAM
  - Generic Resources
  - MNPS - Multi-Node Persistent Sessions
- RRS – shared log stream
  - Archive
  - Restart
  - Resource Manager Log
  - Unit of Recovery Logs
- SmartPipes – a.k.a BatchPipes
- MQSeries
- Intelligent Resource Director (IRD)
- WLM multi-system enclaves
- DFSMShsm Common Recall Queue
- TCP/IP system-wide security associations
- TCP/IP Sysplexports
- OEM
  - OEM – Other Equipment Manufacturers (i.e. Non-IBM)
  - There are also OEM products that support coupling facility functions
  - Support for list, lock, and cache

Data Sharing Structures Oriented Exploiters

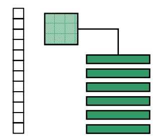
- CICS
  - Shared primary and second system logs
  - Shared journals
  - Forward recovery logs
  - Temporary Storage Queue Pool
  - Named Counter Server
- DB2
  - IRLM lock manager
  - Shared Communication Area (SCA)
  - Buffer Pools
- VSAM RLS
  - Cross system contention handler (locking)
  - System buffer Pools
- IMS
  - IRLM lock manager
  - Shared IMS log
    - Forward Recovery logs
  - Shared message queues
  - IMS VSO DEDB (?) caches
  - OSAM – (directory only cache structure)
  - OSAM – (store through cache structure)
  - VSAM – (directory only cache structure)

List Structures

- CF can be used to store data organized into list structures
- List structure made up of
  - List entries
  - List elements
  - Optional lock table
- List structures can be organized into
  - FIFO queues
  - Push / Pop structures
  - Static lists
- Uses for list structures include
  - high speed message routing
  - distributing work requests among Sysplex members (as in shared work queues)
  - Maintain shared information such as status



Simple List Structure

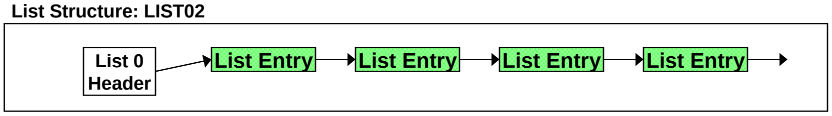


Serialized List Structure

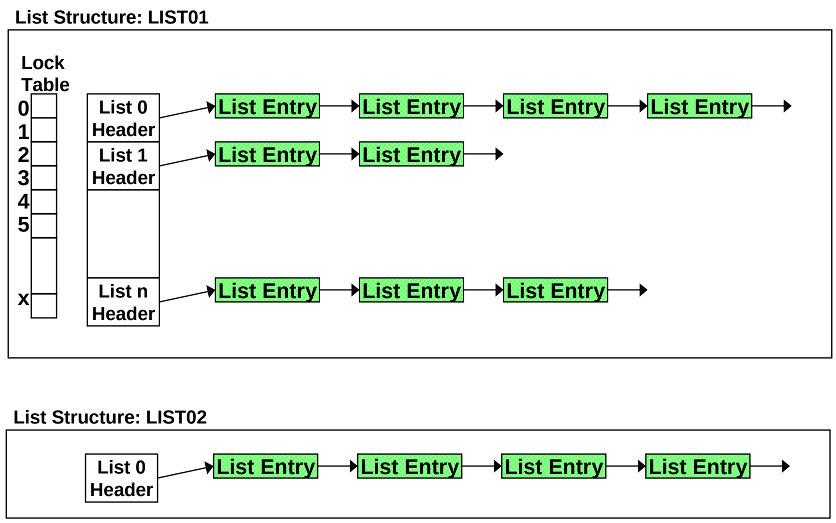
List Structure Components

- The primary components of a list structure include:
  - List header
  - List entry
    - Control information
    - Data
  - Lock table (optional)
    - Not to be confused with a lock structure
    - Could be used to serialize access to list entries
    - (See next foil for example)

List Structure: LIST02



Serialized List Structure Components

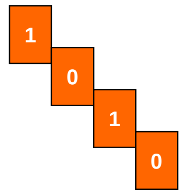


List Structure Exploiters

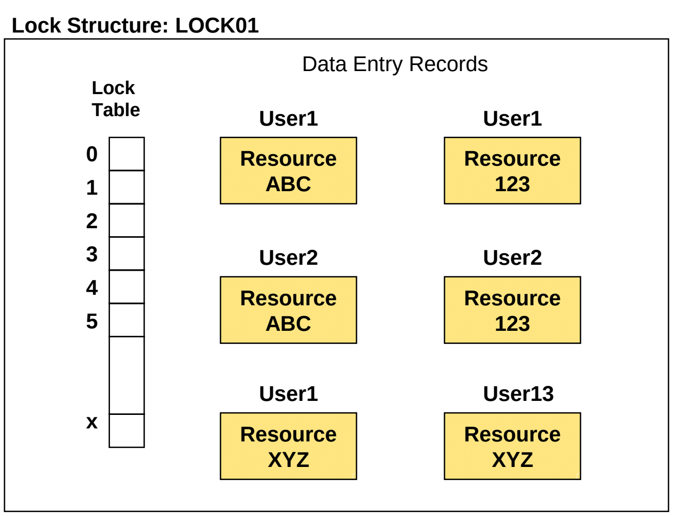
- XCF signaling
- JES2 Checkpoint data set
- Operlog – Shared operations log stream
- Logrec – Shared Logrec log stream
- VTAM
  - Generic Resources
  - MNPS - Multi-Node Persistent Sessions
- RRS – shared log stream
- SmartPipes – a.k.a BatchPipes
- MQSeries
- Intelligent Resource Director (IRD)
- WLM multi-system enclaves
- DFSMShsm Common Recall Queue
- TCP/IP
  - system-wide security associations
  - TCP/IP Sysplexports
- CICS
  - Shared primary and second system logs
  - Shared journals
  - Forward recovery logs
  - Temporary Storage Queue Pool
  - Named Counter Server
- DB2
  - Shared Communication Area (SCA)
- IMS
  - Shared IMS log
    - Forward Recovery logs
  - Shared message queues

Lock Structures

- CF can be used as a high speed locking facility by using lock structures
  - Lock structures are centralized lock tables maintained in the CF
- Lock structure made up of
  - Lock table containing information about the serialized resource
  - Lock record containing information about connected users
- Lock structures support
  - Shared lock state
  - Exclusive lock state
  - Application defined lock state
- Uses for lock structures include
  - Synchronous resource serialization
  - Resource contention detection

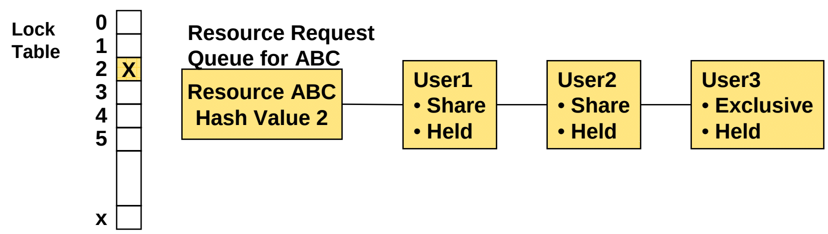


Lock Structure Components



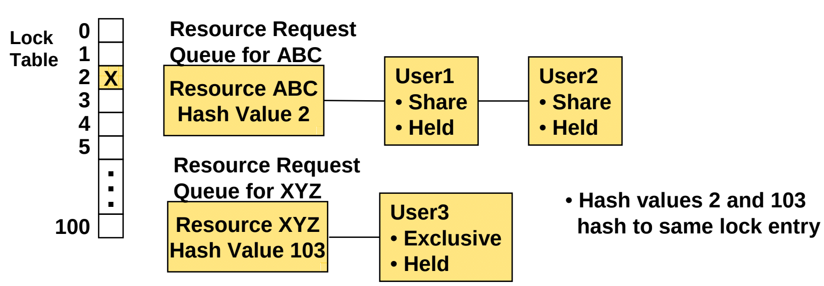
Types of Lock Contention

- Real Lock Contention
  - Contention caused by multiple units of work attempting to serialize on the same resource
  - Factors that influence real lock contention
    - How the locks are being used
    - Amount of time locks are held
    - Degree of data sharing
  - Alleviate real lock contention by tuning the workload (not by tuning the Sysplex or CF structures)



Types of Lock Contention

- False Lock Contention
  - When multiple lock names are hashed to the same lock entry
    - Results in significant excessive processing overhead to resolve
  - Factors that influence false lock contention
    - Size of lock structure
    - Granularity of locking (record, file, block)
    - Concurrent users connected to lock structure
  - Alleviate false lock contention by increasing lock structure size



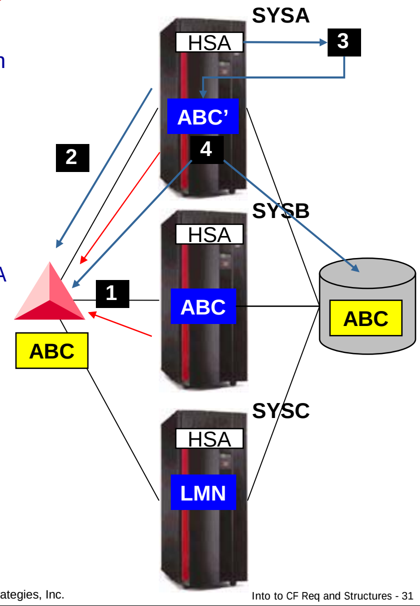
Parallel Sysplex Data Sharing – High Level

1) Both SYSA and SYSB have read record ABC into their own buffers. Both have a registered interest in the data.
   - SYSA wants to update record ABC to ABC'

2) Database manager on SYSA issues request to coupling facility to obtain an exclusive lock for update.

3) SYSA uses local cache vector table in hardware HSA to determine if record in local buffer is valid. (Assume it is valid for this example. Thus, it is not necessary to re-read in the data from disk.)

4) SYSA has lock, so it changes ABC to ABC'
   - Local cache buffer
   - CF cache structure (if store-in algorithm)
   - CF cache structure and disk (if store-through algorithm)
   - Disk only (if directory only algorithm)



Peter Enrico : www.epstrategies.com © Enterprise Performance Strategies, Inc. Into to CF Req and Structures - 31

Parallel Sysplex Data Sharing cont…

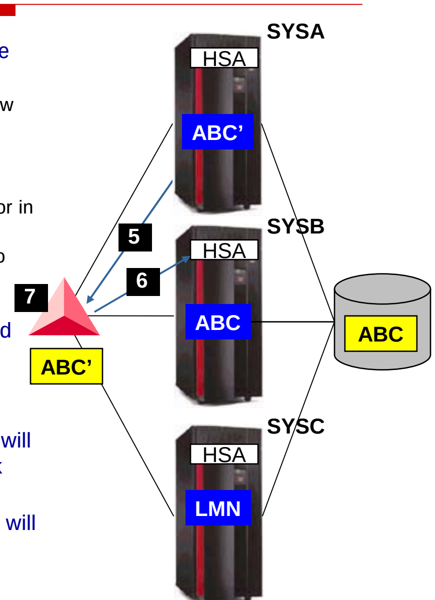
5) Signal sent by database manager to CF to indicate that record ABC has been updated
   - CF cache structure updated so all systems will know ABC has been updated

6) CF invalidates all the local buffers for ABC
   - It does this by setting a bit in the local cache vector in the HSA
   - This *cross invalidation* is done with no interrupts to other systems

7) Update is now complete and serialization of record ABC is now released
   - This is known as *lock release*

- Next time SYSB attempts to access record ABC it will know to get the fresh copy, ABC', from CF or disk

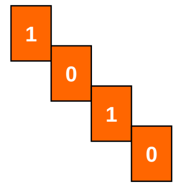
- Next time SYSA attempts to access record ABC' it will know it already has the latest copy in its buffers



Peter Enrico : www.epstrategies.com © Enterprise Performance Strategies, Inc. Into to CF Req and Structures - 32

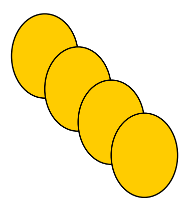
Lock Structure Exploiters

- GRS Star topology
- DB2
  - IRLM lock manager
- VSAM RLS
  - Cross system contention handler (locking)
- IMS
  - IRLM lock manager

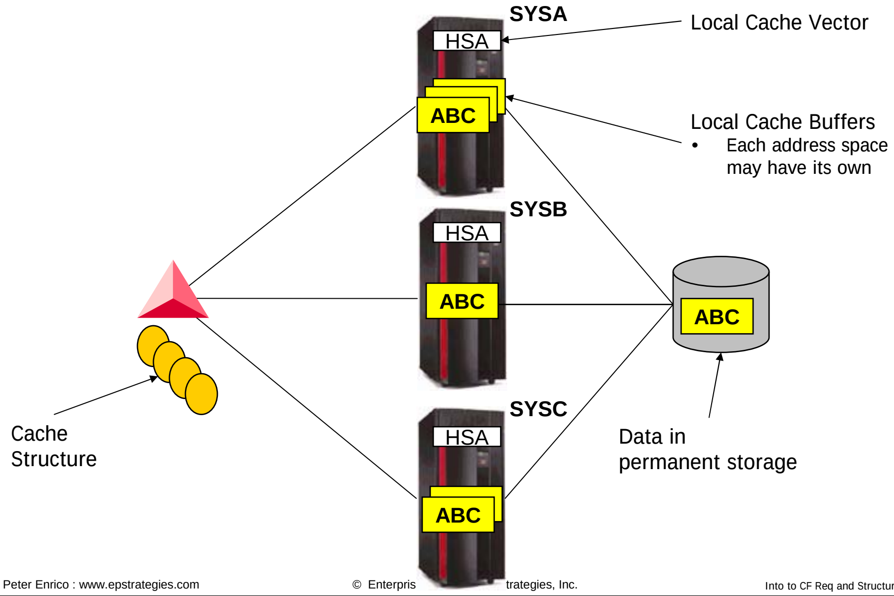


Cache Structures

- CF can be used as a high speed caching facility
- Cache structure made up of
  - directory to keep track of registered data elements
  - optionally, data elements
- Usage of cache structure
  - data consistency / buffer validation
    - ability to maintain a shared copy of data in cache structure in CF
    - ability to keep track of shared data that does not reside in CF
      - permanent storage (i.e. disk)
      - local storage (i.e. z/OS or subsystem buffers)
  - high speed data access
    - Shared data can be stored in cache structure and made available to every system in sysplex
    - Invalid local copy of data can be refreshed with CF cached copy
    - CF access faster than I/O subsystem cache



Cache Structure Components



Cache Structure Terminology

- Local Cache Buffers
  - Buffers in private area storage of cache structure exploiting subsystems
  - Required and allocated by every exploiter of cache structures
  - Contains copies of shared data
  - Populated by disk or CF cache structure
  - Used to refresh CF cache structure or disk copy
- Permanent Storage
  - Final and permanent repository for shared data - usually disk
  - Used to populate local cache buffers
- Local Cache Vectors
  - User defined vector in HSA
  - Allows connectors of a cache structure (i.e. those sharing data) to determine if their local cache buffers contain the latest copy of the data

Cache Structure Terminology

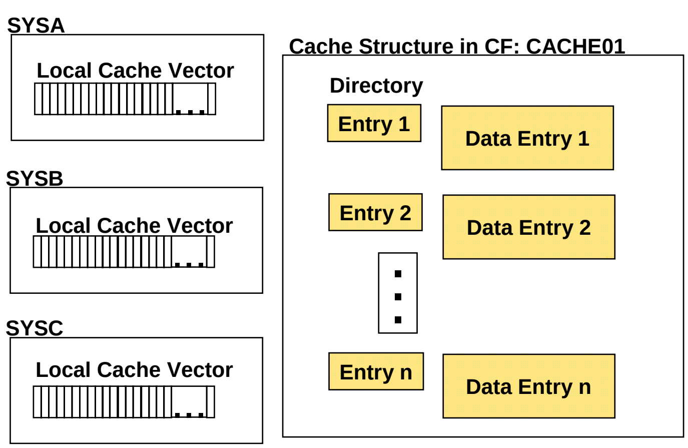
- The cache structure in the coupling facility has two primary components

- Directory Entries
  - Used to keep track of data entries that are shared among multiple systems
  - Every system that has a copy of a particular piece of shared data has a registration entry in this portion of the cache structure.
  - It is this directory whose entries are used to generate cross invalidation signals to indicate that a record in a local cache buffer may be invalid

- Data Entries
  - Used to contain a cached version of the data
  - Optional

Cache Structure Components cont...

- Directory - Used to keep track of share entries
- Data Entries - Used to optionally cache data



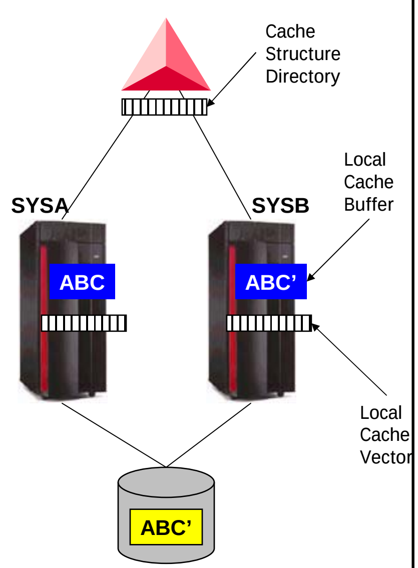
Cache Structure Usage

- There are three ways that CF cache structures are used
- Directory in Cache
  - CF structure is used to assist in maintaining consistency of data in local cache buffers
- Store Through Cache
  - Most recent copy of data is kept in both CF cache structure and DASD
- Store-In Cache
  - Most recent copy of data kept in CF cache structure and hardened to DASD asynchronously

NOTE: The way a cache structure is used and measured is based on the exploiter and how the exploiter is using making use of the cache structure.

Directory in Cache

- CF structure is used to assist in maintaining consistency of data in local cache buffers
- Writes
  - CF cache structure only contains directory entries; no data stored in CF
  - Data always written from local cache buffers to DASD
  - CF used to invalidate local other's buffers
- Reads
  - Local cache buffer version of data is used if local cache vector indicates that it is still valid
  - If local cache vector indicates that local copy is invalid, a fresh read from permanent storage is done



Directory in Cache Example Explained

1) SYSA reads in record ABC from DASD
   - Lock manager requests a shared lock from CF cache directory structure
   - Since ABC has never been read in before a registration of record entry is done in the CF cache structure directory
   - Registration of record is made in SYSA local cache vector

2) SYSB reads in record ABC
   - Lock manager requests a shared lock from CF cache directory structure
   - SYSB added to list of interested systems in CF cache directory entry for ABC
   - Registration of record is made in SYSB local cache vector
   - Data read in from DASD

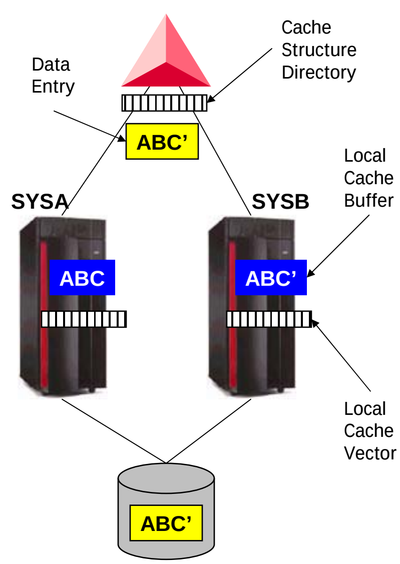
3) SYSA wants to update ABC to ABC'
   - SYSA granted exclusive lock to in CF directory entry for ABC
   - SYSB unaware of any activity
   - Local cache buffers in SYSA updated to ABC'
   - DASD updated to ABC'
   - SYSB's local cache vector is marks entry for ABC to indicate local in buffer copy invalid

4) SYSB wants to use record ABC (after the update)
   - SYSB sees that local cache vector entry for ABC indicates that in-storage copy is invalid so fresh version of record ABC' is read in from permanent storage

Peter Enrico : www.epstrategies.com © Enterprise Performance Strategies, Inc. Into to CF Req and Structures - 41

Store Through Cache

- Most recent copy of data is kept in both CF cache structure and permanent storage
- Writes
  - CF cache structure contains directory entries
  - CF cache structure contain both changed and unchanged data
  - Data written to CF and DASD at the same time (serialized)
  - No cast out processing
- Reads
  - Local cache buffer version of data is used if local cache vector indicates that it is still valid
  - If local cache vector indicates that local copy is invalid, a fresh read from CF



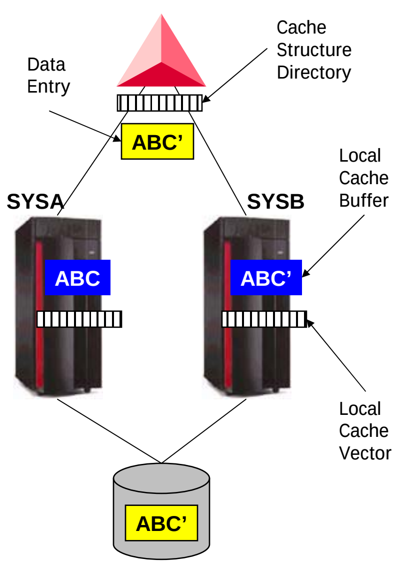
Peter Enrico : www.epstrategies.com © Enterprise Performance Strategies, Inc. Into to CF Req and Structures - 42

Store Through Cache Example Explained

1) SYSA reads in record ABC from DASD
   - Lock manager requests a shared lock from CF cache directory structure
   - Since ABC has never been read in before a registration of record entry is done in the CF cache structure directory
   - Copy of ABC read into cache structure data entry
   - Registration of record is made in SYSA local cache vector
2) SYSB reads in record ABC
   - Lock manager requests a shared lock from CF cache directory structure
   - SYSB added to list of interested systems in CF cache directory entry for ABC
   - Data read into SYSB from CF cache structure
   - Registration of record is made in SYSB local cache vector
3) SYSA wants to update ABC to ABC'
   - SYSA granted exclusive lock to in CF directory entry for ABC
   - SYSB unaware of any activity
   - Local cache buffers in SYSA updated to ABC'
   - CF data entry and DASD updated to ABC'
   - SYSB's local cache vector is marks entry for ABC to indicate local in buffer copy invalid
4) SYSB wants to use record ABC (after the update)
   - SYSB sees that local cache vector entry for ABC indicates that in-storage copy is invalid so fresh version of record ABC' is read in from CF.

Store-In Cache

- Most recent copy of data kept in CF cache structure and hardened to DASD later
- Writes
  - CF cache structure contains directory entries
  - Changed data stored in CF
  - Updated data written to DASD later by subsystem. Known as cast out processing.
  - Data in CF may be different than data on DASD
- Reads
  - Local cache buffer version of data is used if local cache vector indicates that it is still valid
  - If local cache vector indicates that local copy is invalid, a fresh read from CF
  - DASD is check as last resort



Store-In Cache Example Explained

1) SYSA reads in record ABC from DASD
   - Lock manager requests a shared lock from CF cache directory structure
   - Since ABC has never been read in before a registration of record entry is done in the CF cache structure directory
   - Copy of ABC read into cache structure data entry
   - Registration of record is made in SYSA local cache vector

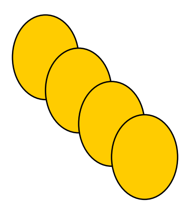
2) SYSB reads in record ABC
   - Lock manager requests a shared lock from CF cache directory structure
   - SYSB added to list of interested systems in CF cache directory entry for ABC
   - Data read into SYSB from CF cache structure
   - Registration of record is made in SYSB local cache vector

3) SYSA wants to update ABC to ABC'
   - SYSA granted exclusive lock to in CF directory entry for ABC
   - SYSB unaware of any activity
   - Local cache buffers in SYSA updated to ABC'
   - CF data entry updated to ABC'; DASD updated at a later time by subsystem
   - SYSB's local cache vector is marks entry for ABC to indicate local in buffer copy invalid

4) SYSB wants to use record ABC (after the update)
   - SYSB sees that local cache vector entry for ABC indicates that in-storage copy is invalid so fresh version of record ABC' is read in from CF.

Cache Structure Exploiters

- Enhanced Catalog Sharing (ECS)
  - store in cache structure
- RACF
  - store through cache structure
  - For frequently accessed data
- DB2
  - Buffer Pools – store in cache structure
- VSAM RLS
  - System buffer Pools – store through cache structure
- IMS
  - IMS VSO DEDB – store in cache structure
  - OSAM – directory only cache structure
  - OSAM – store through cache structure
  - VSAM – directory only cache structure



Duplexed Structures

- Structures can be duplexed
- Two types of duplexing
  - User Managed Structure Duplexing
    - User is defined as the exploiter of the CF (such as DB2 and IMS)
  - System Managed Structure Duplexing
    - The customer (system programmer) defines and the system will manage
- Usually customers do not bother to duplex if
  - The Sysplex is configured with 2 standalone coupling facilities
  - Or the Sysplex is configured with 1 standalone and one ICF
    - But depends upon structure placement
- It is highly recommended to duplex when the Sysplex is configured with 2 ICFs
  - Since more likely to lose both a z/OS and CF during a CEC failure

User Managed Structure Duplexing

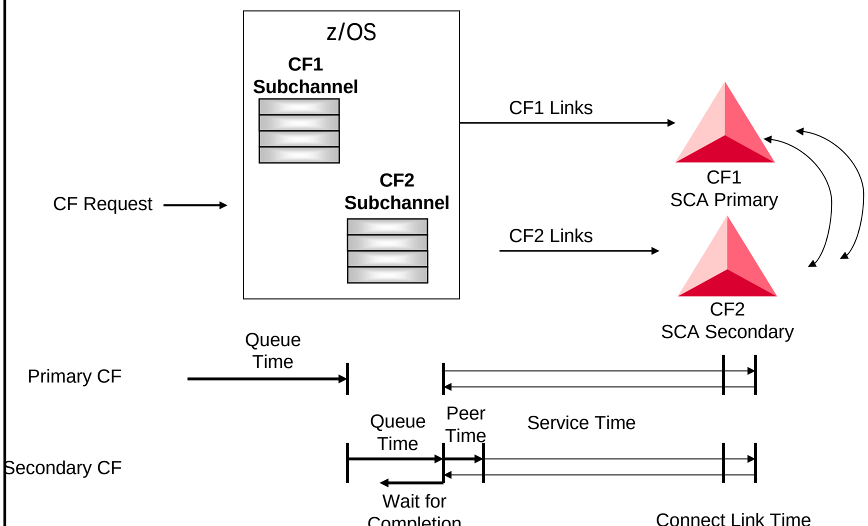
- User Managed Structure Duplexing
  - User is defined as the exploiter of the CF (such as DB2 and IMS)
- Exploiters include
  - IMS Shared VSO
  - DB2 Group Buffer Pools
- Automatically done
  - Example: DB2 will turn on when it sees attributes on the structure
- Costs more, but since most requests are reads and not writes, the cost is kept down since a read only has to be from one CF
- Example:
  - DB2 says it wants duplexing and the structure is then allocated as duplexed
  - DB2 is in charge, so DB2 will issue a command twice
  - In regards to Group Buffer Pools, since 2 DB2s could be issuing a write, a special DB2 lock is maintained to keep the GBPs in sync

System Managed Structure Duplexing

- System Managed Structure Duplexing
  - Optionally turned on for any structure by the system programmer
  - So customer decides what will be duplexed and what will not
- Common exploiters include
  - DB2 locks
  - Should be selectively enabled via a cost / value analysis
- Customer specifies when the structure is define
- When a structure is system managed duplexed
  - z/OS will send the request to both primary and secondary structure
  - But similar to User Managed Duplexing, there must be some sort of locking mechanism to make sure two z/OS systems do not interfere with each other
  - This coordination is done via a CF to CF request
    - Request granted only when both CFs agree
  - Results in extra costs
- With CFCC 15 and some z/OS exploitation (2Q2008) there will be enhancements to cut down on the number of exchanges between CF and CF
  - Rather than 2 exchanges per request there will be only 1

Analyzing Duplexing Performance

- The response time of a duplex request looks as follows:



Data Sharing Over Distances

- When data sharing over distances the following must be considered
  - Speed of CF processor
  - Link technology
  - Distance between z/OS system and the CF
    - Remember, service times are elongated by 10 mics per km due to speed of light through fiber
  - Host impact is capped due to synch to async conversion
- Results
  - Possible longer transaction waits for sync and async requests
  - Could cause an impact to subsystems queues and aggravate lock contention
    - Possible elongated application transaction response times and internal queues
  - Will most likely need more links between z/OS host systems and remote CFs
    - Link buffer (subchannel) busy will grow linearly with elongated service times

Section Summary

- Some information on couple datasets
- Coupling facility request types
  - Asynchronous requests
  - Synchronous requests
- Coupling facility structures
  - List structures
  - Lock structures
  - Cache structures
- How data sharing works with lock structures
  - What is data sharing
  - Data sharing logic using the coupling facility