Introducing:

Extreme CF Locking

Session EE
GSE UK Conference
November 2011

Paul.arnerich@zedskills.com

Contact:
Paul.Arnerich@tsdd.co.uk

Agenda

- **Introduction**
- **Key Plex components**
- **Coupling Facility Performance Factors**
- **Coupling Facility Options**
- **Case Study 1 – Normal Locking**
- **Case Study 2 – Extreme Locking**

- **Acknowledgement**
  - **Several of the performance background slides were pulled from IBM UK STG Training Services course UZ03, with IBMs permission and my thanks.**

Terms

- Plex will be used for Parallel Sysplex
- CF will be used for Coupling Facility
- µ is the SI unit for microseconds
  - ½ the lifetime of a muonium particle (an exotic atom)
  - 1,000 of them makes a millisecond
- References to DB2
  - Whilst other Database Managers exploit the Coupling Facility, this pitch will use DB2 as an example of such a Database manager.
  - Where this pitch says DB2, please substitute for any DBRM you fancy

Introduction

- Wanted to share some recent experiences of CF Locking
- Thought I had seen it all, but …
  - In terms of CF Locking that is
- Advocate of current "Sausage machine" approach to configuration for Sysplex
  - 2 CPCs
  - 2 ICFs
  - 4 LPARs
  - Systems Managed Duplexing
  - …simples
- Wrong again Paul, "one size fits all" doesn't alwa

Diversion – Are you awake test

- What are the key differentiators of System z ?
  - Availability
  - Operability
  - Reliability
  - Security
  - Manageability
  - Other …ities

  - Price (TCO/TCA)
  - Green Stripe
  - Age of admin
  - Male domination

- My favourite is
  - Mixed workload capability
  - Can get a lot of pints to London with one of these

  .. in one trip

Sysplex concept

- Multiple systems viewed as a single system image capable of sharing resources and data
- Achieved by clustering System z hardware and software to provide:
  - Ability to do dynamic workload balancing
  - "Unlimited" capacity with granularity
    - Limited by budget and architecture, but still pretty big
  - Reduced software charges - potentially
  - Single system view, so that multiple system images should be transparent to the applications

  And…(drum roll please)

  - **Continuous availability**
  - Based on no single point of failure, where any z/OS image can actively replace any other z/OS image in a planned or unplanned outage
  - No single point of failure is pretty key

Plex characteristics

- Supports from two to 32 z/OS images
- Key elements:
  - XCF address space
  - XCF (Communications) and XES (Access to CF)
  - GRS
  - Shared Couple Data Sets
  - Coupling Facility to provide data sharing
  - Time synchronization between two or more servers
    - 9037 Sysplex Timer - very old
    - Server Time Protocol – not so old

  ("Time is important, lunch time doubly so" - Ford Prefect)

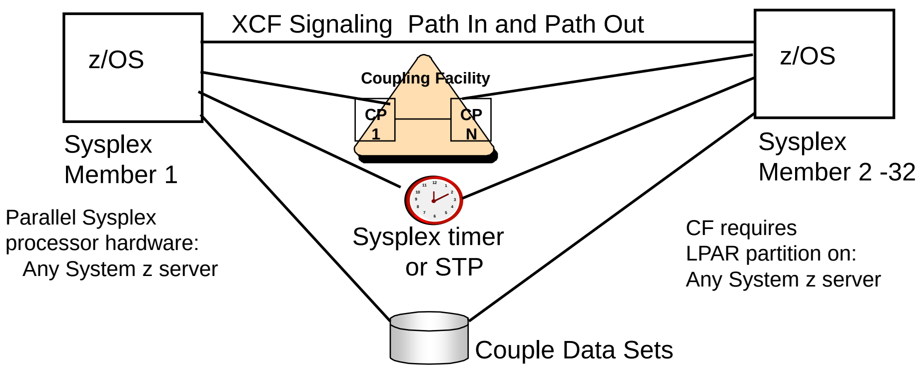
- Double up:
  - If you need one of something, buy/configure two
  - If you need two of something, buy/configure four
    - and so on

Plex enablers

- Two key requirements:
  - Communicate efficiently between z/OS LPARs
  - Fast Shared data area so data can be shared…fast
- Communications – XCF and Paths
  - Delivered via a set of APIs called XCF and some communications paths
  - Paths originally CTCs but can use other means
  - LPAR can tell another LPAR when it needs resources
  - To act as a heartbeat, so when heartbeat disappears, one LPAR can be the 'cleaner'
- Shared data area – XES and Coupling Facility
  - Same DB accessed on two LPARs = integrity problem
  - Solution = a locking mechanism
  - Better be fast and efficient if I want to lock at row level
  - Disk will not do
  - z/OS memory will not do
    - OC4 common enough without introducing I/O based access to memory

Key Sysplex components

- **Need:**
  - **z/OS – more than one most likely**
  - **Communication paths**
  - **Common Time Base**
  - **Shared Memory – Coupling Facility**
  - **Some shared datasets for management**



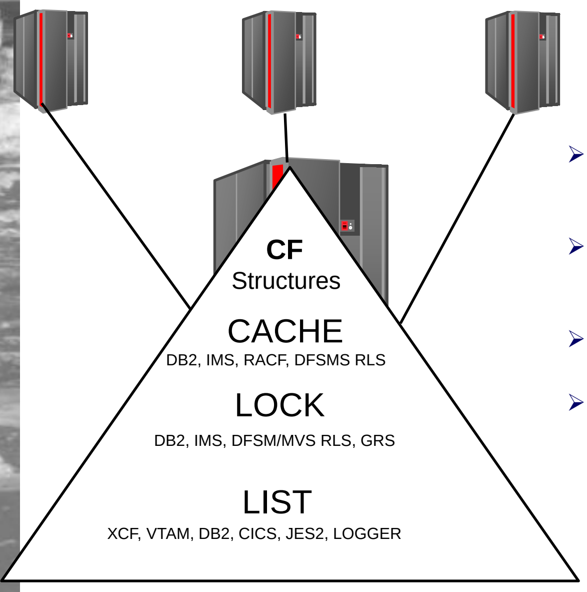
Plex allows.. (1/2)

- **Resource Sharing**
  - **There is such a thing as a free lunch, but….**
  - **Now we can communicate over XCF, lets share logical things**
  - **Nice to have, really nice to have**
  - **Can simplify and improve single point of management**
  - **Delivered by Subsystems that want to exploit it**
  - **Implementation dependent on Subsystem developers**
  - **Can use:**
    - **Just Time**
    - **just the shared datasets**
    - **just the communications paths**
    - **The shared memory area – the CF**
  - **Fantastic, but not the reason you bought the CF though**
    - **Unless you really did justify the CF hardware for a free lunch**
  - **Examples:**
    - **VTAM Generic Resource, HSM Recall Queue, Enhanced Catalog Sharing, Sysplex Consoles, RACF DB Cache, GRS Star, etc.**

Plex allows.. (2/2)

- Data Sharing
  - The money shot (as in action thriller film term)
  - Can deliver very high availability if properly configured
  - Allows multiple database servers on multiple z/OS servers to share, with integrity, the same database
  - This means that your single points of failure can be eliminated which equals higher availability
  - No free lunch, someone has to do some work
  - Subsystem developers must exploit this
    - Need to keep copies of local locks in shared location
    - Need a lock manager to manage this
  - Customer must buy hardware
    - CF CPU
    - CF Links
  - Will have a z/OS CPU effect – known as the 'Host CPU Effect'
    - It will cost MIPs

Data sharing implementation



- Data sharing can implement various structure types depending on the CF exploiter
- A structure is a named area of storage determined by the exploiter
- An exploiter is a software subsystem or application
- This pitch is focusing on Lock structures
  - They cost the most MIPs

Resource v Data Sharing

- How do we distinguish between them ?
- Simple ROT is how you react when you lose a CF
- Resource Sharing = don't panic
  - Ops ring - "CF has crashed"
  - You reply - "So what"

- Data Sharing = panic
  - Ops ring - "CF has crashed"
  - You reply - "Just remembered an urgent doctors appointment, gotta dash, good luck with all that, see you on Monday, ahh.. I forgot, its half term next week so will be back the week after next."

Coupling Facility Factors

- OS called CFCC - CF Control Code
- Usually drawn as a pyramid or triangle shape
- Runs counter to the von Nuemann principle z/OS is based on
- Workload balancing ?
  - No, CPU runs a tightly polling loop
  - Any work ? yes, do it, any work ? no, any work ? no, any work ? yes, do it, any work ? etc
- Need CPU, enough so it doesn't have to wait – ever
  - Special CPU, called ICF, special due to pricing
- Sharing CPU with other workloads ? Pah !
- Data areas in CF known as structures
  - Lock (small), List (bigger), Cache (biggest)
- Needs special I/O connections
  - CF Links, come in a variety of costs, speeds and distance limitations
  - Best ones are the fastest-shortest-most costly

Where do I put the CF

TsdUK

- It matters
- CFCC runs in an LPAR on a System z CPC
- When you lose a CF:
  - DB2 will rebuild the Locks in your spare CF
  - Its magic, just a blip, no outage
  - Its not really magic
    - DB2 gets the locks from the DB2 members on all LPARs
- CF CPC cannot contain z/OS LPARs in the same plex as the CF
  - At least, not if they are running the same DB2
  - If it did and that z/OS was running the same DB2 Data Sharing group, DB2 couldn't recreate the Lock tables, some data will be missing
  - DB2 now has to do group restart and rebuild, some time will be lost, depending on last commit
    - Have seen worst case 32 days with no commit
    - Admittedly it was a zombie

CF considerations

TsdUK

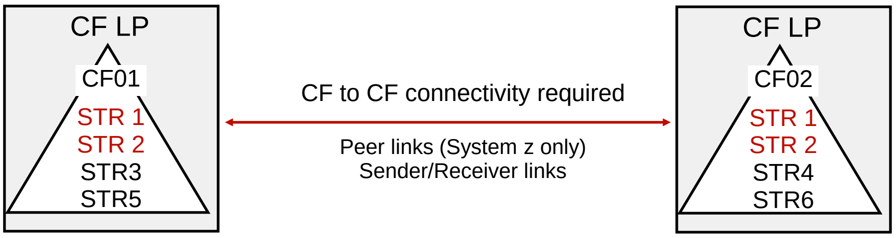
- A hardware failure resulting in both z/OS and a CF failure must not cause an extended recovery time or a Sysplex outage
- This is called Failure Independent or Failure Isolated
- Achieved by:
  - Two "stand-alone" CFs:
  - One "stand-alone" CF and one internal CF:
    - Critical structures should be placed in the "stand-alone" CF for recovery reasons.
  - Two internal CFs in two different CPCs:
- Two CFs in two CPCs is the Current thinking
  - The Sausage Machine solution
  - Because we can Duplex the structures that matter

"Stand alone" CF

- "Stand alone" CF was delivered by 9674
  - Then z900 model 100
  - Then z800 model 0CF
  - Since z990, no more "stand alone" options
- In fact, may as well stop using "stand alone" phrase
- Better to just use FICF
  - A CF in a CPC with other LPARs
  - So long as they are not in the same plex
  - ...actually, so long as they are not in the same plex AND running the same DB2
- Better still, since early 2000's can duplex CF structures
  - Result is, a CF in a CPC with z/OS in the same plex can still be Failure Isolated

System-managed duplexing

- Availability Benefits
  - Faster recovery of structures by having the data already in the second CF when/if a failure occurs
  - Consistent rebuild procedures
  - Allows backup for structures that would otherwise not have any backup capability
- Configuration Benefits
  - Enables the use of non-stand-alone CFs for all resource sharing and data sharing environments.



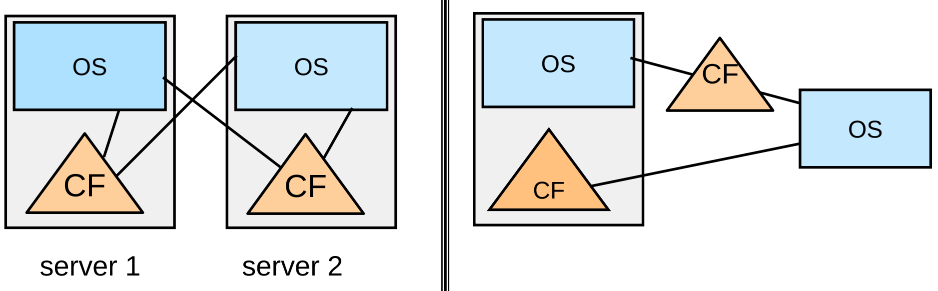
Structures that are duplexed have the same name in both CFs

FICF yes or no ?

Determining failure independence (failure isolation) requirements for ICF configurations

If failure independence requirements=no, this is a valid configuration (requirements=yes, if CF duplexing is enabled properly).

If failure independence requirements=yes, this is a valid configuration.
(best availability configuration with an ICF)



Tuner's View of the CF

- **3 different types of structure**
  - **LIST and "Serialized List" (LIST with a LOCK)**
  - **LOCK**
  - **CACHE**
- **Each exploiter has a different implementation**
  - **Beware of general concepts like "good or bad"**
  - **Use the performance data provided by the 'exploiter'**
- **Need to have a view of workloads and rates**
  - **Are the CF accesses equivalent?**
  - **Are the "service times" equivalent?**
  - **Are there any indicators of delay for links or subchannels?**
  - **What are the CPU indicators?**
    - **CF CPU and z/OS CPU**

CF Request Types

- Synchronous immediate
  - Lock type access
  - Stay SYNC unless XES observes it takes longer than 36 µ
- Synchronous non-immediate
  - Cache type access (transfers of data <4K*)
  - Often converted to ASYNC, depending on Service Time
- Asynchronous
  - Cache type access (transfers of data >4K *)
- But it all depends on the subsystem coder
  - Could request ASYNC for 64K cache

CF performance factors

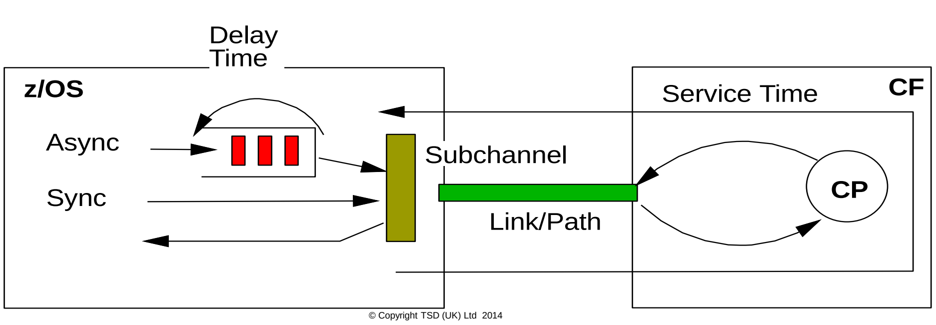
- Data Sharing = no free lunch
- If request is SYNC = spin on z/OS CPU
- Spin = z/OS MIPS
- Ah.. the 'software cost' of Data Sharing
- We measure this by 'Service Time'
- Most requests are ASYNC because:
  - Most exploiters request ASYNC, or
  - XES algorithm converts them to ASYNC
    - Even DB2 Locks if slow enough
- DB2 can require performance of SYNC
  - Higher DB2 software cost caused by Delay management overhead
    - "Are we there yet Dad?"

Other performance factors

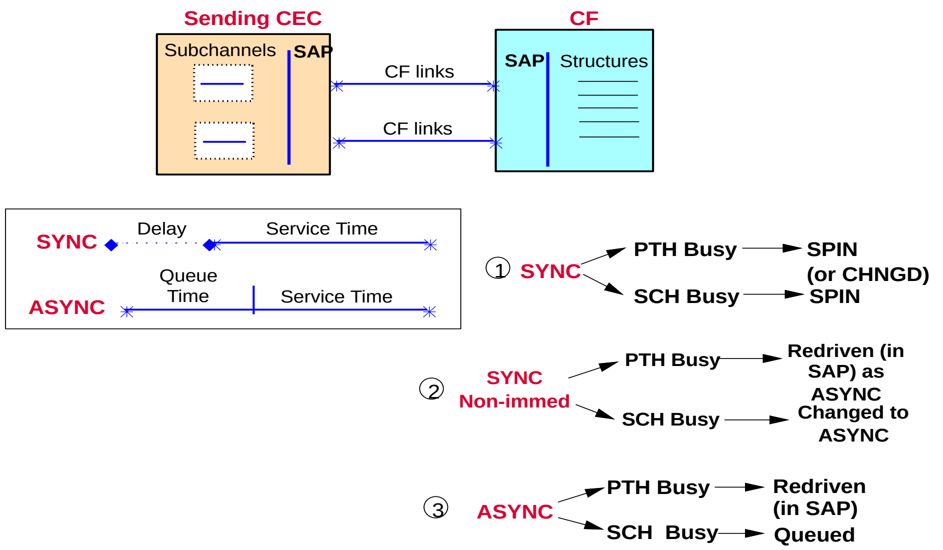
- Number of Links
  - Subchannel/Link busy condition
  - Do not overcommit through excessive "MIFing"
    - Enough SCHs per LPAR to do the work
    - Enough underlying Paths to do the work
  - Need 'enough' CF Links for the workload
- Amount of ICFs – CF CPU
  - ROT Less than 30% util max if single ICF
  - ROT Less than 50% util max if multiple ICFs
- Arrival Rates
- Dedicating CPU to the CF
  - Do not want arrival rates to collide with not being dispatched by PR/SM

Service Time

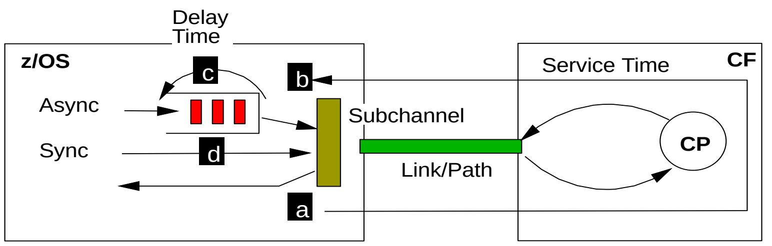
- The key metric
- How we measure CF Access
- The time it takes for a request to leave XES and get back again
- Beautifully reported in SMF 74-4s
- Elongated Service Time may be bad/okay if ASYNC
- Elongated Service Time is very very bad if SYNC
  - z/OS CPU Spin time



Detailed View of a CF Access



CF: Measurements Basic



NB: Async delay not included in Service Time
Av El time is Service Time + Av Delay (/ALL)

```
                              SUBCHANNEL  ACTIVITY
-------------------------------------------------------------------------------------
----------- REQUESTS -----------   ------------------ DELAYED REQUESTS --------------
        #    -SERVICE TIME(MIC)-                 #     % OF   ------ AVG TIME(MIC) ------
       REQ       AVG       STD_DEV              REQ    REQ     /DEL    STD_DEV    /ALL
              b - a
SYNC    247394    38.7      22.3   LIST/CACHE    13    0.0     15.4  c    6.5      0.0
ASYNC   688869   145.6     392.5   LOCK           0    0.0      0.0  d    0.0      0.0
CHANGED      6  INCLUDED IN ASYNC  TOTAL         13    0.0
UNSUCC       0     0.0       0.0
```

- **Don't forget to look at the interval!**

CF Service Time?

- The CF service time is a function of :
  - Hardware speed of the "sender"
  - Hardware implementation of the "sender"
  - CFCC level
  - Number of CF link adapter
  - Type of link adapters (both on the "sender" side and the CF side)
  - Speed of the links
  - Distance (length) of the links
  - Request type and content (SYNC/ASYNC)
  - Distribution of the CF requests arrival – help !
  - CF speed and number of CPs
  - CF CPs dedicated versus shared
  - ISC links shared (Sender "MIFed" between z/OS LPARs)
  - Availability of IOP/SAPs for ASYNC requests

z/OS Dispatching and LUE

- Potentially crucial factor for ASYNC Service Time
- Known as Low Utilisation Effect
- SYNC requests
  - z/OS spins waiting for CF request to complete
  - No need to wait for z/OS to dispatch, it is spinning just dying to dispatch, so completion has no delay
- ASYNC Requests
  - CF completion posts a bit in HSA
  - At some point z/OS must test bit to discover completion
  - Will only happen during a trip through the dispatcher or various interrupt handlers
  - So, time to complete can be affected by dispatch rate
    - If this is a lightly loaded system, that could be … ?
  - And, for dispatcher to run, PR/SM must have assigned a logical CP to a physical CP
    - If LPAR has low weight, that could be .. ?
    - Need MVS Busy to be high in order to get good ASYNC times
    - And HiperDispatch effect ? Lets not even go there

Duplexing Performance

- System-managed duplexing performance issues:
  - Cost to initiate and receive messages will increase for writes
  - Response time for updates will increase
  - CF utilization and link utilization will go up
- Remember, you are gated by the slowest resource
- Some estimates on the cost of duplexing

Costs	Storage	z/OS CPU	Link Time	CF CPU
User Managed (DB2 GBPs)	2x	2x	2x	2x
System Managed (Lock)	2x	4x	5x	8x
System Managed (List)	2x	3x	4x	6x

- RMF provides a separate report on CF to CF link activity
  - Peer Wait and Peer Completion – story for another day

CF Engine Performance

- Shared or Dedicated CPs?
  - Recommend using dedicated engines for CF's
- If you can't follow this recommendation:
  - Give a weight to guarantee share close to one physical CP
- NEVER cap a CF partition, never ever
- Keep a ratio logical/physical as low as possible
- Performance problems that could result:
  - High SYNC times with high standard deviations
  - CPU costs increased
  - Decrease of throughput since tasks have lower priority
  - Note: Dynamic Dispatch is NOT recommended for production CFs (really only valid for Sandpit)
- Optimization hints for managing CF CPs
  - Do not share CF engines among partitions
  - Monitor CF CP utilization
  - CPU processing power required: approx 10% sysplex capacity
    - Very old ROT, could do with re-evaluating

Butchers bill

- Need to calculate the "Host CPU effect"
  - How many MIPs it costs to service requests
- Varies based on:
  - Portion of workload involved in data sharing
  - Access rate to shared data
  - Type of hardware for Host, CF and CF links
  - Number of systems
  - Typical system-level effects
  - Resource Sharing: 3% versus single image
  - Data sharing primary production application: 5% to 10%
  - Individual Transaction/Job effects - can have wide variation
- ITSO workshop has calculation methodology
  - Also in IBM White paper "Systems Managed CF Structure Duplexing" – Appendix A
  - Google ZSW01975USEN - link too long to paste

Case Study 1 – Normal

- European Financial organisation
- Mixed workload
  - OLTP, OLAP, Batch - DB2, CICS, WAS
- Sausage machine config
  - 2 CPCs, z9, 4 Prod LPARs, 2 CFs – one in each CPC
  - CPCs have many GCPs
  - CFs have two ICFs each
  - Other z/OS LPARs in each CPC in other Plexes
  - ISC3 Links 3 km Data Centre distance, z/OS v1.11
- Mix of lock structures
  - DB2 – most converted to ASYNCH
  - GRS – stays SYNC

Case Study1 - Stats

- Stats are taken from 15 minute interval, peak time, typical processing weekday
- GRS Lock
  - 253 per second, SYNC, av. 5.9 µ
- DB2 Lock
  - 3000 per second, ASYNC, 146 µ
- All stats within appetite considering workload and hardware capabilities
- Note DB2 Lock requests get modified to ASYNC
  - XES tries 'some' SYNCH requests every second, if Service time outside 36 µ, modifies remainder to ASYNC
    - Can override value of 36 µ using IBM supplied IXCMIASY
    - Batch driven program to modify value - APAR OA23208
  - Can't say "CHNGED" because this is due to the Hueristic algorithm which does not report as CHNGED – or at all
- Proves Systems Managed Duplexing costs are manageable
- Works well with Balanced Workload

Case Study2 - Extreme

- European Financial organisation
- Single workload
  - Acts as a Database server for *nix based front end application servers
  - 99% of DB2 workload is DDF from *nix
  - QA phase so only minimum production
  - Many application environments
    - QA, Unit test, System test, Roll out test, Engineering test, Test test
    - But no true mixed application workload to performance test, yet
- Sausage machine config
  - 2 CPCs, z10, 4 Prod LPARs, 2 CFs – one in each CPC
  - Other z/OS LPARs in each CPC in other Plexes
  - PSFIB 10m distance, z/OS v1.11
  - CPCs have many GCPs
  - CFs have two ICFs each
- Just DB2 lock structures
  - Most converted to ASYNCH

Problem 1 – Bath Time

- Concern over general CF performance
  - Observed ASYNCH service time as high as 600 µ
- Ran stress test using single lock intensive workload
  - Observed wide range of ASYNCH Service Times
  - 2,000 Locks per second ranging from 400 to 800 µ
- Checked all the usual suspects
  - Including CFCC Level 16 Duplex Completion Protocol
- Eventually raised a PMR
- After much ado, the culprit was deemed to be dispatching - LUE
- Solution, it needs a soak (or bath ?)
  - introduced a soaker workload
  - BR15, odd way to spend your MIPs
  - But it worked superbly
- With a soaker, observed ASYNC Service Time:
  - 140 µ at 2,000 Locks per second
  - Hmm.. Not even as good as z9s, 2km apart, on ISC Links

Problem 2 – Perfect Storm

- No names, no pack drill, but …
  - Several of the application server design points can result in the locking effect being extreme, and ..
  - It uses row level locking, and ..
  - In an HA environment, there is no provision for transaction affinity
    - In reality, workload distribution mechanisms usually employed almost provide "counter-affinity"
- This particular application server architecture coupled with DB2 in a Sysplex has been described as "The Perfect Storm"
- One unnamed senior DB2 specialist comments:

  "It would be hard to design a system that would maximise locking more"

- At the locking levels expected, 140 µ is just not good enough

So another PMR

- Now its about expectations of the architecture
- Projected production would require capacity to sustain 150,000+ locks per second
- CPU cost of this in terms of DB2 Delay management a big concern
- Expectations based on Redbooks, ITSO workshops and migratory swallows:
  - 50 to 250 µ
- Conditions for expectations:
  - z10, using ICB, zero distance, ISGLOCK
  - High number if for large lock (64K)
- Our conditions:
  - z10, PFSIB, zero distance, DB2_Lock
  - Small lock size so expect to see low end, e.g. 50 µ.
- Much testing, hit the wall at 12,000 locks per second
- Still only 140 µ (with soaker just in case)

Expectation realignment

- IBM provided new expectations
  - For this customers config and conditions only
  - Caveat emptor, product may contain nuts, do not use whilst intoxicated, do not iron whilst wearing
- Simplex SYNC Lock
  - IC = 3 to 8 µ
  - ICB4 = 8 to 12 µ
  - PFSIB = 11 to 16 µ
- Simplex ASYNC Locks
  - Any link = 50 to 250 µ
- Duplexed ASYNC Lock
  - Any link = 100 to 400 µ
- Conclusion
  - SMD is great but will not deliver fast enough Service Times for extreme locking
- Solution
  - Acquire additional CPC to act as Failure Independent CF

FICF Results

- All Locks stayed SYNC – less than 36 µ
- High Lock Rate workload
  - Elapsed time from 30 minutes to 5 minutes
  - Lock throughput per second from 550 to 173,000
    - Explains the elapsed time improvement
  - Path delay from 1.5% to 0.4%
  - z/OS CPU cost reduced by 21% - XES (SYSSTC) and DB2
  - 50% less CF CPU consumed (on top of 50% less!)
  - Lock rate of 275,000 per second in one test - smoking !
- Moderate Lock rate workload
  - Elapsed time identical
  - Lock throughput stayed roughly the same
  - Path delay from 0.4% to 0.3%
  - z/OS CPU cost reduced by 11%
    - Mostly DB2
  - 75% less CF CPU consumed (on top of 50% less!)
- Slightly elongated recovery time due to rebuild time
  - From 3 seconds to 5 seconds

Conclusions

- Many factors affect Plex performance
- Need metrics, lots of them
  - CF request Service Time is a crucial one
  - SMF 74-4 is most helpful - need short interval ?
  - RMF PP using OVW very helpful – or Spreadsheet Reporter
- There is no such thing as a free lunch
  - SMD performs well but is not a one size fits all solution
- Watch out for Low Utilisation Effect
- Monitor CF Link PATH delay
- Monitor CF CPU usage
  - Greater than 30% on uni means Service Time will grow
  - Greater than 50% on multi means Service Time will grow
- Even ASYNC Service Time can cost CPU
  - Mostly the XES overhead, also DB2 Contention Management relating to Locks
  - XES runs mostly in SYSSTC, some in SYSTEM

ACL concerns (3)

- **Undercutting**
  - ➔ ACLs make it possible to undercut the access granted in the POSIX settings at User or Group level
- **Management**
  - ➔ Extremely costly to manage if defaults are used
  - ➔ As files/directories created, new ACLs are created
  - ➔ Once ACL exists it is static, no longer tied to default
  - ➔ Makes administration of ACLs a manual effort
  - ➔ Affects every ACL in existence
  - ➔ Any attempt to change user/group structure will result in many hours if not days of ACL analysis just to identify the ACLs that require changing.
- **Agility**
  - ➔ By implementing ACLs, the data centre becomes tied to the depth of discrete definitions for file/directory access and has lost agility

- **AVOID LIKE THE PLAGUE !**

42/50