Now What: So you've loaded up on fast storage, what do you do with it.

Quentin Presley
IBM

Session Code: D8
Tuesday, Oct 15, 2013 (04:30 PM - 05:30 PM)

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988

Agenda

- Locally Attached or Network Attached
- Exclusive vs. Shared Storage
- Where do SSD's fit in?
- Where the best results come from
  - Data Layout
  - Page Cleaning
  - Prefetching
- Storage Groups
  - To Rebalance or Not to Rebalance

NAS in a Nutshell

- Specialized server providing both optimized NFS and optimized CIFS
- Optimized: usually using a stripped-down kernel, sometimes implemented in hardware
- Simpler than SANs
- SANs typically have better performance

SANs in a Nutshell

- SCSI protocol hides physical format of disks
- Initiator sends commands to target, Target sends responses
- Originally, the medium was specified to be a parallel cable
- Target is <SCSI ID, Logical Unit Number>
  - This is the technical definition of LUN
- Traditional deployment is over a Fibre Channel Adaptor
- Fibre Channel over Ethernet could reduce costs with some performance loss.

SAN vs. NAS

SAN	NAS
More complex to manage / maintain.	Easier to manage / maintain.
Traditional deployments (FC) more expensive. FCoE could change this.	Traditional deployments (Ethernet cheaper than SAN.
Typically higher performance.	Typically lower performance.
Much more customizable (RAID 10 for some LUNS, RAID 5 for others)	Not as customizable – built in redundancy varies from vender to vendor, model to model.

SAN vs. NAS

- In reality, both are appropriate within most enterprises.
- NAS is very convenient for general purpose shared file system access:
  - Home directories.
  - Staging spot for backup images, log archives etc.
- High volume database servers tend to use SAN storage over NAS.
  - There are however successful deployments of high end databases using high-end NAS hardware.

Exclusive vs Shared Storage

- Give a server exclusive access to a LUN using LUN masking: allow a server to see only given LUNs
  - Different implementations exist
- What if you want servers to share storage?
  - e.g. DB2 pureScale – data sharing clustered database
- Using SANs for shared storage requires a clustered file system
- Our “fair and balanced” opinion is that GPFS is the best

Where do SSD's fit in?

- Solid state disk:
  - No moving parts
  - No need for a spindle to wait for the sector to spin into place
- Usually uses NAND flash-based memory
- Same interface as HDDs (SCSI, SATA, PCI, etc.) typically used

✓ Fast
✓ High MTBF
✓ Energy-efficient

× Expensive
× Limited number of writes

Database SSD Usage Patterns

- Entire Database on SSD
  - Very simple model
  - Not cost effective today
  - Longer term trend – particularly for OLTP systems
- Temp on SSD
  - Simple model
  - Beneficial for workloads that are I/O intensive and include large sort spill operations
    - Complex Queries/Index Creation/Reorg for clustering
    - NOTE: Containers still need to be accessible after a crash (reorg recovery) – so host attached flash memory is not necessarily an option.
- Logs on SSD
  - Cost effective today
  - Benefit is debatable given write cache in storage controller
  - Possibly beneficial for high write workloads where log is bottleneck and storage controller has small write cache.

Challenges with SSDs

- Temporary table spaces on locally-attached SSDs or flash memory (e.g. PCI attached Fusion I/O card) can be an HA issue
  - Active/passive HA system – active system crashes with a reorg in progress
  - Reorg may have written data out to temp containers and at crash recovery time, DB2 will find that the temps are gone!

TIP Solution:

- Assume you have a temporary table space FASTTEMP which is backed by locally attached SSDs or flash storage
- Ensure that the above is bound to a nice (normal) large buffer pool
- Create a separate temporary table space REORGSP specifically for use with reorg – it should be backed by storage that will be accessible in the event of a crash e.g. traditional HDD storage on a SAN
- Bind it to a separate, small buffer pool (e.g. 1000 pages) – will ensure it does not get picked by the optimizer vs FASTTEMP
- Explicitly specify REORGSP to be used in reorg commands using the “USE <tbspace-name>” clause

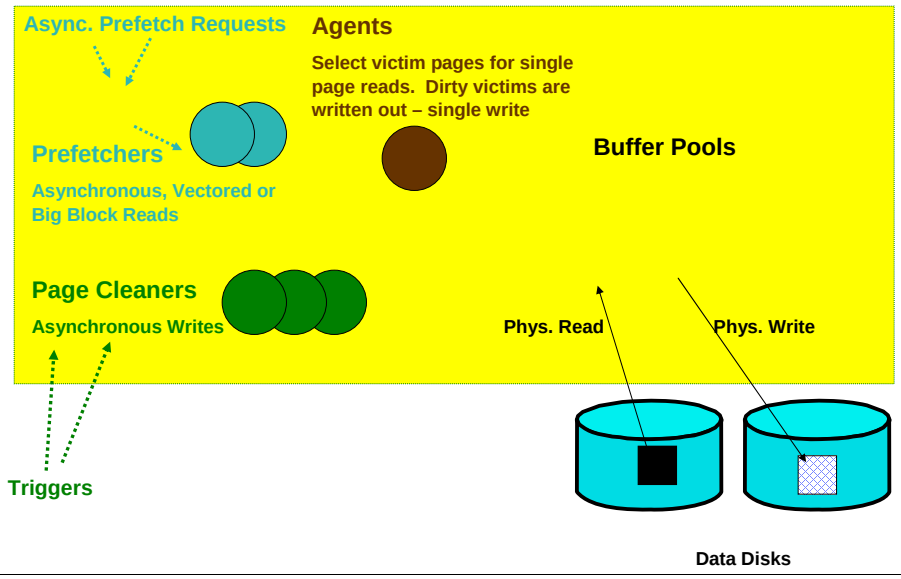
How data layout helps (ESE and DPF)

- Each database has a dedicated set of LUNs
- One LUN per data file system
  - DB2 does the striping
- Separate log file system and storage path file system(s) (i.e. table spaces)
  - Different LUNs and storage arrays ideally

How data layout helps (pureScale)

- One or more LUNs per data file system (pureScale)
  - Either DB2 or GPFS does the striping
- Single log file system that is shared by multiple members – ensure enough I/O bandwidth for logging
  - Typical 15k RPM disk can do ~150 IOPS
  - Some measurements might be necessary to estimate log bandwidth required
- Too many file systems for storage paths is not necessarily a good thing!
  - Each file system adds to file system recovery time – can impact overall recovery times for member and CF failures.

How A Page Flows Through The Buffer Pool



Page Cleaning

- There are two types of dirty pages that are written by Page Cleaners:
  - 'Old' pages
  - 'Cold' pages
- DB2 Page Cleaners respond to 'triggers'
  - LSNGAP
    - Controlled by SOFTMAX configuration parameter
  - Threshold
    - Controlled by CHGPGTHRESH database configuration parameter
  - Dirty Steal
    - Ideally eliminated by careful tuning of CHGPGTHRESH and NUM_IO_CLEANERS configuration parameters

Page cleaning best practices

- NUM_IO_CLEANERS = AUTOMATIC
  - The automatic setting takes into account system specific values and is sufficient for most workloads.
  - Choosing a fixed value should be done with thorough testing and monitoring of asynchronous vs. synchronous write activity.
- SOFTMAX setting should be a function of the desired recovery time
  - High SOFTMAX = Less page cleaning and longer recovery time
  - Low SOFTMAX = More page cleaning and shorter recovery
- CHNGPGS_THRESH setting should be tuned based on thorough testing and monitoring of dirty steals
  - pool_no_victim_buffers monitor element

Alternate Page Cleaning (APC)

- db2set DB2_USE_ALTERNATE_PAGE_CLEANING = ON
- APC is a proven alternative on heavy pure OLTP workloads (e.g. TPC-C)
  - APC does not clean from block based buffer pools at all.
- Traditional Page Cleaning
  - Threshold trigger
  - Dirty steal trigger
  - LSNGAP trigger
- Alternate Page Cleaning
  - No external triggers exist

Alternate Page Cleaning (APC)

- CHNGPGS_THRESH configuration parameter is ignored
- SOFTMAX configuration parameter is still respected.

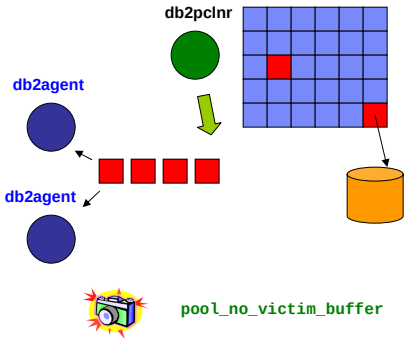
How does it work ?

Victim selection

- The page cleaner pre-selects victim pages for agents.
- If this list of pages falls below 1% of the buffer pool, agents internally trigger cleaners. Cleaners populate, this list up to 2% of the buffer pool.

SOFTMAX (LSNGAP) cleaning

- Instead of waiting until a LSN gap interval occurs, page cleaners use a predictive algorithm to prevent a LSN gap from occurring.
- The result is a much more even write I/O throughput.

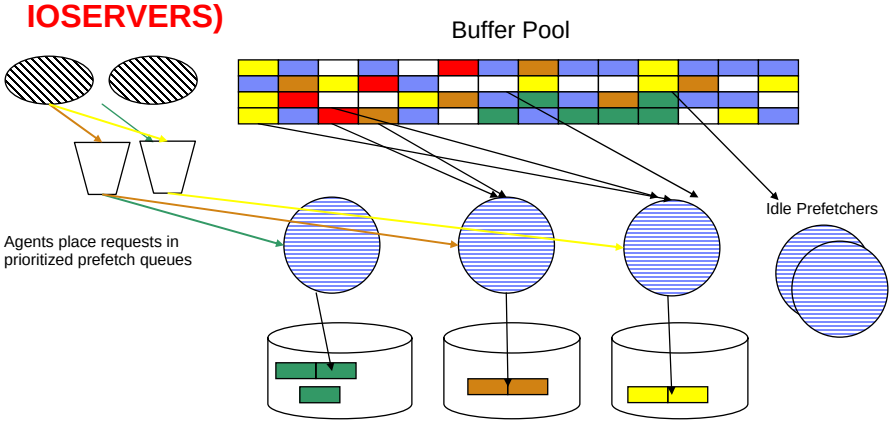


Rules of thumb when considering APC

- Pure OLTP workloads can benefit from APC.
- For DSS workloads traditional page cleaning is recommended
- For mixed workloads APC has shown benefit in many cases and should be considered on a case by case basis.
- Pages are cleaned more aggressively with APC than with traditional page cleaning.
  - 32 I/Os outstanding with traditional
  - 256 I/Os outstanding with APC

Prefetching

I/O Prefetching done by DB2 by Prefetchers (aka. IOSERVERS)



Pages are read directly into buffer pool memory on most platforms. Due to limitations on some platforms I/O is done in big block reads to a private buffer within the prefetcher then copied into buffer pool memory

Prefetching

- DB2 performs three kinds of prefetching
  - RANGE
    - Sequential access either in the query plan or through sequential detection at run time.
  - LIST
    - Prefetches a list of pages that are not necessarily sequential (although they may be)
    - DB2 will convert the LIST request into one or more RANGE request if sequential ranges exist
  - LEAF
    - Used to prefetch an Index leaf page and the Data pages pointed to by the leaf.
      - Leaf page is done as a single I/O.
      - Data pages on the leaf are submitted as a LIST request

Prefetching – Best Practices

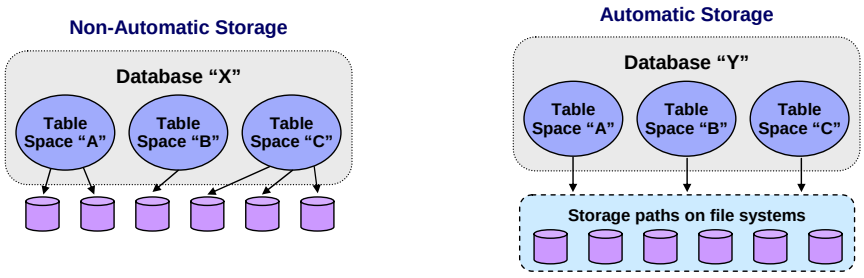
- Set NUM_IOSERVERS and PREFETCHSIZE to AUTOMATIC and monitor to see if further tuning is required.
  - Automatic should be sufficient for most workloads
- Don't allow prefetched pages to be victimized BEFORE the application gets to use them.
  - unread_prefetch_pages monitor element
  - A high value is a sign of over configured Prefetching
- High prefetch_wait_time values indicate prefetching is under configured
  - For high prefetch_wait_time, consider making prefetching more aggressive through increased PREFETCHSIZE or check to ensure that you have enough prefetchers configured (NUM_IOSERVERS)
- For workloads that do not perform well under AUTOMATIC, start with the following guidelines.
  - Start with on prefetcher per CPU and tune from there.
  - Start with PREFETCHSIZE equal to a multiple of extent size and tune from there.

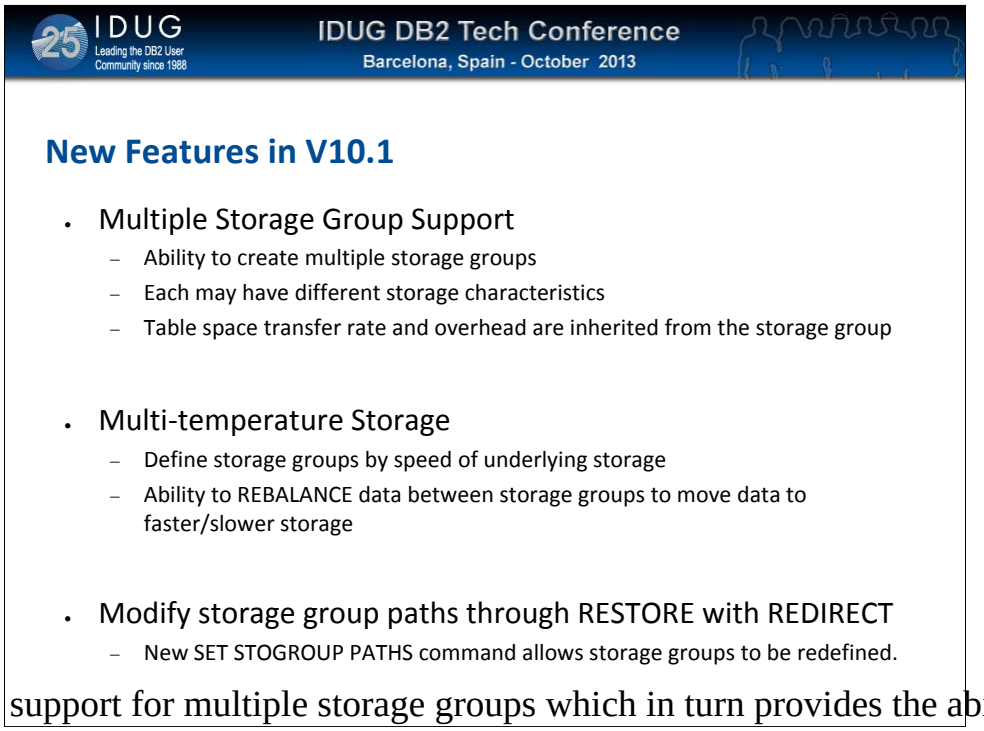
What to tune

- Set EXTENTSIZE = one raid stripe (i.e. contiguous data on one RAID array)
- Set PREFETCHSIZE to AUTOMATIC
- Generally no need to set DB2_PARALLEL_IO provided you are using multiple containers / multiple storage paths

Why the need for Automatic Storage?

- Today, there are multiple points of storage management in DB2:
  - Table data, indexes, long fields and LOBs, log files, etc.
- Automatic storage provides a single point of storage management for the table spaces in the database
  - Addresses the complexities involved with table space storage management, resulting in less time spent planning and managing
  - Significantly reduces human costs
  - Automatic Storage is IBM's strategic direction going forward for DB2 LUW





V10.1 adds support for multiple storage groups which in turn provides the ability to define sets of storage based on different storage characteristics (read/write times, etc.). Two specific characteristics (TRANSFERRATE, OVERHEAD) frequently used within DB2 can be set at the storage group level and all associated table spaces will inherit the values.

To help ensure predictable performance, all the paths that you assign to a storage group should have the same media characteristics: latency,device read rate, and size.

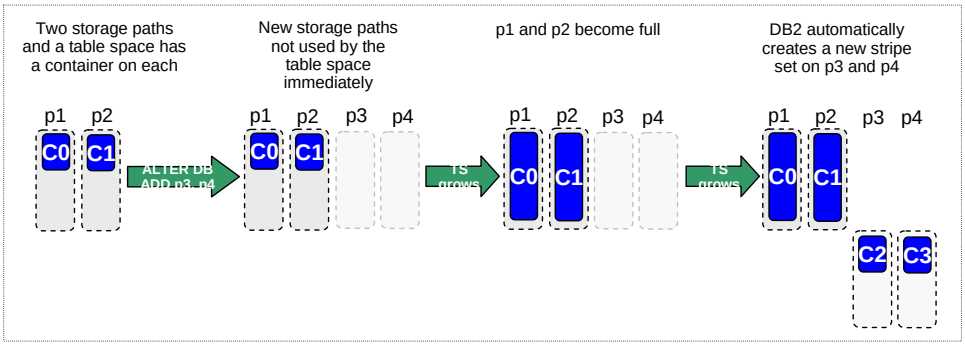
When an automatic storage table space inherits the TRANSFERRATE attribute setting from the storage group it is using, the DEVICE READ RATE attribute value of the storage group, which is in megabytes per second, is converted into milliseconds per page read accounting for the PAGESIZE attribute setting of the table space.

The REBALANCE command has been updated in V10.1 to support rebalancing between storage groups thus providing a method of moving table spaces between multiple levels of storage. This is commonly known as Multi-temperature Storage as storage groups can be defined to use faster and slower storage types.

If you have multiple tables that share a table space, ensure that they have the same temperature characteristics. All the data in a table space is moved when its storage group is altered.

Rebalance

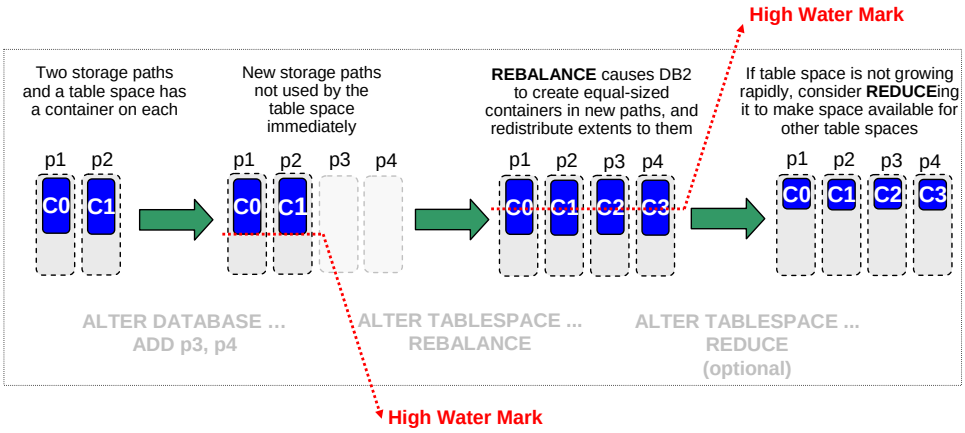
- Let's review what occurs when paths are added to your automatic storage pool…



- Works fine – and is automatic – but what if you wanted to stripe your data across all 4 devices immediately (perhaps to gain more I/O parallelism)?

Rebalance (cont.)

```
ALTER TABLESPACE <tsName> REBALANCE
```



So When Do I Rebalance?

- REBALANCE is expensive and can cause system slowness (very I/O intense)
  - Proper use of storage groups and data layout will reduce the need to REBALANCE.
- REBALANCE once complete can result in better concurrency.
  - By striping data across more devices you can improve prefetching and page cleaning concurrency.
  - Too much of a good thing can cause complexity and inconsistent performance. Over rebalancing can result in difficult to maintain striping requiring changes in prefetcher and page cleaner tuning.

Multi-temperature Storage (by example)

- Multiple Storage Groups can be used to separate hot data from cold data
  - Storage Group SG1 uses only SAN attached SSDs and is used for the current months data.
  - Storage Group SG2 uses SAN attached HDDs and is used for frequently accessed historical data (previous 12 months).
  - Storage Group SG3 uses Tape Drives and is used for historical data accessed very infrequently (previous 7 years).
- REBALANCE is the mechanism used to move data from one Storage Group to another.
  - At month end a REBALANCE would be issued to move the current month from SG1 to SG2.

Rebalancing Best Practices

- When adding storage to a Storage Group use consistent numbers and sizes of storage devices to avoid the need to REBALANCE.
  - Adding consistent numbers and sizes of storage devices ensures that data striping is consistent throughout the table space.
- When dropping a storage path immediately REBALANCE if possible.
  - Dropped storage paths are not fully released from use until all data has been rebalanced off the dropped path.
- Use REBALANCE to move between storage groups.
  - Systems using multiple storage groups to ensure hot data is on fast storage and colder data is on slower storage should make use of REBALANCE to move between storage groups.

Questions?

Quentin Presley
IBM
qpresley@ca.ibm.com

Session
Now What: So you've loaded up on fast storage, what do you do with it.

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988