DB2 for Linux, UNIX, Windows

Tuning DBM and DB Configuration Parameters

Philip K. Gunning
DB2 Master Practitioner
Gunning Technology Solutions, LLC

21 September 2005

- DB2 ™ is a registered trademark of IBM Corporation. Oracle is a registered trademark of Oracle Corporation and SQL Server is a registered trademark of Microsoft.
- DB2 Universal Database (DB2 UDB) is a registered trademark of International Business Machines Corp.
- Other logos and product/trade names are registered trademarks or trademarks of their respectivecompanies.

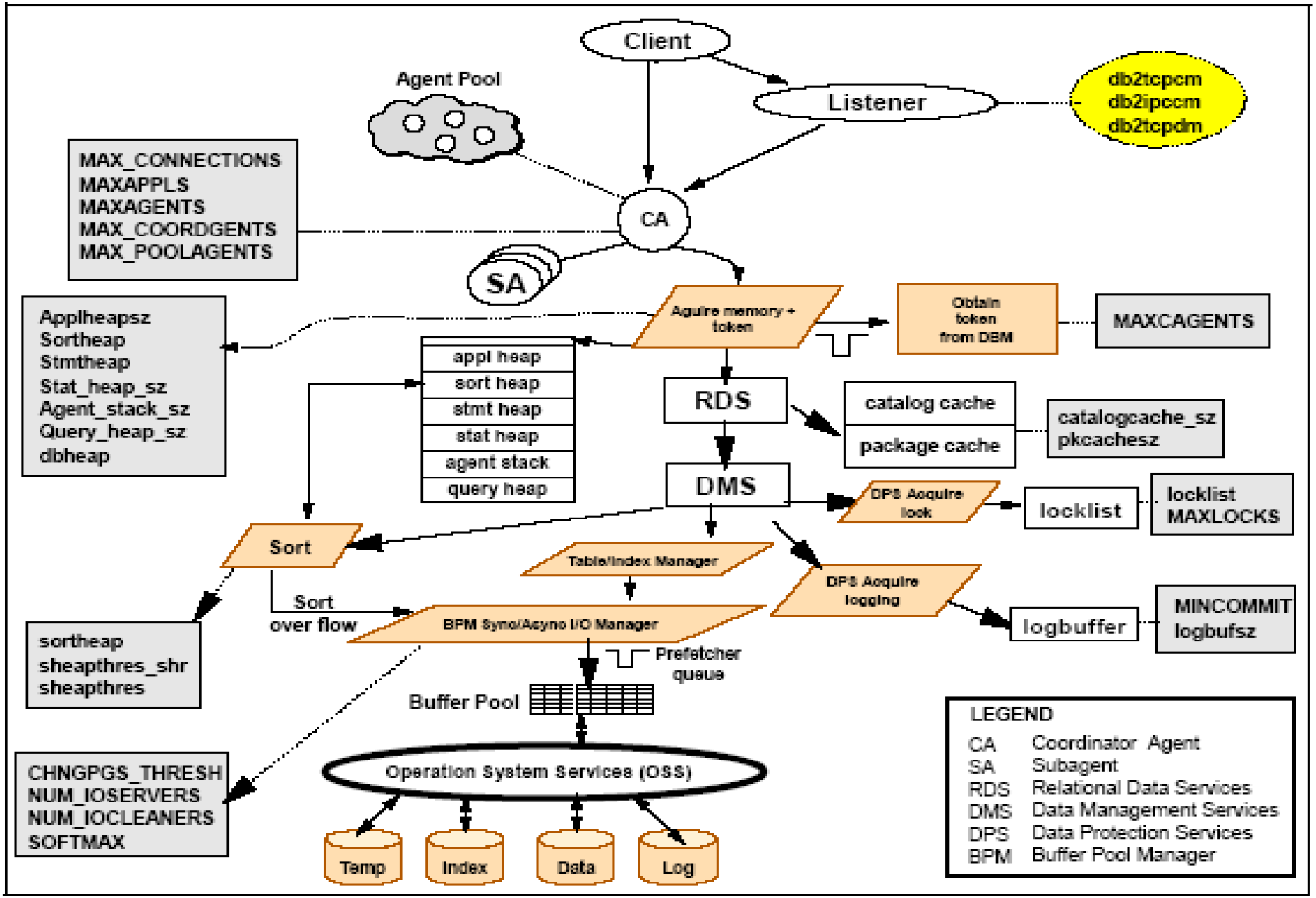
Agenda

- Introduction
- DB2 Process Model
- DB2 Memory Model
- Configuration Parameters
- Agent Related Parameters
- Conclusion

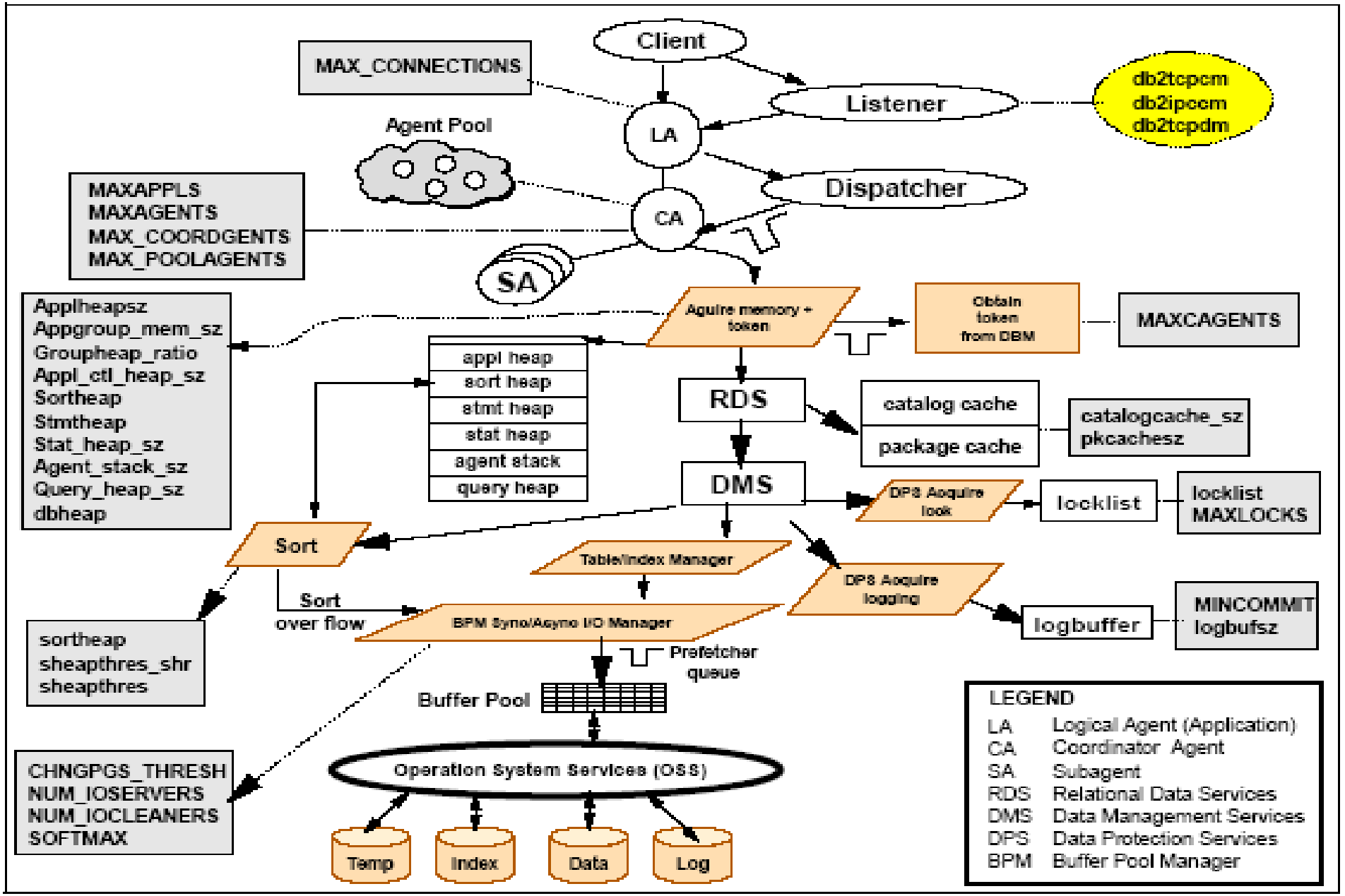
Introduction

- Successful tuning requires knowledge of DB2 processing and available monitoring facilities

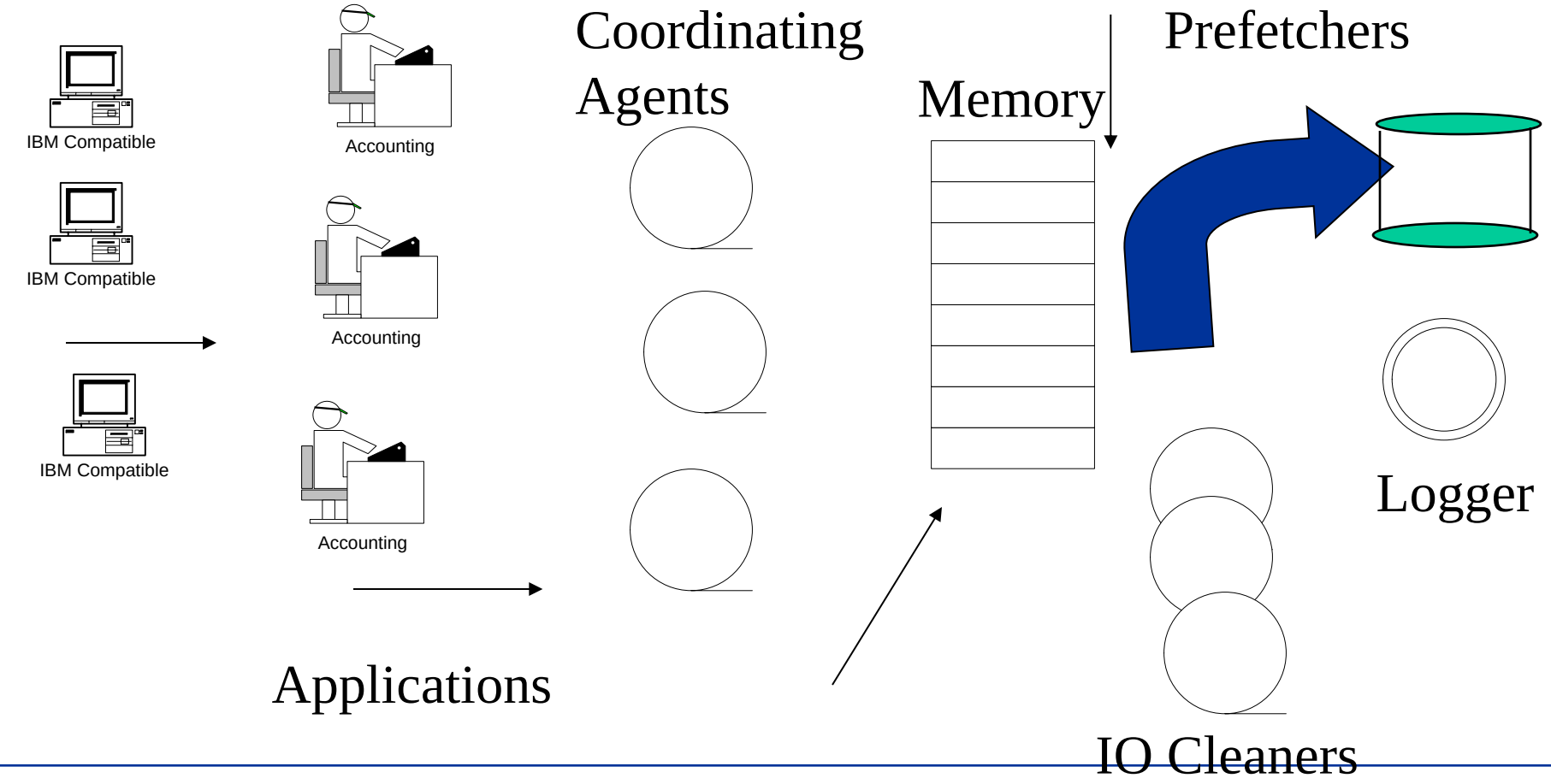
DB2 Process Model



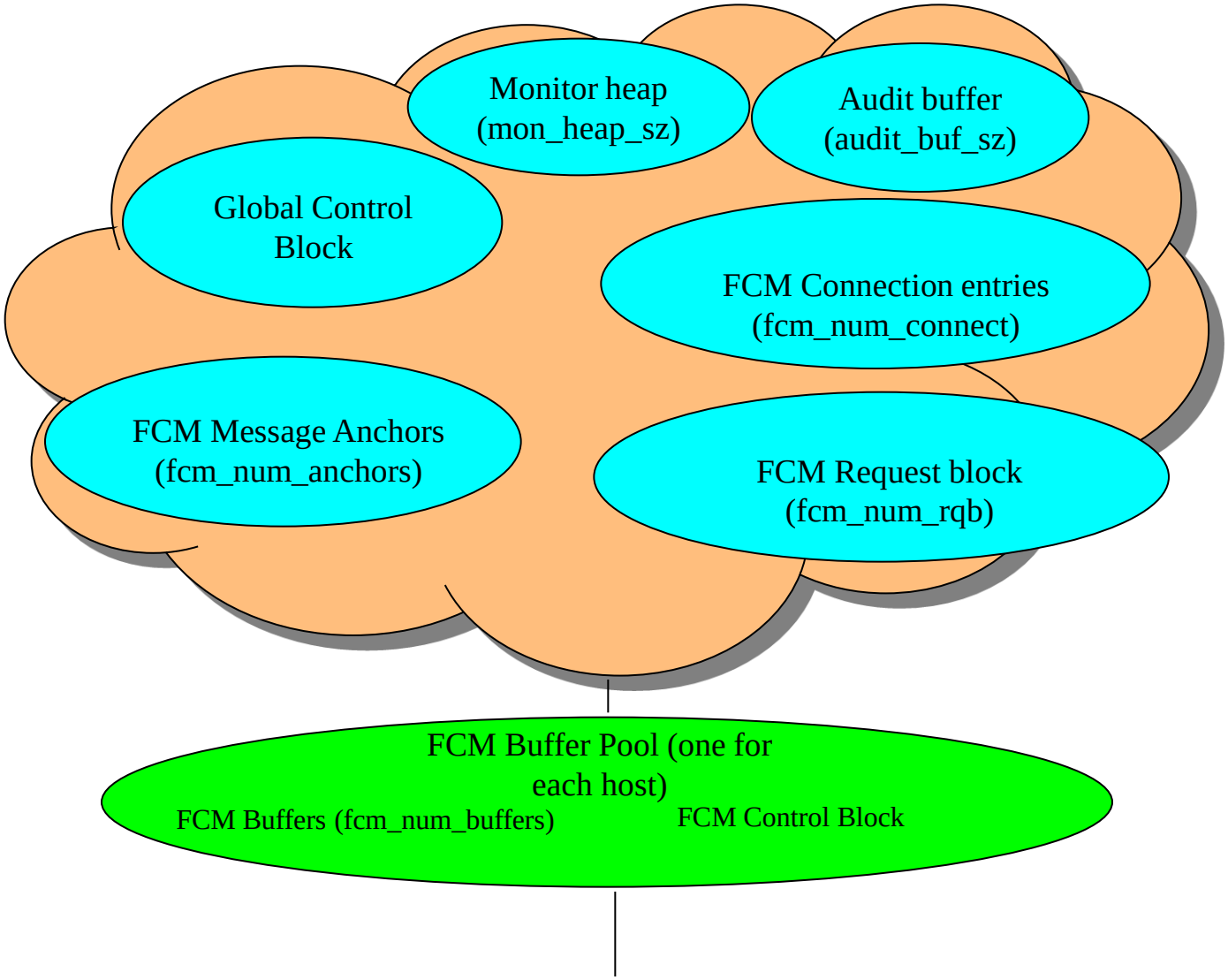
DB2 Process Model - Concentrator



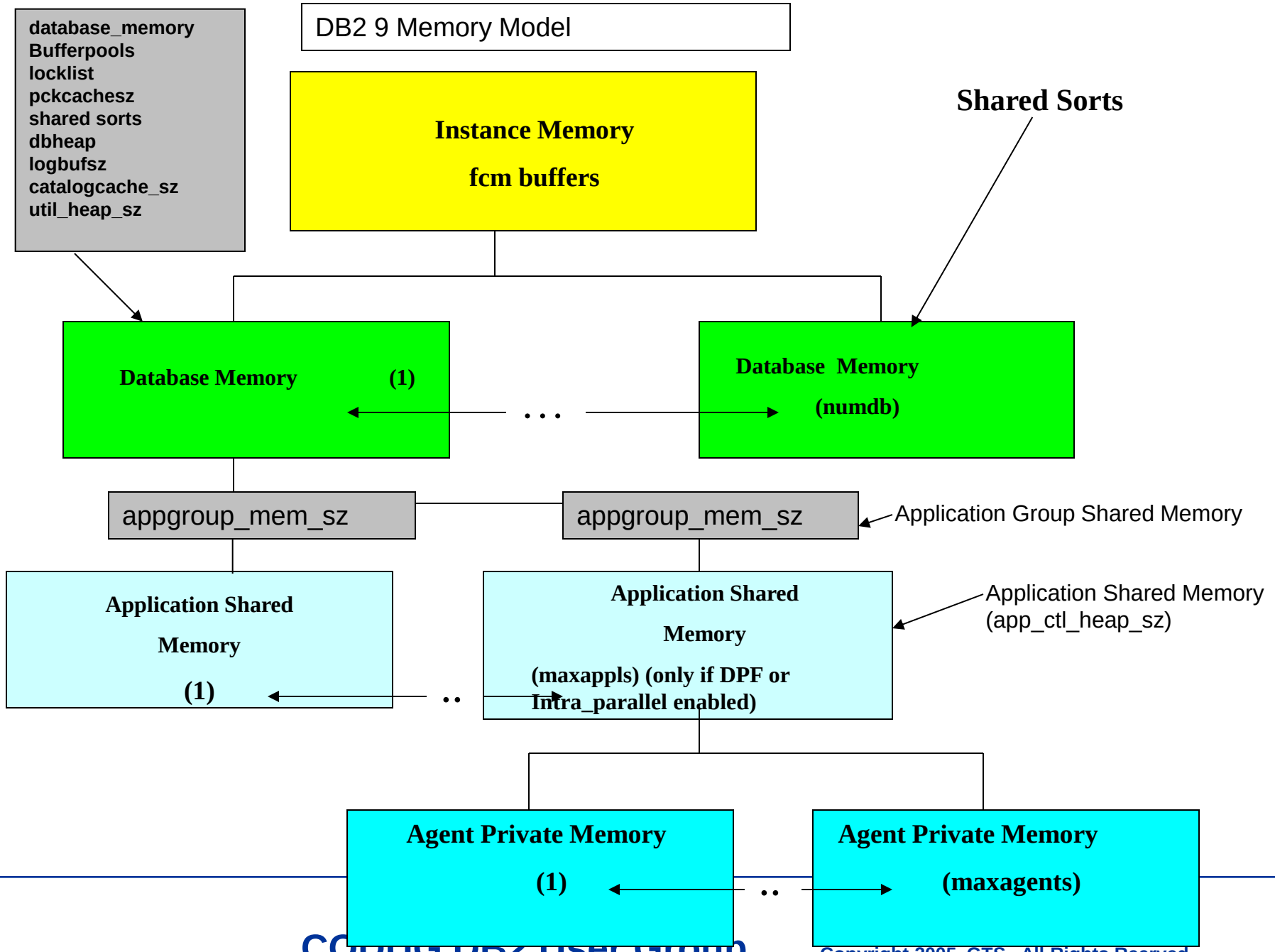
DB2 UDB Process Model



Database Manager Shared Memory



Database Global Memory



Application Groups

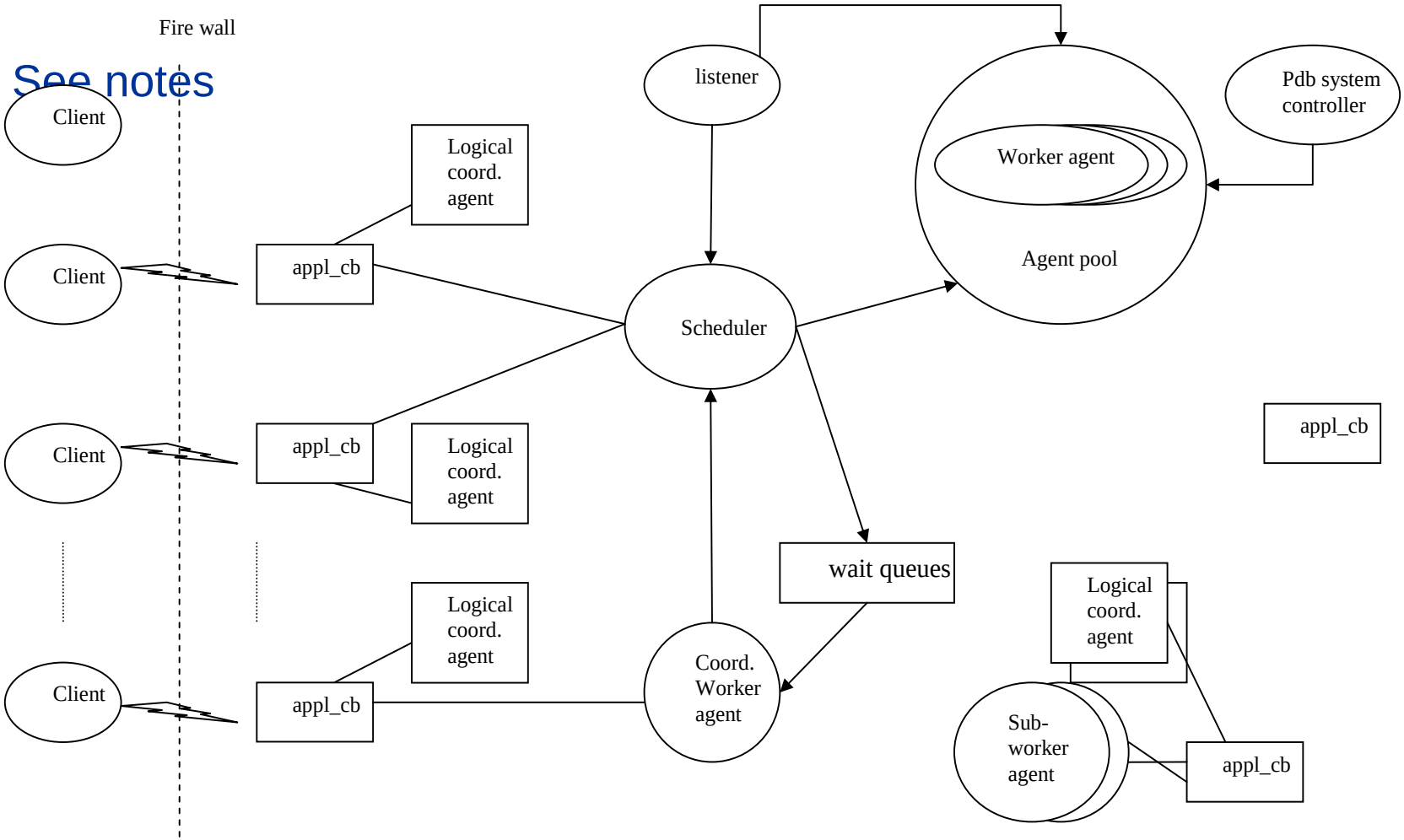
- See notes

Application Groups

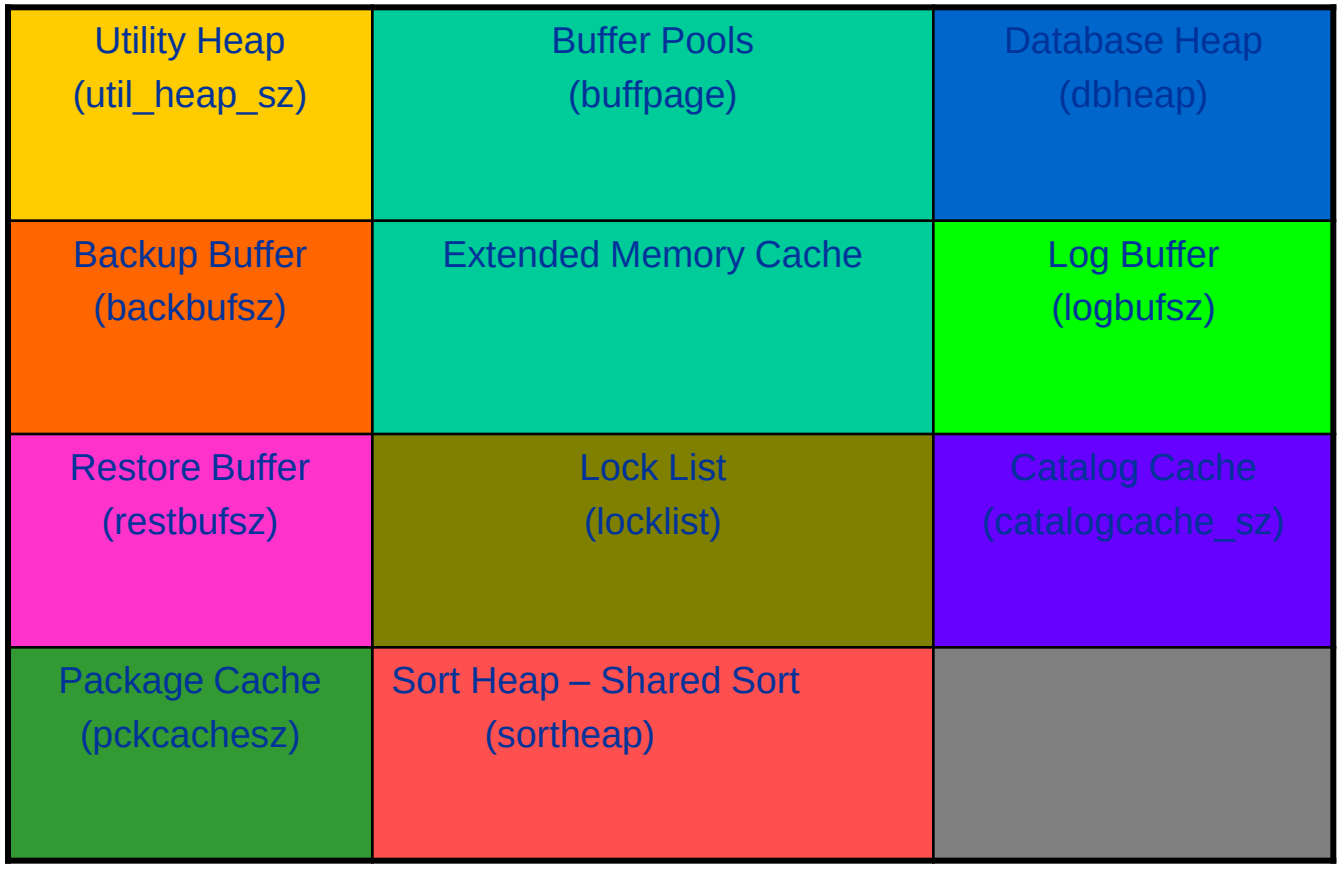
- See notes

Connection Concentrator

- See notes



Database Shared Memory



Database Shared Memory

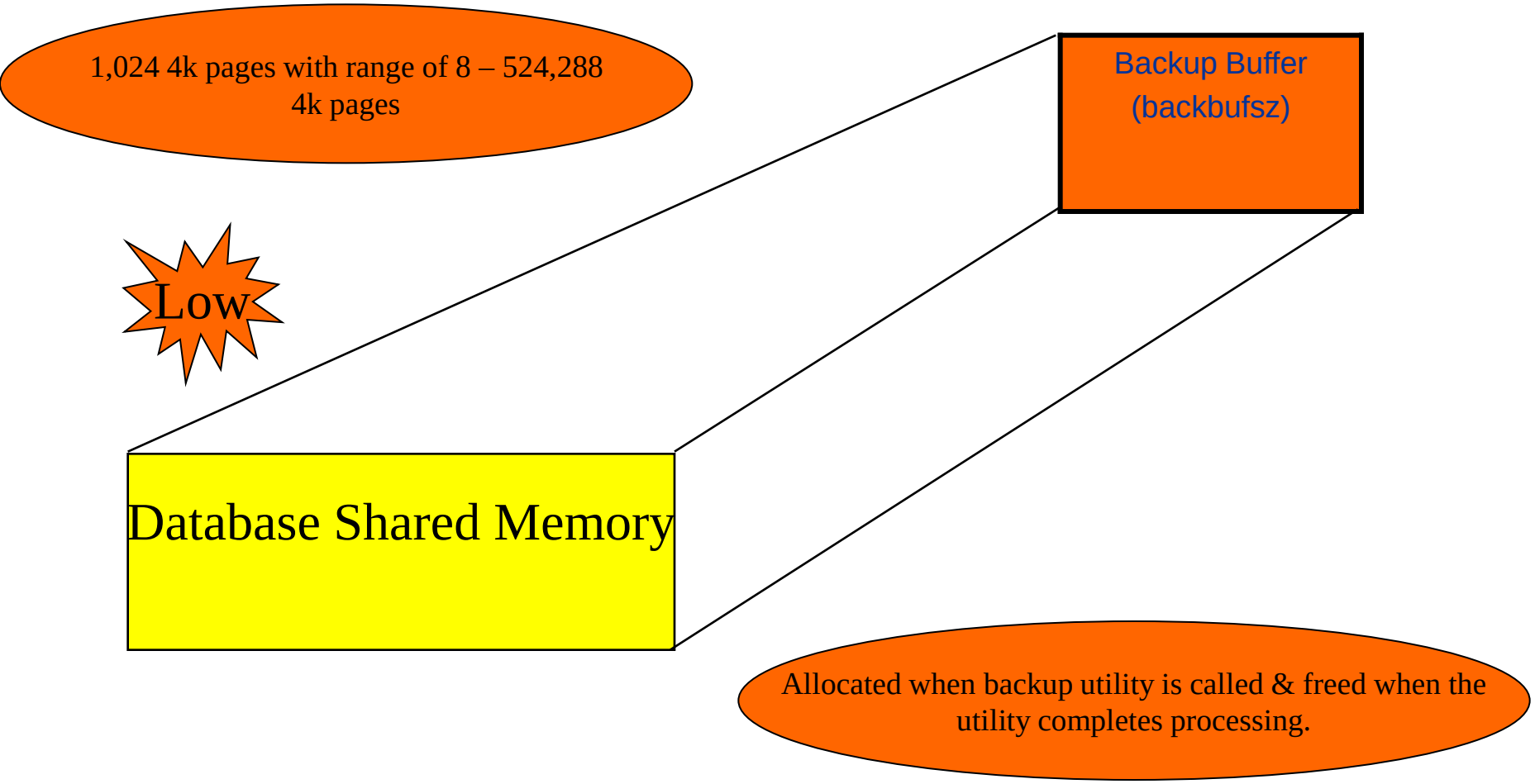
5000 4k pages with range of
16 – 524,288 4k pages

Allocated when needed by
backup, restore, and load utility & freed
when no longer needed

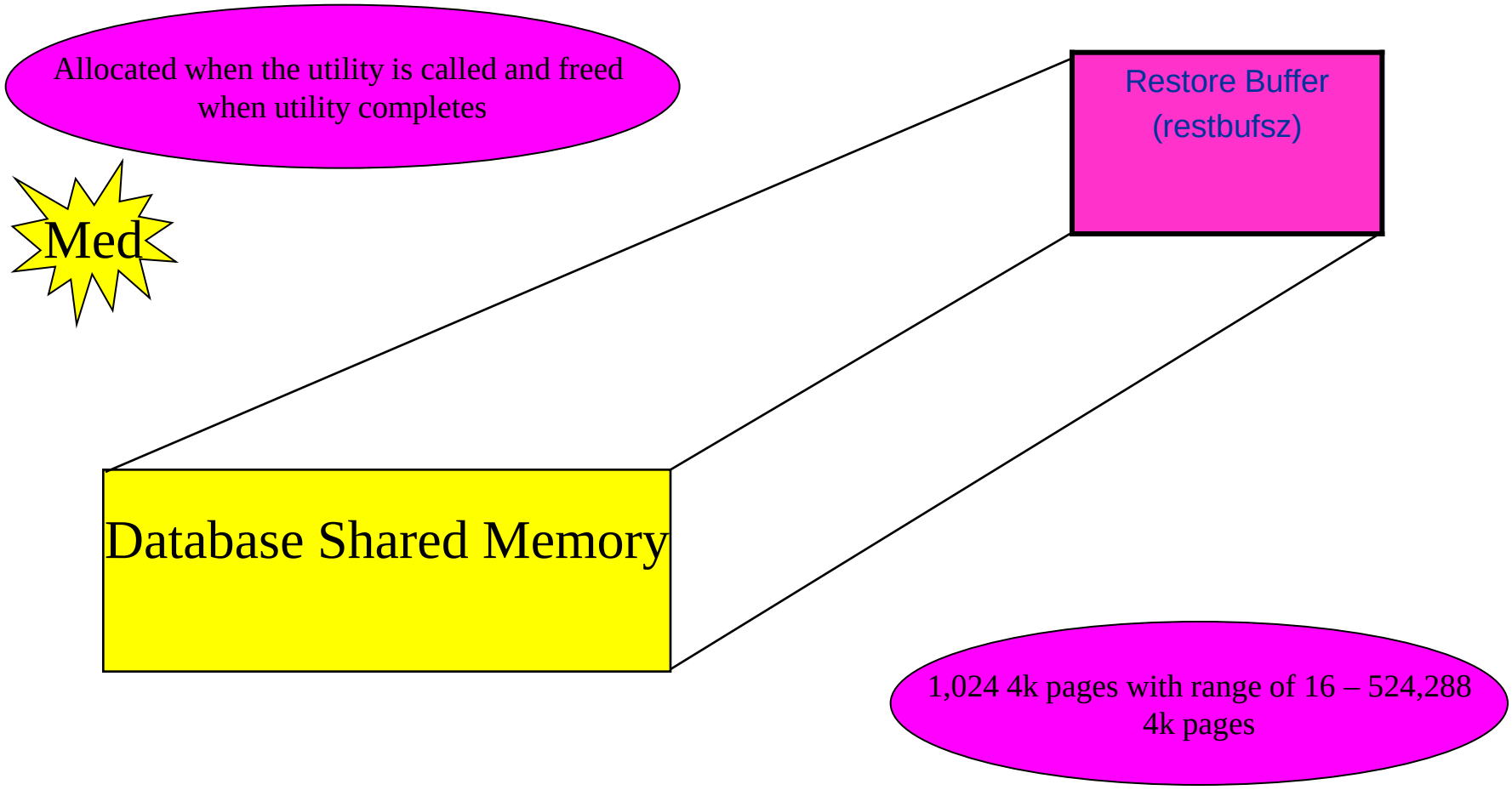
Utility Heap
(util_heap_sz)

Database Shared Memory

Database Shared Memory



Database Shared Memory



Database Shared Memory

8x maxappls or 32, whichever is largest with upper limit of 64,000 (32 bit) or 524,288 (64 bit) 4k pages depending on OS

Package Cache (pckcachesz)

Database Shared Memory

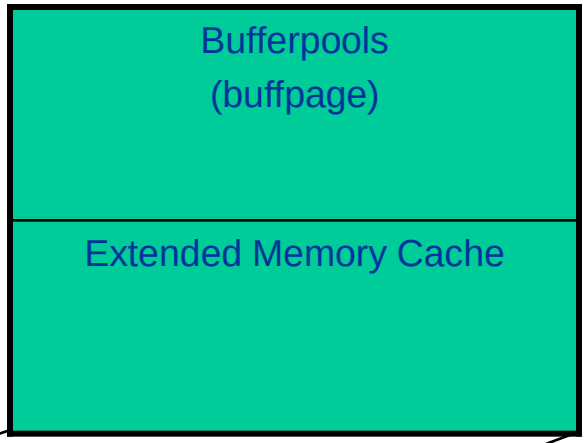
Allocated when the database is initialized and when the database is shutdown

Must be large enough to hold all SQL statements that are executing concurrently. Package cache reduces overhead by eliminating the need to access catalog and by eliminating a prepare or the load of a package

Good use of bufferpools can give you the biggest bang for the buck. Can offset bad design to some extent

50 - 75% of physical memory can be devoted to bufferpools if dedicated database server

Database Shared Memory



Database Shared Memory

Recent presentation indicated one of the top reas
poor performance is using default buffpag

Use of extended memory cache can be beneficial when the amount of real memory available exceeds the addressability of the OS, workload is mostly read only, and when the workload is IO bound.

Database Shared Memory

Allocated at first connect and freed when last application disconnects from database

Lock List
(locklist)

Database Shared Memory

Default depends on OS with range of 4 – 60,000 4k pages

Used in conjunction with maxlocks. Maxlocks specifies percent of locklist any one application can use before escalation takes place

Database Shared Memory

Allocated at first connect and freed when last application disconnects from the database.

Database Heap (dbheap)

Database Shared Memory

Default depends on the OS with a range of 32 – 524,288 4k pages

Log Buffer, bufferpool control blocks, and Catalog Cache are allocated from dbheap

Database Shared Memory

Allocated as part of dbheap

Log Buffer
(logbufsz)

Database Shared Memory

8 4k pages with range 4 – 4,096 4k pages
(32 bit) & 4 – 65,535 4k pages (64 bit)

Database Shared Memory

Default depends on OS with range of 16 – 60,000 4k pages

Catalog Cache (catalogcache_sz)

Database Shared Memory

If not large enough increase a few pages at a time

Stores table descriptor info used when table, view, or alias referenced during compilation of an SQL statement

Monitor using elements:
cat_cache_lookups, cat_cache_inserts, cat_cache_overflows, cat_cache_heap_full

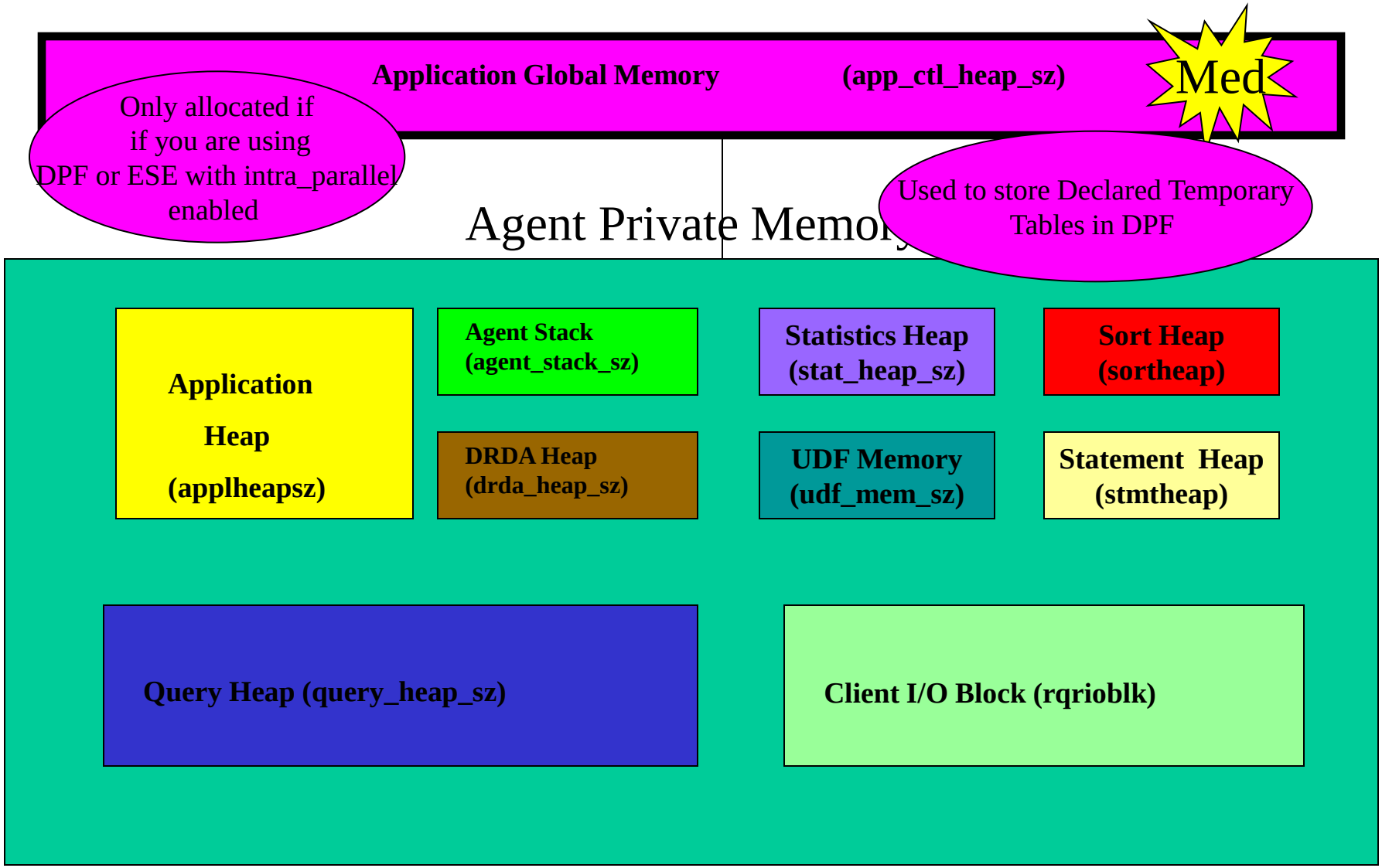
Database Shared Memory

Sort Heap – Shared Sort
(sortheap)

Database Shared Memory

SHEAPTHRES is an instance wide soft limit for private sorts

SHEAPTHRES for shared sorts is an instance wide hard limit on the on total amount of memory used by shared sorts at any given time



Agent Private Memory

Allocated when agent initialized and freed when agent completes work for an application. Stores copies of executing SQL statements

Application Heap (applheapsz)

Default of 128 or 64 4k pages depending on DPF or not with a range of 16 – 60,000 4k pages

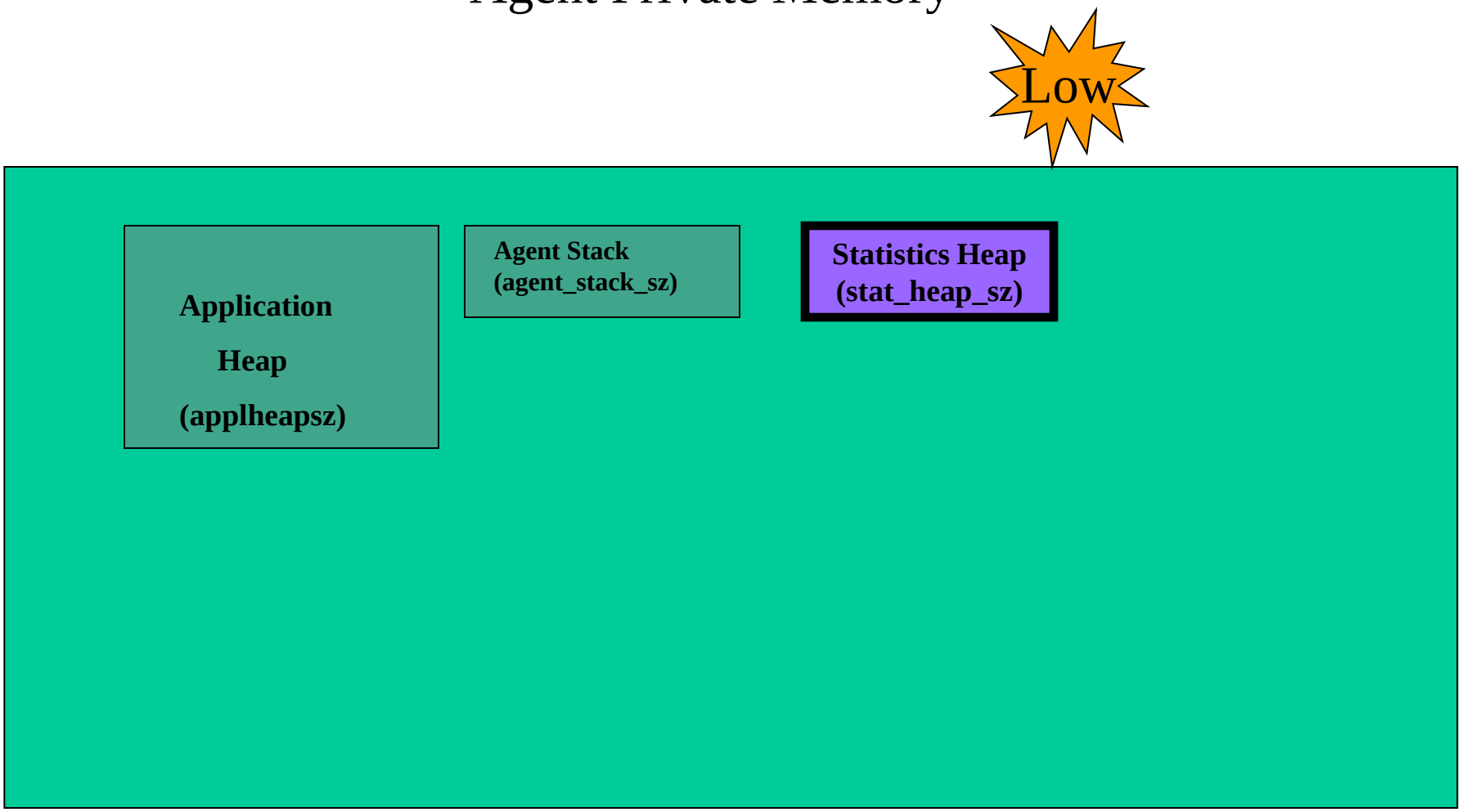
Agent Private Memory

Application Heap (applheapsz)

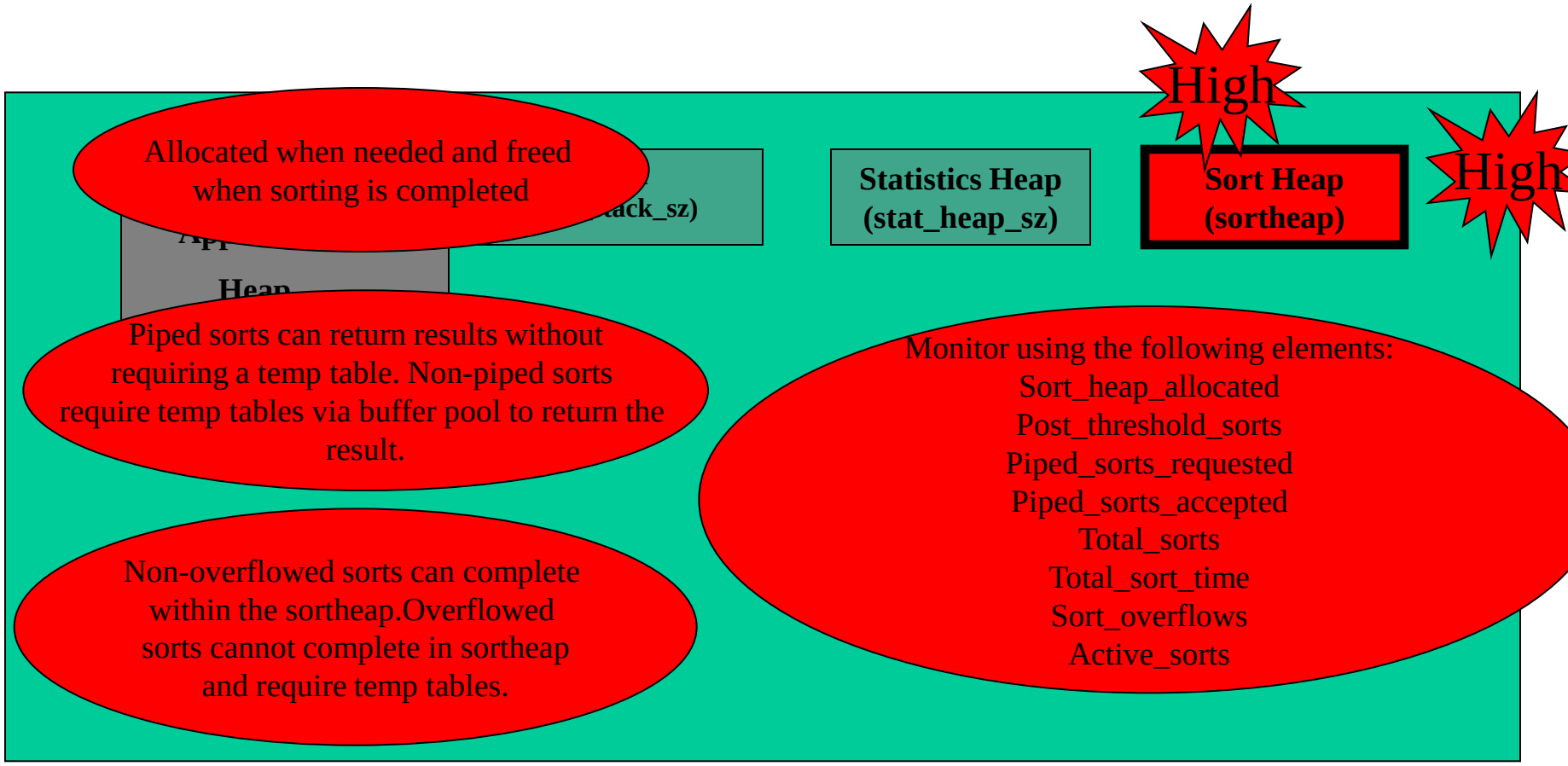
Agent Stack (agent_stack_sz)

Low unless over-allocated then OS paging may occur

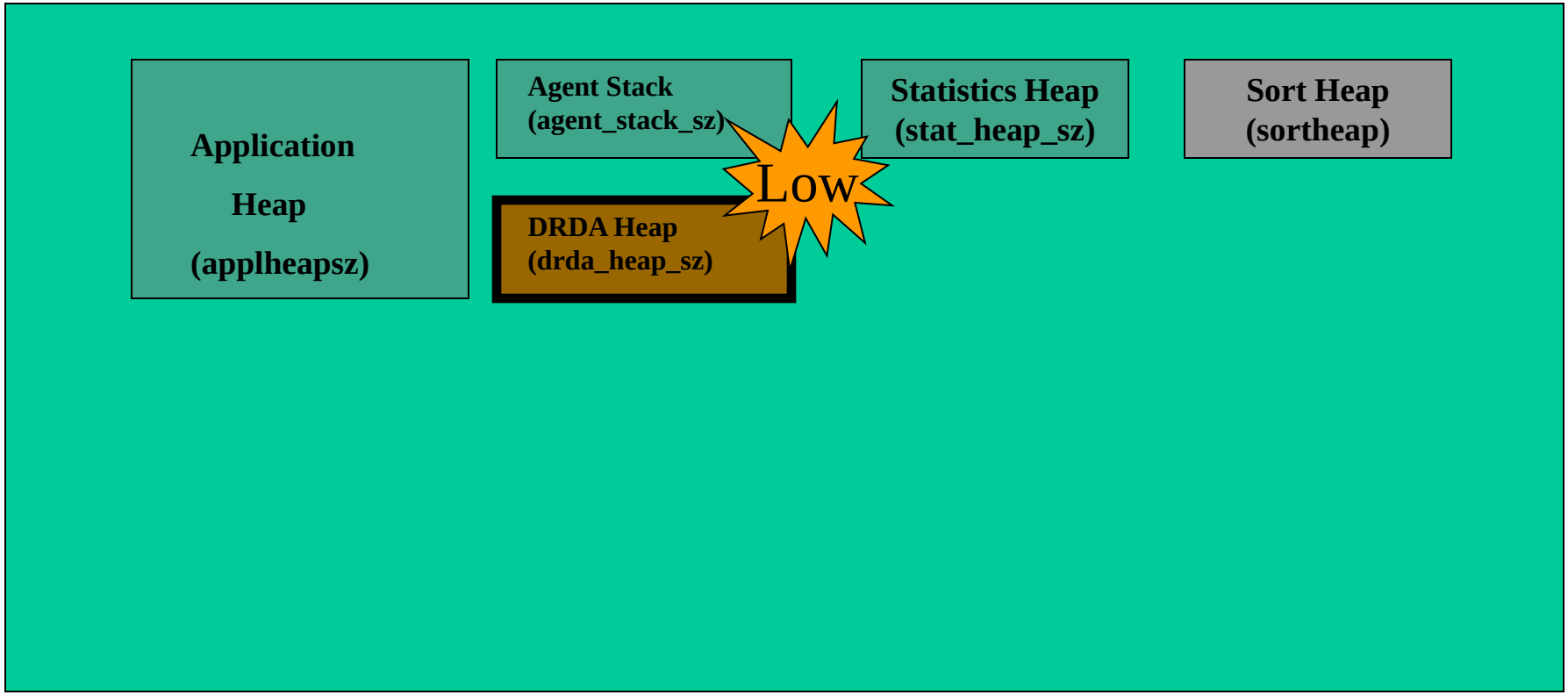
Agent Private Memory



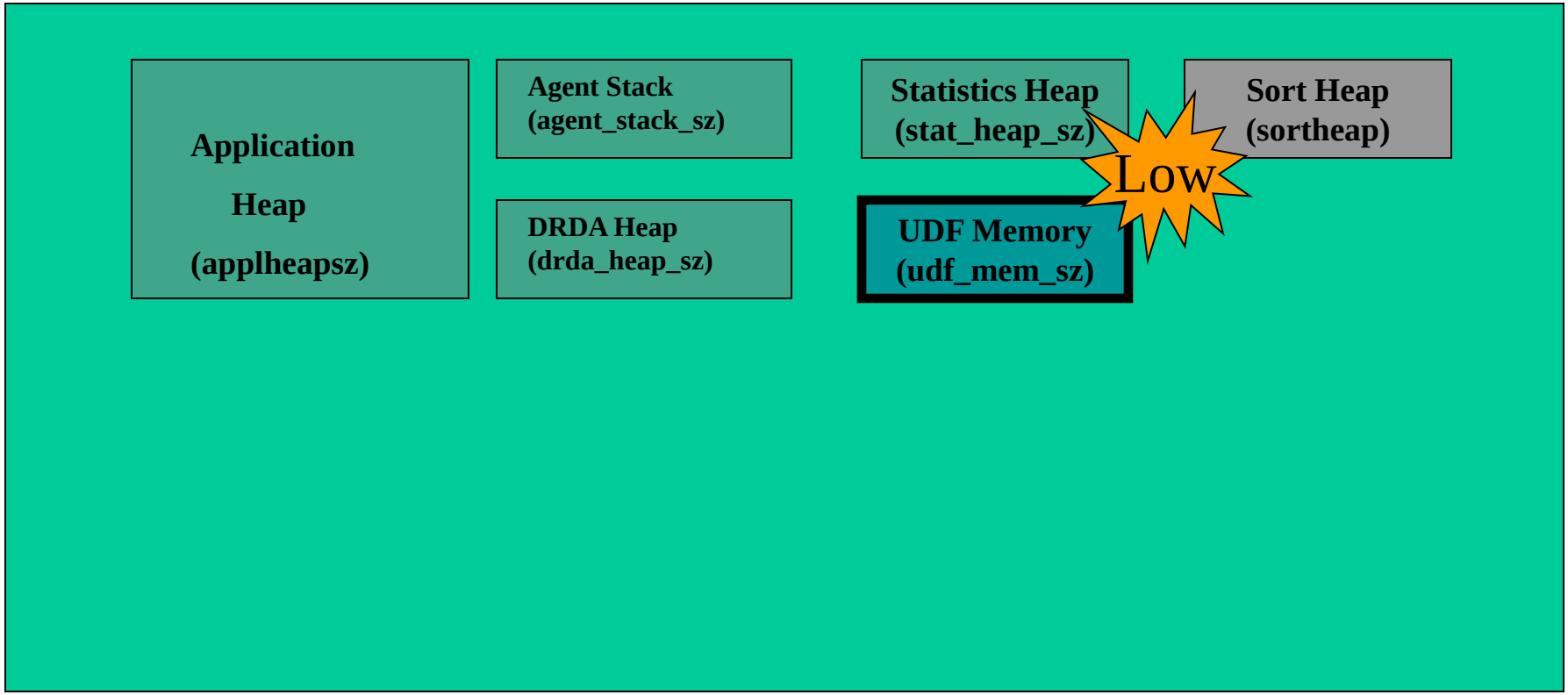
Agent Private Memory



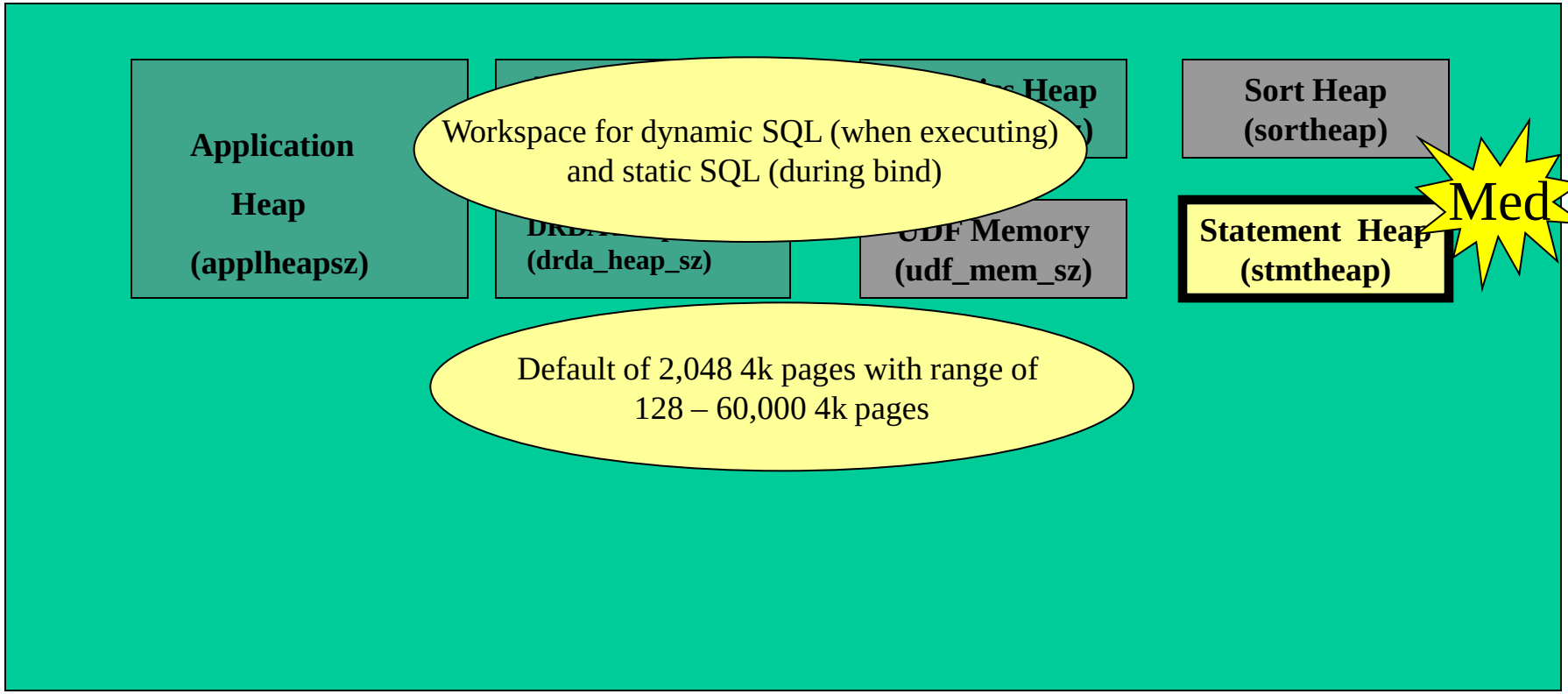
Agent Private Memory



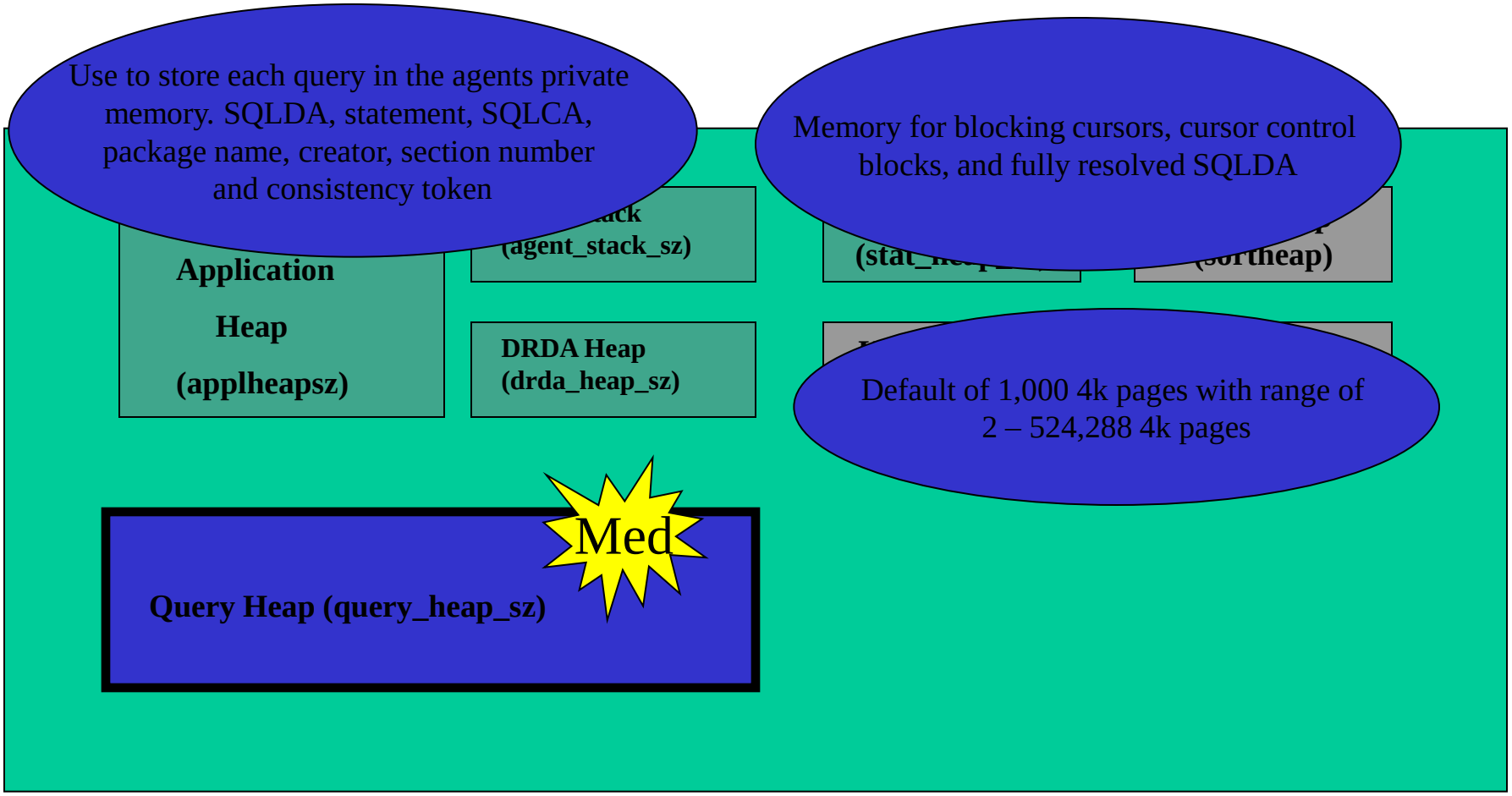
Agent Private Memory



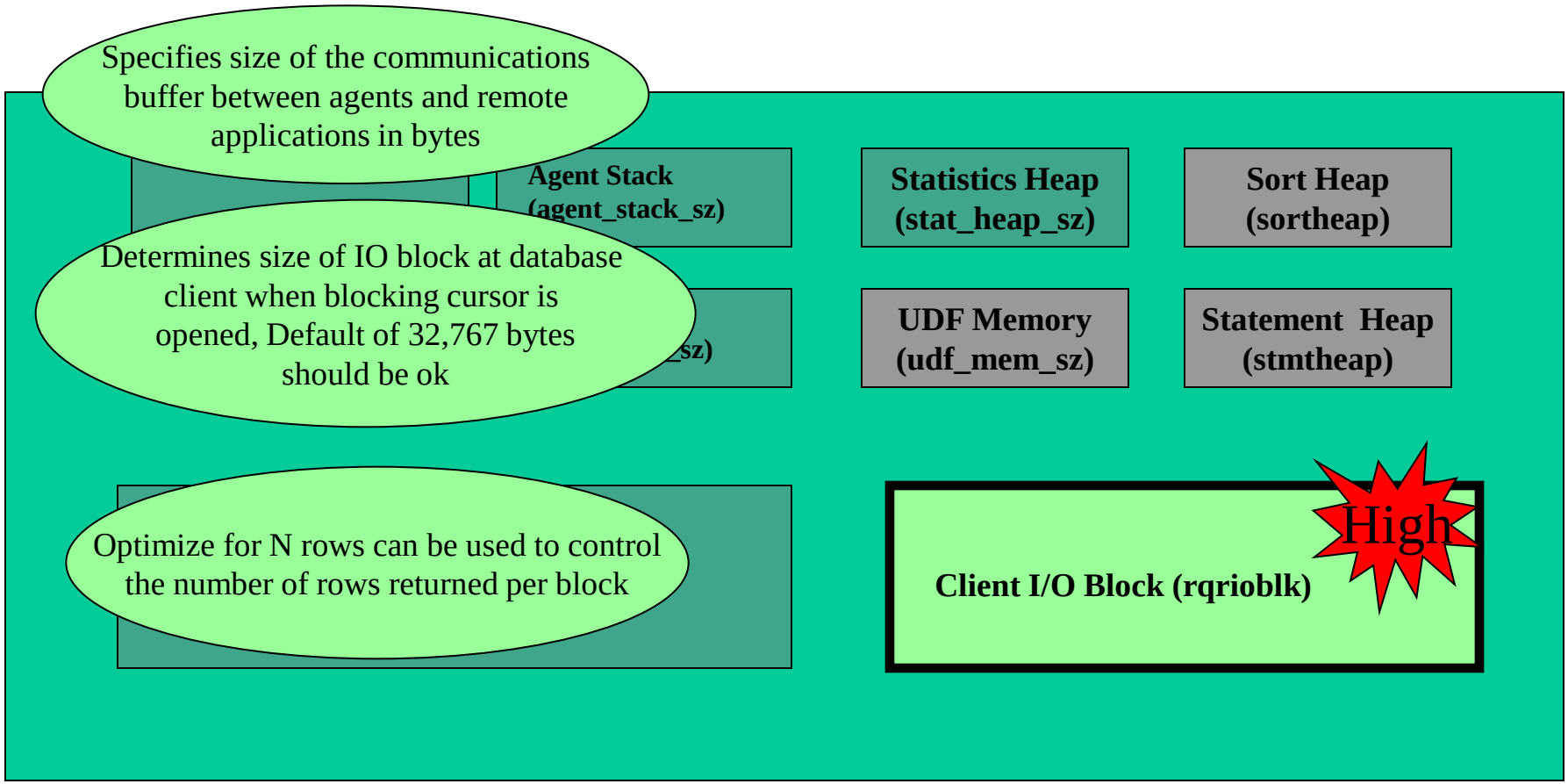
Agent Private Memory



Agent Private Memory



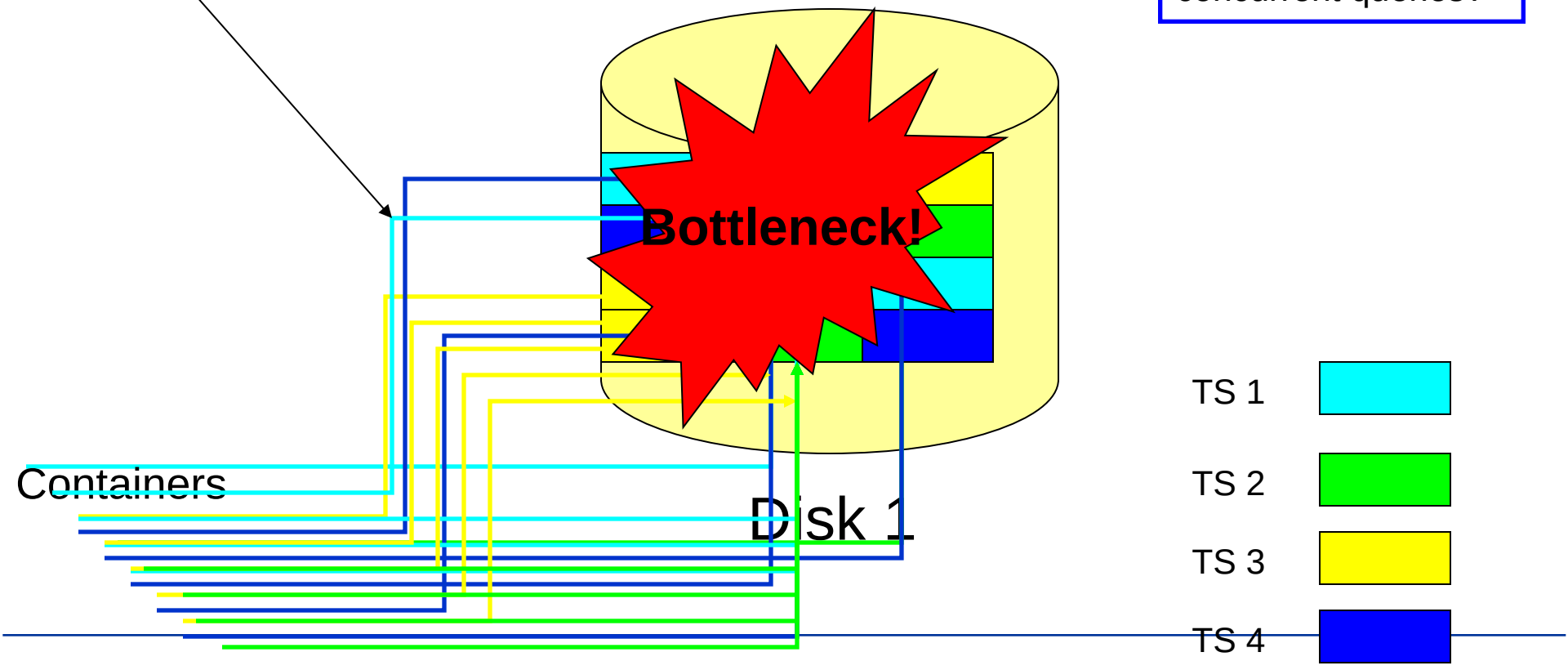
Agent Private Memory



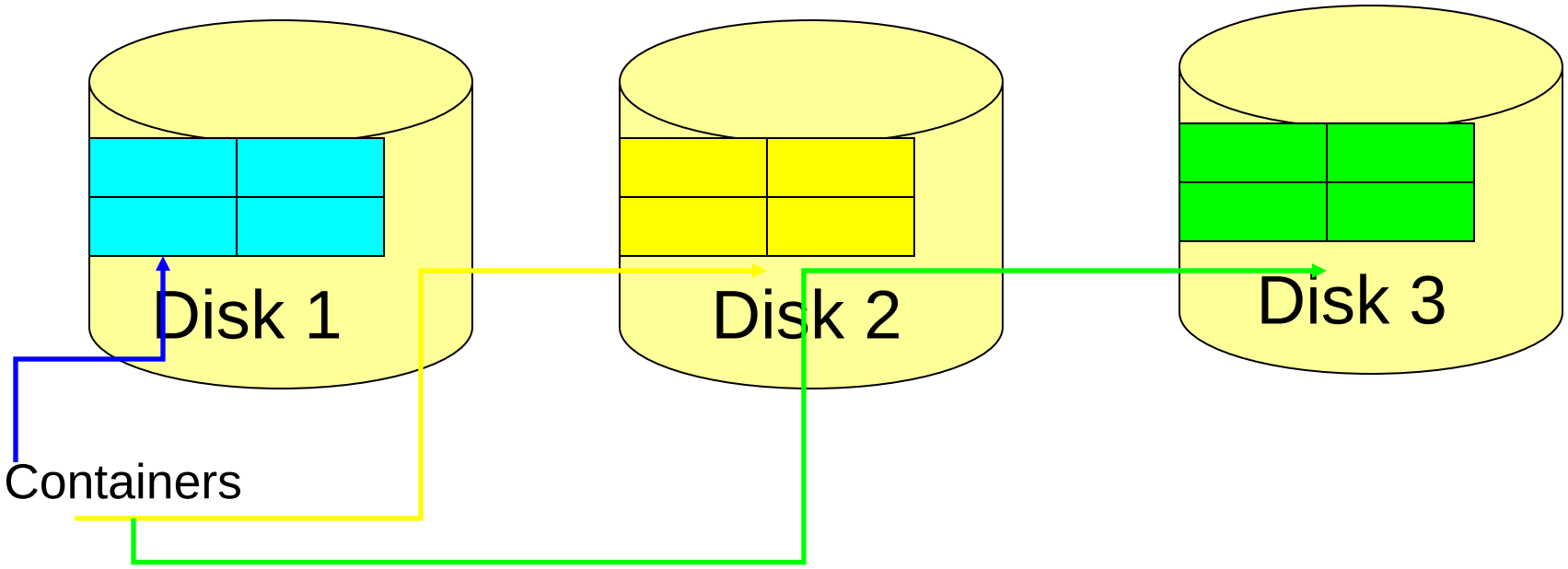
Unbalanced container placement – 4 table spaces with one container each on same physical disk

Lines depict 5 concurrent queries against each of thee table spaces.

What happens when there are 20 or 30 concurrent queries?



DB2 Striping with 1 table space with 3 containers spread over 3 disks



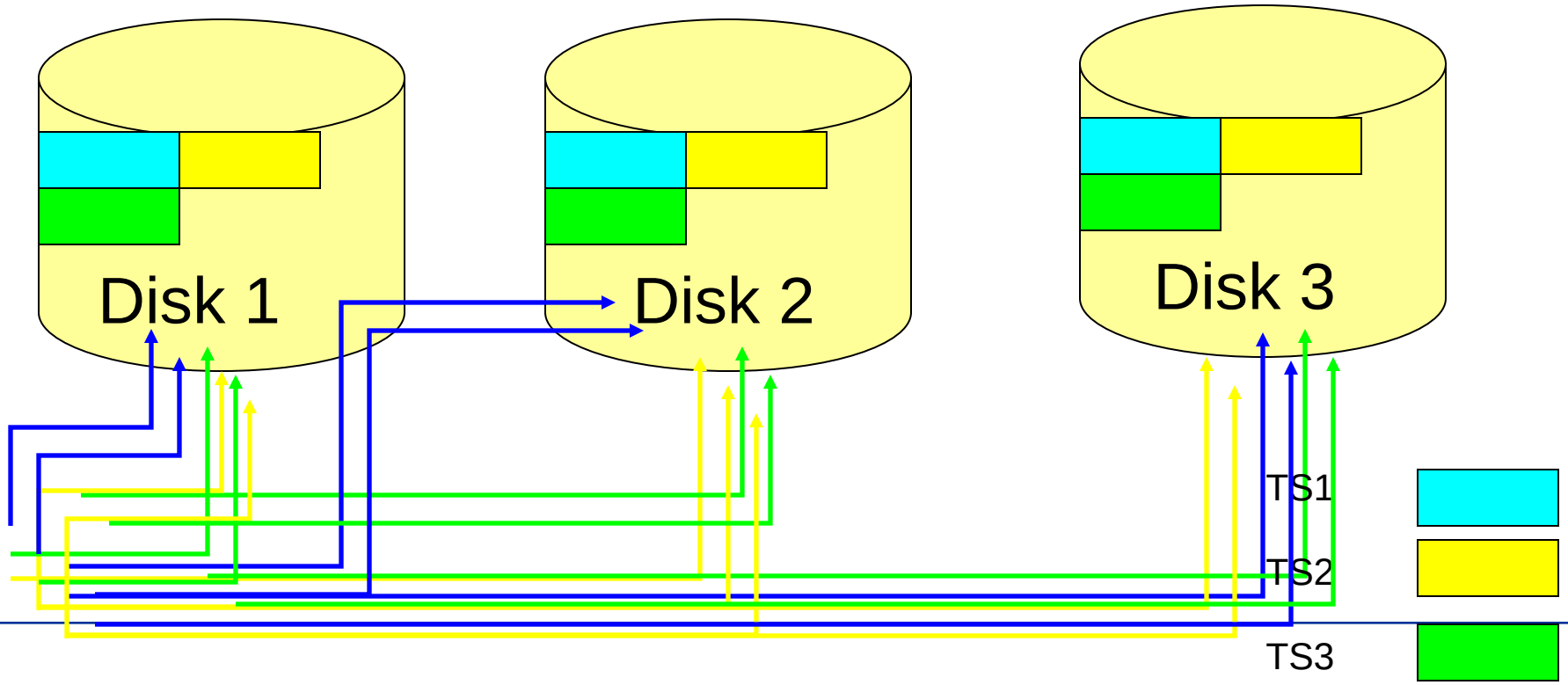
Number of Disks

- How many disks does your database need?
- This is the million dollar question!
- Too few disks is the #1 cause of poor performance
- Good rule of thumb is 6-10 disks per CPU (20 or more for DW)
- Problem is no one follows this rule of thumb

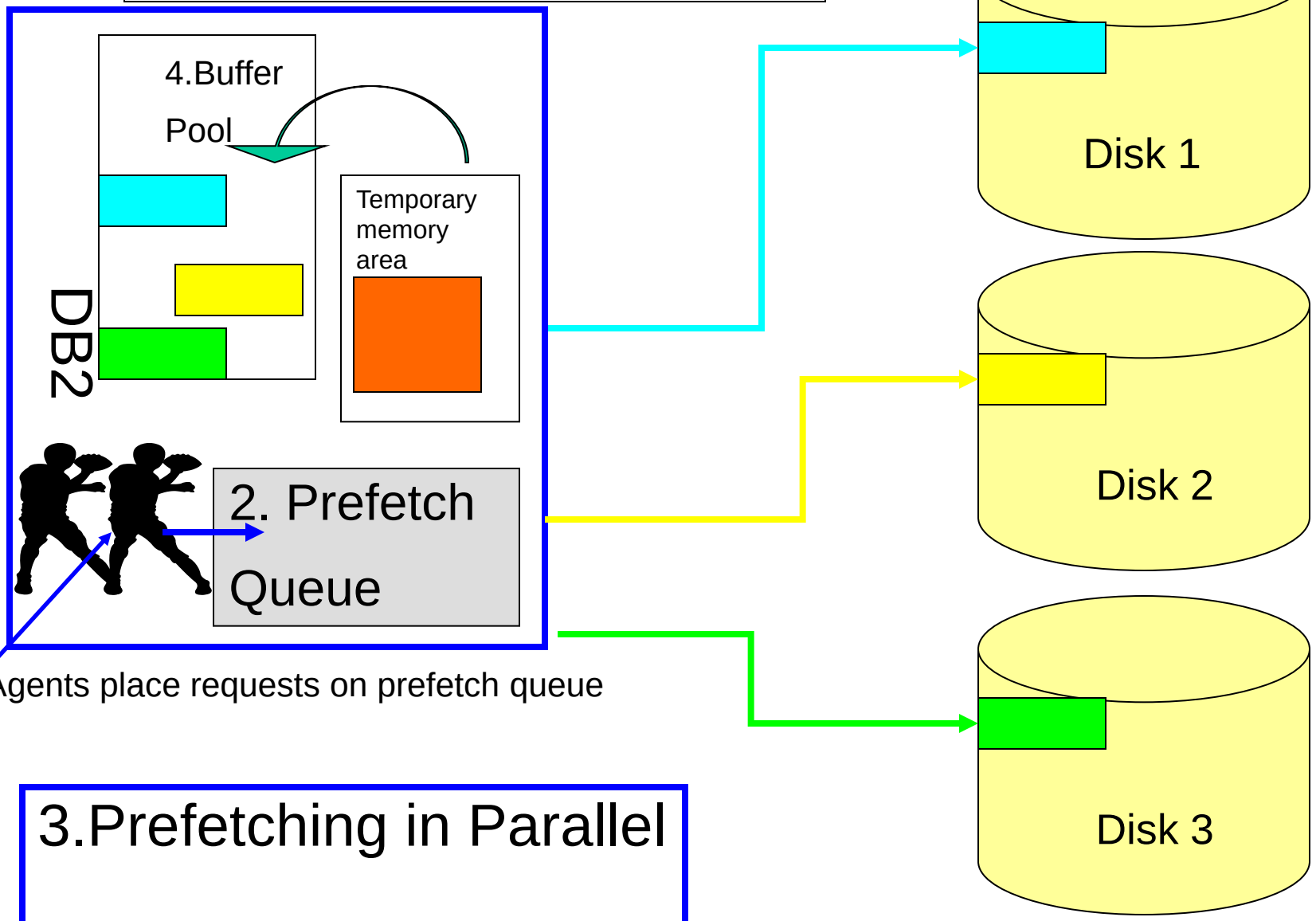
Show me the money!

DB2 Balanced Container Placement

Example: 3 Table spaces with 3 containers each with containers spread and balanced over 3 physical disks



Prefetching



3.Prefetching in Parallel

Automatic Database Storage

- Enable a database for automatic storage by specifying the AUTOMATIC STORAGE YES option on CREATE DATABASE
- Key is to specify enough storage paths which map to multiple drives/mount points
  - Use ALTER DATABASE to add additional storage paths
- Only available for databases created in DB2 V8.2.2
- Cannot Alter a database to use new feature and once a database is created it cannot be changed back to non-AUTOMATIC STORAGE

Automatic Storage Paths V8.2.2

- Example command to create a database using AUTOMATIC STORAGE
- db2 create database gtstst3 on C:\path1,C:\path2, C:\path3 DBPATH on C:

  DB20000I The CREATE DATABASE command completed successfully.
- db2 create tablespace ts1 automatic storage yes

  DB20000I The command completed successfully.
- DB2 automatically created 3 containers over the defined paths: path1, path2, path3
  - Associates tablespace with one or more storage paths
  - Containers not explicitly defined

- **By default DB2 will balance containers across defined paths**

NUM_IOCLEANERS

- DB CFG Default 1 Range(0 –255)
  - Specifies number of asynchronous page cleaners for a database.
  - Write changed pages from Bufferpool to disk
  - Triggered by CHNGPGS_THRESH which specifies a percentage of used pages at which asynchronous page cleaners will start writing out pages
  - Set this to the number of CPUs

NUM_IOSERVERS

- DB CFG Default 3 Range(1 –255)
  - Used to specify the number of prefetchers that work on behalf of database agents to perform prefetch IO and asynchronous IO for utilities such as backup and restore.
  - Set to the number of physical disks available

Agent Parameters

- Coordinator Agent – Each application has one which does work on its behalf and in a parallel environment distributes work to subagents
- Upon disconnect or detach from an instance the coordinating agent is freed and marked as idle if max number of pool agents not reached else it is terminated and storage freed if max number of pool agents reached
- DBM CFG parameter max_coordagents

Agent Parameters

- Maximum Number of Agents (maxagents) — specifies the maximum number of database manager agents, whether coordinating agents or subagents, available at any given time to accept application requests
- Can be used in resource constrained systems to limit memory usage

Agent Parameters

- Maximum Number of Active Applications (maxappls)
  - Specifies the maximum number of concurrent applications that can be connected to a database
  - When reached, an error is returned to the application and connection is not allowed
- Can be used to throttle applications in a resource constrained system

Agent Parameters

- Maximum Number of Concurrent Agents (maxcagents)
  - Specifies the max number of database manager coordinating agents that can be concurrently executing a database manager transaction
  - Does not limit the number of applications connected but limits the number of database manager agents that can be processed concurrently
- Can be used to throttle applications if resource constrained

Agent Parameters

- Initial Number of Agents in Pool (Num_initagents)
  - Specifies the initial number of idle agents that are created in the agent pool at DB2START
- By specifying a value, agents are available in the pool for initial requests and the overhead of repeated agent creation is avoided

Agent Parameters

- Agent Pool Size (num_poolagents)
  - Specifies how large the agent pool can get
  - Contains subagents and idle agents
  - Idle agents can be used as coordinating agents or subagents
  - If more agents created than this parameter they will be terminated when the current request is completed rather than returned to the pool

Dynamic Configuration Parameters

- Deferred
  - Get DBM or DB CFG show detail
  - db2pd
- Immediate
- Transaction boundary

Monitoring Essentials

- Snapshot Monitoring Essentials
- SQL Tuning
- Locking Considerations
- Table space container configuration and placement

Routine Monitoring

- Consists of online real-time monitoring
  - Classic snapshot functions
  - Insert to table SQL snapshot functions
  - New SYSCATV82 release specific views
  - db2pd command line tool
- Create snapshot repository for online real-time analysis and for historical performance, problem determination and trending purposes

Routine Monitoring

- Automatic online real-time monitoring with automated analysis of key performance metrics
  - Both ad hoc and scripted and collected and computed every 30 minutes, every hour, for every day
- OS level monitoring at same interval
  - IOSTAT
  - VMSTAT, PERFMON
  - TOP
  - SAR

Exception-based Monitoring

- Consists of unattended agent-based monitoring in the background
  - Monitors pre-defined threshold breaches
  - Alerts DBA's/Operations staff via email, text message, page, etc
  - Can take corrective action via script
- After DBA's are alerted, they use online real-time (point based) monitoring to drill down to the problem
- Exception based monitoring runs 24x7 without regard to database environment

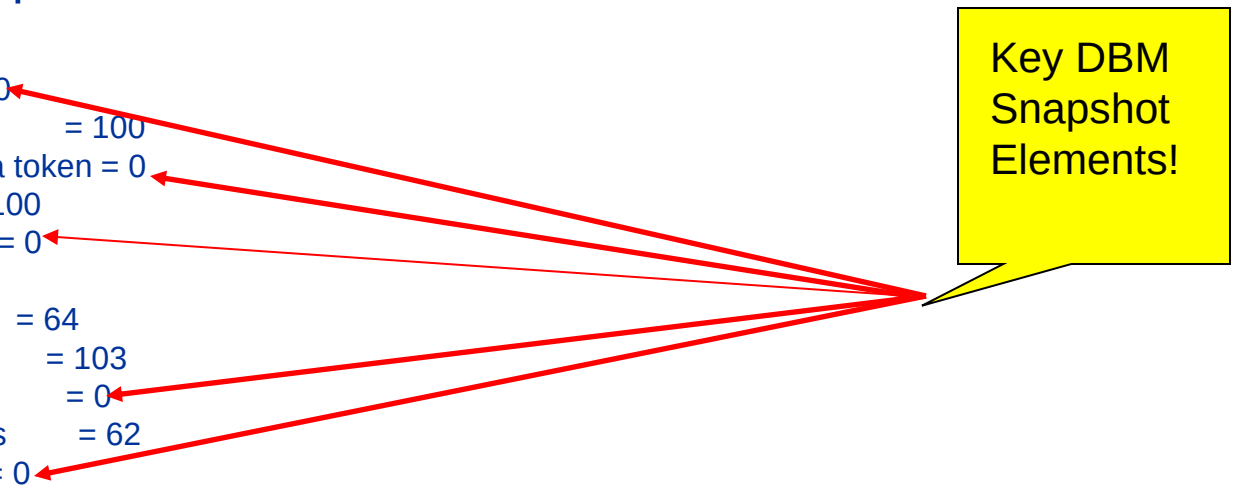
Essential Snapshots

- Database Manager
- Database
- Bufferpool*
- Tablespace
- Application*
- Dynamic SQL
- Table
- Locks*

Key Database Manager Snapshot Elements

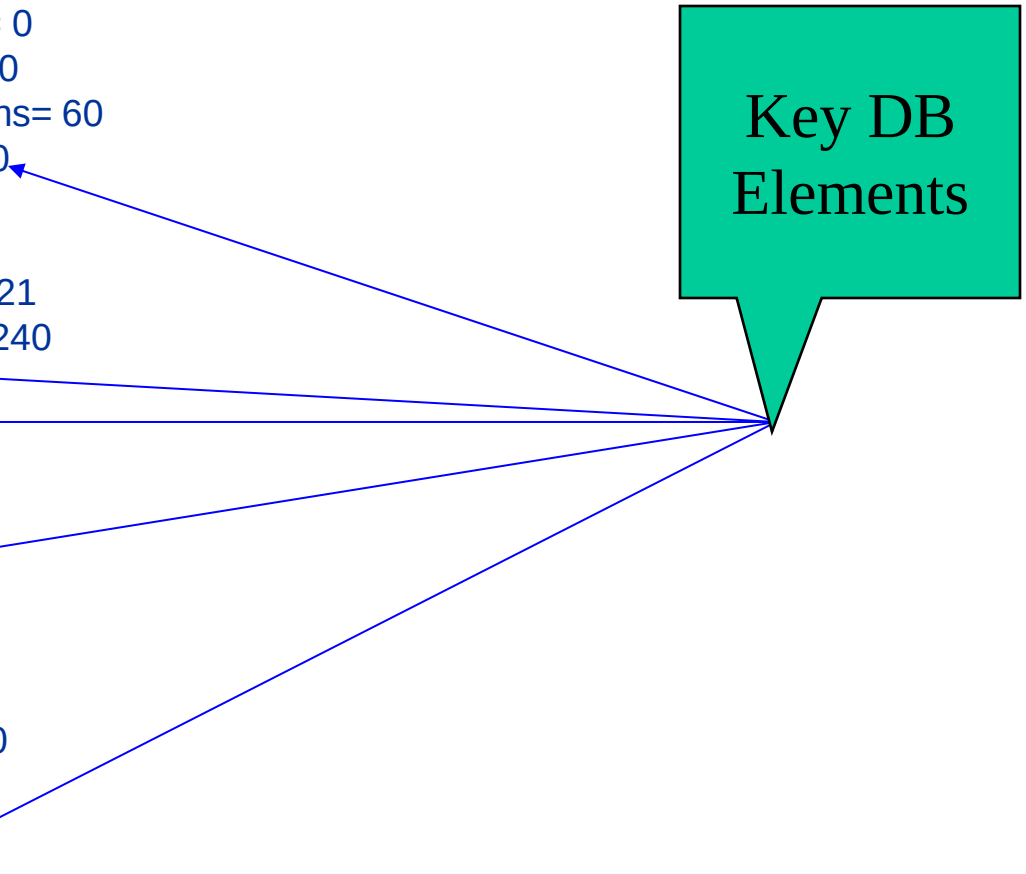
- Post threshold sorts
  - Occur when sheapthres has been reached
- Agent elements

Post threshold sorts = 0
High water mark for agents registered = 100
High water mark for agents waiting for a token = 0
Agents registered = 100
Agents waiting for a token = 0
Idle agents = 39
Agents assigned from pool = 64
Agents created from empty pool = 103
Agents stolen from another application = 0
High water mark for coordinating agents = 62
Max agents overflow = 0
Hash joins after heap threshold exceeded = 0



Key Database Snapshot Elements

High water mark for connections = 60
Applications connected currently = 60
Appls. executing in db manager currently = 0
Agents associated with applications = 60
Maximum agents associated with applications= 60
Maximum coordinating agents = 60
Locks held currently = 0
Lock waits = 1
Time database waited on locks (ms) = 21
Lock list memory in use (Bytes) = 55240
Deadlocks detected = 0
Lock escalations = 0
Exclusive lock escalations = 0
Agents currently waiting on locks = 0
Lock Timeouts = 0
Number of indoubt transactions = 0
Total Private Sort heap allocated = 0
Total Shared Sort heap allocated = 0
Shared Sort heap high water mark = 0
Total sorts = 63
Total sort time (ms) = 1959
Sort overflows = 15
Active sorts = 0



Sort Overflows

- Sort overflows occur when sorts cannot complete in sortheap
- Sort overflows and other sort problems can be monitored via the following snapshot monitoring elements:
  - Post threshold sorts (DBM)
  - Pipe sorts accepted/rejected (DBM)
  - Sort overflows (DB)
  - Sort time

Eliminate Sort Overflows

- Sort overflows can be eliminated (or at least controlled) through proper index design

- Create indexes defined on columns in ORDER BY sequence
- Create indexes defined with “Allow Reverse Scans”

Sort Best Practices

- Use explain to evaluated amount of sortheap required and to determine if defined sortheap is adequate
- Only increase sortheap after reviewing sortheap requirements
- Eliminate sort overflows (OLTP) via proper in index design
  - Eliminate sort overflows in OLTP/Web environment
  - Keep sort overflows <3% in mixed environments
  - Keep sort overflows < 10-15% in data warehouse environments

Suboptimal SQL

- Biggest problems are:
  - Poorly written SQL
  - Vendor packages with poor indexing
  - Lack of understanding of DB2 predicates
  - Improper index design
  - Lack of SQL reviews and little use of Explain during development process
- Problems are pervasive in both large and small companies

SQL Coding Best Practices

- Use Explain or third party vendor tools to review and tune SQL during development and on an ongoing basis

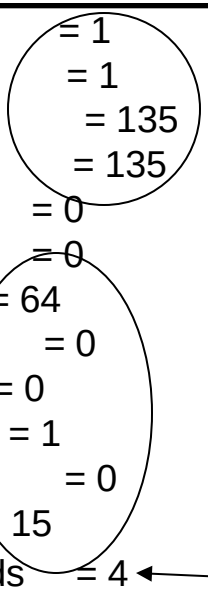
- Understand DB2 predicate rules
  - Use Range Delimiting and Index Sargable predicates whenever possible

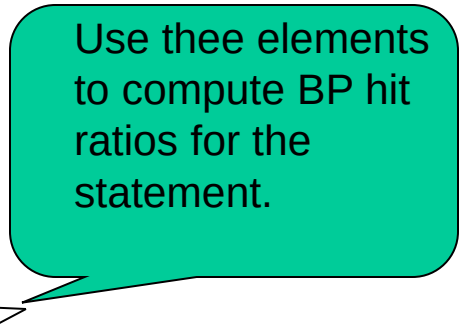
Key Dynamic SQL Snapshot Monitoring Elements

- Take Dynamic SQL and Application snapshots and search for suboptimal SQL indicators
- Look for
  - High CPU usage
  - High ratio of rows read vs rows selected
  - Sort overflows
  - Sorts
  - Use classic SQL snapshots, SQL snapshot functions or new SYSCATV82 views
- Use event monitoring if necessary for hard to find suboptimal SQL

Dynamic SQL Snapshot

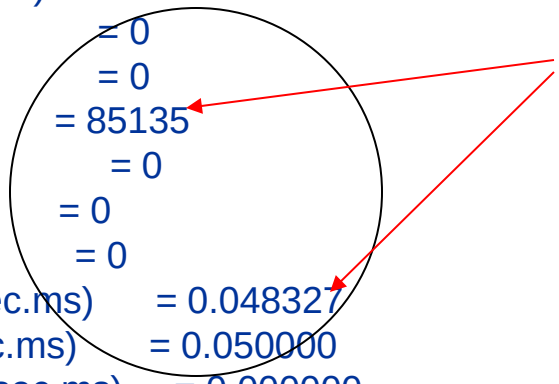
Number of executions = 1
Number of compilations = 1
Worst preparation time (ms) = 135
Best preparation time (ms) = 135
Internal rows deleted = 0
Internal rows inserted = 0
Rows read = 64
Internal rows updated = 0
Rows written = 0
Statement sorts = 1
Statement sort overflows = 0
Total sort time = 15
Buffer pool data logical reads = 4
Buffer pool data physical reads = 1
Buffer pool temporary data logical reads = 0
Buffer pool temporary data physical reads = 0
Buffer pool index logical reads = 0
Buffer pool index physical reads = 0
Buffer pool temporary index logical reads = 0
Buffer pool temporary index physical reads = 0
Total execution time (sec.ms) = 0.063596
Total user cpu time (sec.ms) = 0.000000
Total system cpu time (sec.ms) = 0.000000





Dynamic SQL Snapshot

Number of executions = 1
Number of compilations = 1
Worst preparation time (ms) = 1
Best preparation time (ms) = 1
Internal rows deleted = 0
Internal rows inserted = 0
Rows read = 85135
Internal rows updated = 0
Rows written = 0
Statement sorts = 0
Total execution time (sec.ms) = 0.048327
Total user cpu time (sec.ms) = 0.050000
Total system cpu time (sec.ms) = 0.000000
Statement text = SELECT OID_ID FROM T_OID WHERE DATE_CREATED = '2003-04-22-17.46.30.746521'



Common Lock-related Problems

- LOCKTIMEOUT (DB) set to -1
- LOCKLIST (DB) set too small
- MAXLOCKS (DB) parameter set too high

- Lock full conditions hard to detect and not well understood

Locking Best Practices

- Set LOCKTIMEOUT to a value other than -1
  - For OLTP good starting point is 30 seconds
  - Test applications and adjust but do not set too high
- Set LOCKLIST so that 50% of LOCKLIST is unused under normal workloads
  - This can prevent locklist full conditions from occurring which cause unnecessary lock escalations
- In DW, when queries will scan most of table, consider using lock table in exclusive mode

Hash Join Best Practices

- Monitor via database and application snapshots
- Monitor hash join overflows via the following monitor elements:

Number of hash joins = 768
Number of hash loops = 3
Number of hash join overflows = 3
Number of small hash join overflows = 128

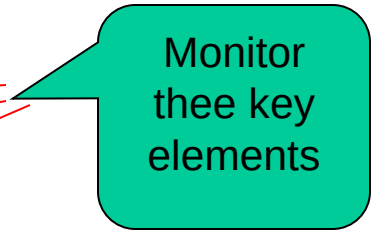
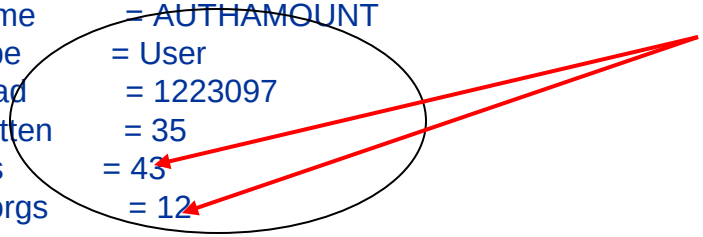


Table Snapshot

- Table snapshot example:

```
Table Schema       = PGUN
Table Name         = AUTHAMOUNT
Table Type         = User
Rows Read          = 1223097
Rows Written       = 35
Overflows          = 43
Page Reorgs        = 12

Table Schema       = <323><PGUNX  >
Table Name         = TEMP (00003,00009)
Table Type         = Dropped
Rows Read          = 9
Rows Written       = 1
Overflows          = 0
Page Reorgs        = 0
```

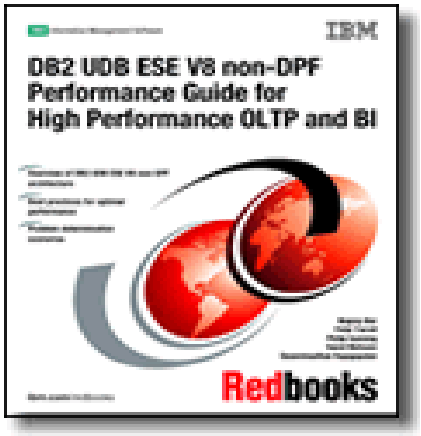


Key metrics

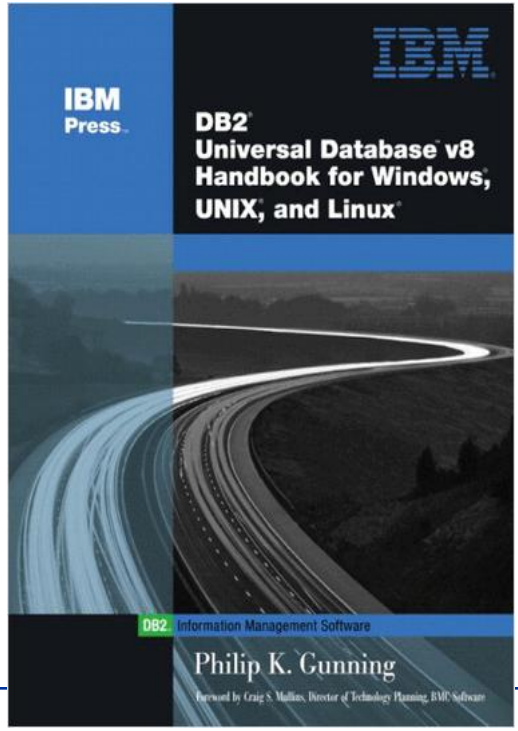
Conclusion

- Successful system tuning requires knowledge of:
  - DB2 processing
  - Available monitoring facilities
  - Instance Configuration Parameters
  - Database Configuration Parameters
  - Cause and Effect of parameters to processing
- *Available References

Thank you!



Phil Gunning
GunningTechnology Solutions, LLC
pgunning@gunningts.com
888-241-1070



CODUG DB2 User Group