TUNING DATA CACHE IN ASE 15

SYBASE
An SAP Company

DIAL NUMBERS:
1-866-803-2143
1-210-795-1098
PASSCODE: SYBASE

TUNING DATA CACHE IN ASE 15

Your host…

Terry Orsborn

Product Marketing Manager

Our speaker today…

Jeff Tallman

Sr. SW Engineer/Architect

HOUSEKEEPING

Questions?

Submit via the ‘Questions’ tool on your Live Meeting console, or call **1-866-803-2143** United States
001-866-888-0267 Mexico
0800-777-0476 Argentina
01800-9-156448 Colombia

Password **SYBASE**
Press *** 1** during the Q&A segment

Presentation copies?

Select the printer icon on the Live Meeting console

TUNING DATA CACHE IN ASE 15

Jeff Tallman
Senior Staff SW Engineer/Architect

December 1, 2010

TUNING DATA CACHE IN ASE 15

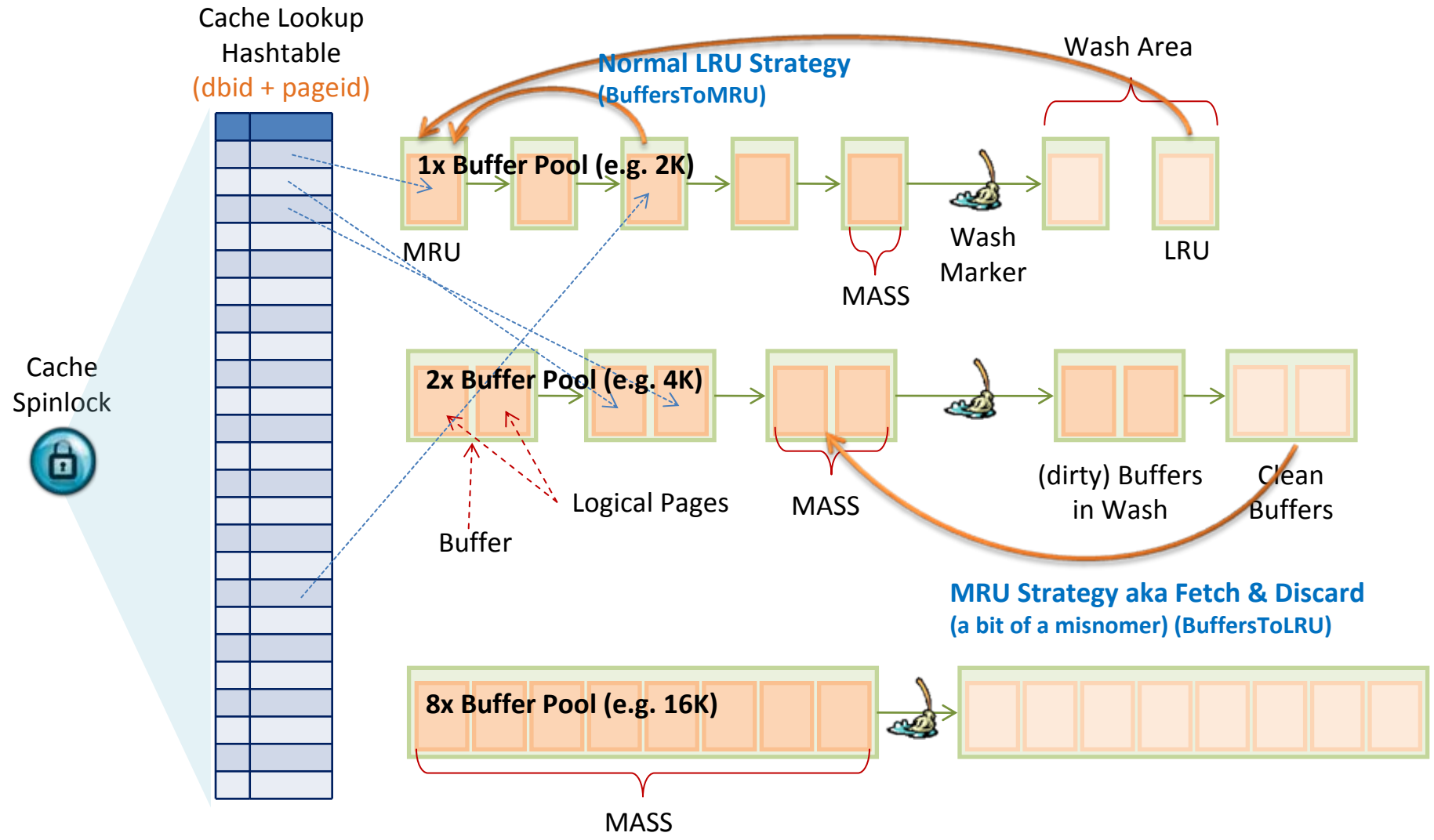
Today's Agenda

- **Data Cache Internals Review**
  - Buffer Sizes
  - Cache partitioning
  - Cache replacement strategies
- **Recommended Cache Configurations**
  - Transaction log caches
- **Using Monitoring Data to Configure Caches**
- **Procedure Cache Tuning**

SPINLOCKS & HASH BUCKETS

- **Locating items in memory is usually done via a hash table**
  - Size of the hash table may or may not be configurable
  - Each entry in the hash table is considered a hash bucket
  - Item attribute is hashed according to some hash function
  - Hash value determines which hash bucket is used
  - Hash bucket likely covers more than one value
  - Result is a serial scan through the hash chain associated with hash bucket
- **Modifying the hash chain requires grabbing the spinlock**
  - Need to find which spinlock guards that hash bucket
  - Grab the spinlock, add/remove item, release spinlock
- **Reading the hash chain may require grabbing the spinlock**
  - If you want a consistent picture vs. a 'dirty read'
- **Tuning ASE often is tuning hash buckets/spinlocks**
  - Understanding what you can change and what you can't for each
  - Understanding how to control concurrency
    - Upper limit of contention = concurrent processes = engines online

CACHE MANAGEMENT (DEFAULT)



UseMRU = PoolSize * 0.5 >= PoolSize - PagesScanned
(when the pages to be scanned would use more than 50% of the configured PoolSize, the MRU strategy can be chosen by the optimizer for pages ***read from disk***)

CACHE MANAGEMENT

Key Things to Remember When Monitoring the System

- **Changes to what is in cache is reflected in cache hash table**
  - The cache hash table is part of the cache overhead when sizing a cache
  - Changes to the cache hash table require grabbing the cache spinlock
    - Essentially, every physical read and every new page allocated (including page splits)
    - Every new #temp table created
- **All Physical I/O's are done in MASS units**
  - A large I/O can be 4K, 8K or 16K
  - A MASS is just a group of contiguous pages on disk read in during a single IO operation
  - When writing, all pages are written - whether dirty or not
  - To block further changes when writing (DMA access), rather than using a spinlock, each cache buffer has a "MASS bit"
    - Rather than spinning - you sleep…..spinlock = spin mutex; mass bit = sleep mutex
- **Logical Reads count as a cache hit**
  - In a strict (default) cache replacement, this results in the buffer being relinked to the MRU end of the MRU→LRU chain
  - Any changes (relinkage) to the buffer chain requires grabbing the cache spinlock
    - Essentially any logical read will result in a buffer relink which means spinlock grab

DO THE FOLLOWING ALL USE LARGE I/O??

Pop Quiz:

A. Large scans such as table scans, range scans & index leaf scans
B. Maintenance operations such as dbcc, update statistics
C. Create index
D. Bcp in on heap tables
E. Select/into
F. Insert/select
G. Asynchronous Prefetch

DO THE FOLLOWING ALL USE LARGE I/O??

Pop Quiz Answers

A. **Large scans such as table scans, range scans & index leaf scans (Yes - but *only* if not in cache)**

B. **Maintenance operations such as dbcc, update statistics (Yes)**

C. **Create index (Yes)**

D. **Bcp in on heap tables (Yes)**
   - Note that bcp out is implemented as a select, so likely (A)

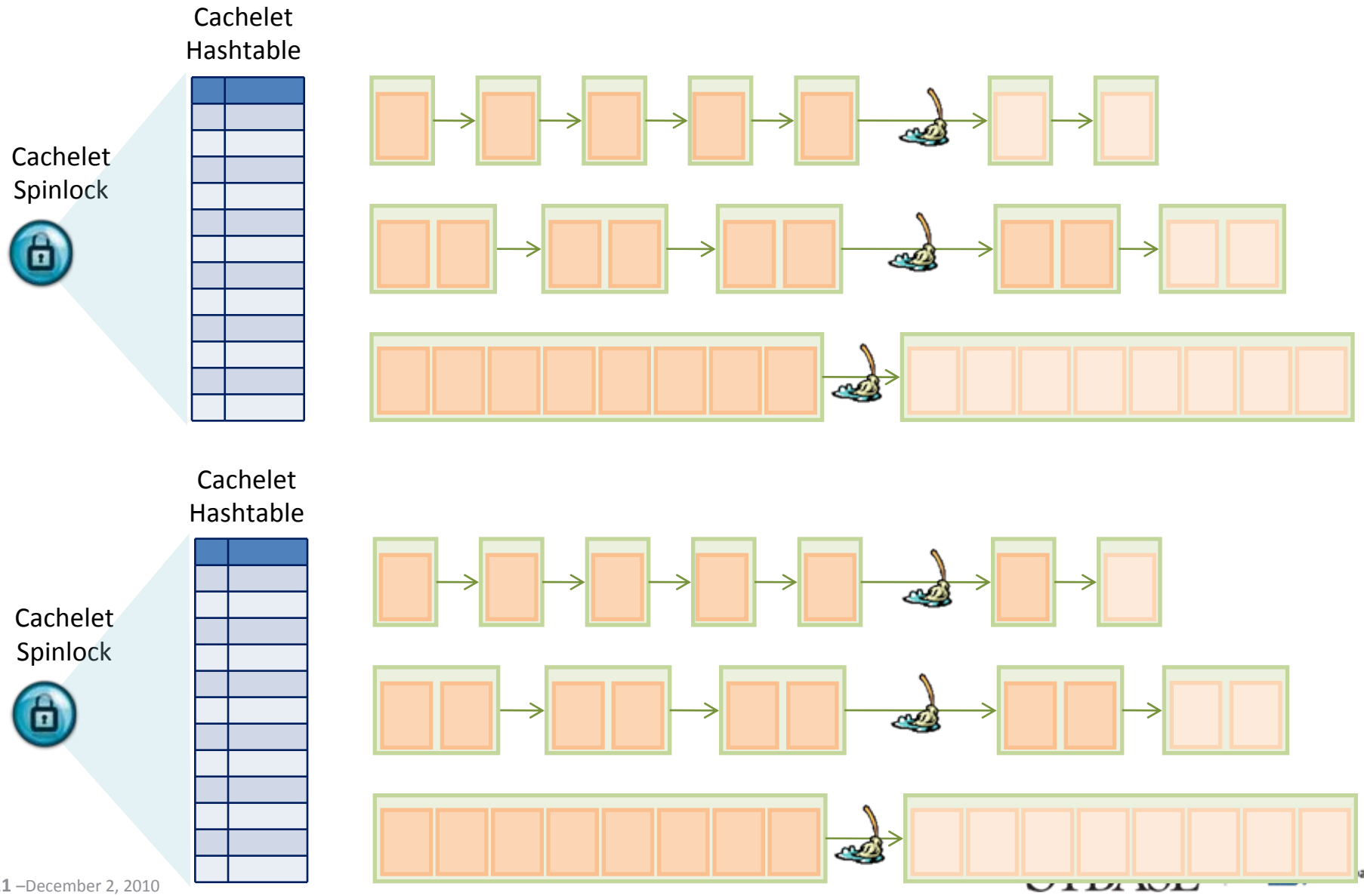
E. **Select/into (Yes)**
   - Even though in cache, the pages are allocated out of the large IO pool

F. **Insert/select (No)**
   - Even if the table is a heap, new pages are allocated out of the pagesize pool vs. the large IO pool

G. **Asynchronous Prefetch (Unrelated)**
   - APF is a method of scheduling IO - does not make a determination of the IO size. Consequently, it depends on the operation.

CACHE PARTITIONING

One Possible Answer to Cache Spinlock Contention



PARTITION IMPACT ON WASH SIZE

Default wash marker = 20% of pool size for pools < 300MB; 60MB for pools > 300MB

Partitions	350MB 2K	50MB 4K	100MB 16K	Comments
1	290+60MB	40+10MB	70+20MB	
2	140+35MB	20+5MB	40+10MB	
4	70+17.5MB	10+2.5MB	20+5MB	
8	35+8.7MB	5+1.2MB	10+2.5MB	
16	17.5+4.3MB	2.5+0.6MB	5+1.2MB	
32	8.7+2.2MB	1.2+0.3MB	2.5+0.6MB	
64	4.3+1.1MB	640+160KB	1.2+0.3MB	
	20KB	40KB	160KB	Minimum wash area size

Based on hypothetical pool sizes of a 500MB cache. Pool memory format is N+M; where N is non-wash cache and M is wash area size.

Think about it:

- Tempdb does a lot of table scans, create index and select/into's - all large IO available operations.
- A single tempdb cache with a large number of cache partitions may drive tempdb IO higher than desired for 8x pool (16K) due to smaller cachelet sizes/wash area.
- If a lot of tempdb IO, check pool activity with monCachePool or sp_sysmon and consider increasing the 8x (16K) pool size.

SP_SYSMON SAMPLE

```
Cache: default data cache
                          per sec      per xact       count  % of total
------------------------  -----------  -----------  ---------  ----------
  Spinlock Contention             n/a          n/a        n/a       1.9 %

  Utilization                     n/a          n/a        n/a      46.6 %

  Cache Searches
    Cache Hits               444812.4       2550.9   14233997      99.9 %
       Found in Wash          30218.0        173.3     966976       6.8 %
    Cache Misses                361.3          2.1      11562       0.1 %
------------------------  -----------  -----------  ----------
  Total Cache Searches       445173.7       2553.0   14245559

Cache: tempdb_cache
                          per sec      per xact       count  % of total
------------------------  -----------  -----------  ---------  ----------
  Spinlock Contention             n/a          n/a        n/a       1.5 %

  Utilization                     n/a          n/a        n/a      38.2 %

  Cache Searches
    Cache Hits               360397.8       2066.8   11532731      98.7 %
       Found in Wash           5112.2         29.3     163590       1.4 %
    Cache Misses               4680.7         26.8     149782       1.3 %
------------------------  -----------  -----------  ----------
  Total Cache Searches       365078.5       2093.6   11682513
```

Everything looks pretty much okay

Note: Remember, “utilization” reported by sp_sysmon refers to the how much of the cache searches took place in the particular cache - not how much of the cache was used. So, in the above example, there were ~150M cache searches of which 46.6% took place in the default data cache and 38.2% took place in the tempdb_cache (remaining 15.2% was in a cache not illustrated).

ACTUAL SPINLOCK CONTENTION

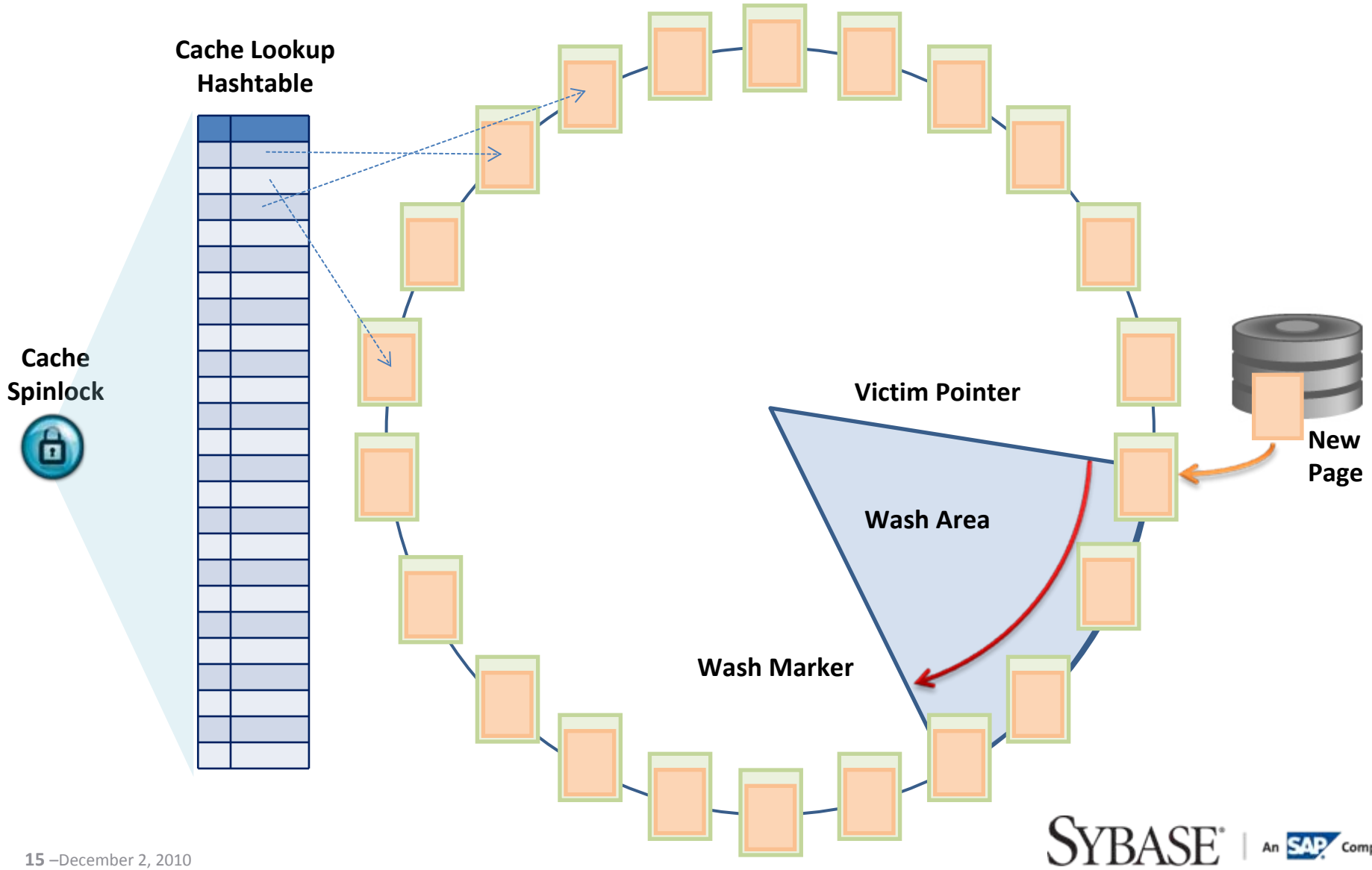
Spinlock Activity Report

Spinlock Waits	per sec	per xact	count	contention
default data cache::47	3350.9	19.2	107230	6.4 %
default data cache::55	2162.1	12.4	69186	5.0 %
default data cache::54	4626.3	26.5	148043	4.0 %
default data cache::58	1326.7	7.6	42453	3.6 %
default data cache::50	982.5	5.6	31439	3.4 %
tempdb_cache::40	593.9	3.4	19004	3.4 %
default data cache::28	726.3	4.2	23240	3.0 %
tempdb_cache::51	565.5	3.2	18096	2.7 %
tempdb_cache::32	541.8	3.1	17337	2.7 %
default data cache::27	474.8	2.7	15194	2.7 %
default data cache::3	593.6	3.4	18994	2.6 %
default data cache::26	470.5	2.7	15056	2.5 %
default data cache::21	523.6	3.0	16754	2.4 %
default data cache::45	402.2	2.3	12869	2.3 %
default data cache::18	507.1	2.9	16228	2.3 %
default data cache::17	361.4	2.1	11564	2.2 %
tempdb_cache::33	448.7	2.6	14358	2.2 %
default data cache::41	456.0	2.6	14593	2.2 %
default data cache::42	421.7	2.4	13495	2.1 %
default data cache::43	423.9	2.4	13565	2.1 %
default data cache::20	452.6	2.6	14483	2.1 %
default data cache::46	326.9	1.9	10460	2.1 %
default data cache::56	306.3	1.8	9800	2.0 %
default data cache::8	497.7	2.9	15925	1.9 %
default data cache::22	294.1	1.7	9410	1.9 %
tempdb_cache::42	285.3	1.6	9128	1.9 %
default data cache::44	229.4	1.3	7342	1.8 %
tempdb_cache::36	352.0	2.0	11264	1.8 %
tempdb_cache::45	312.6	1.8	10003	1.8 %
default data cache::36	316.6	1.8	10130	1.8 %

Whoa!!! Not so okay after all…not bad, but not good!

RELAXED CACHE REPLACEMENT

A Second Solution to Cache Spinlock Contention



RELAXED CACHE REPLACEMENT TIPS

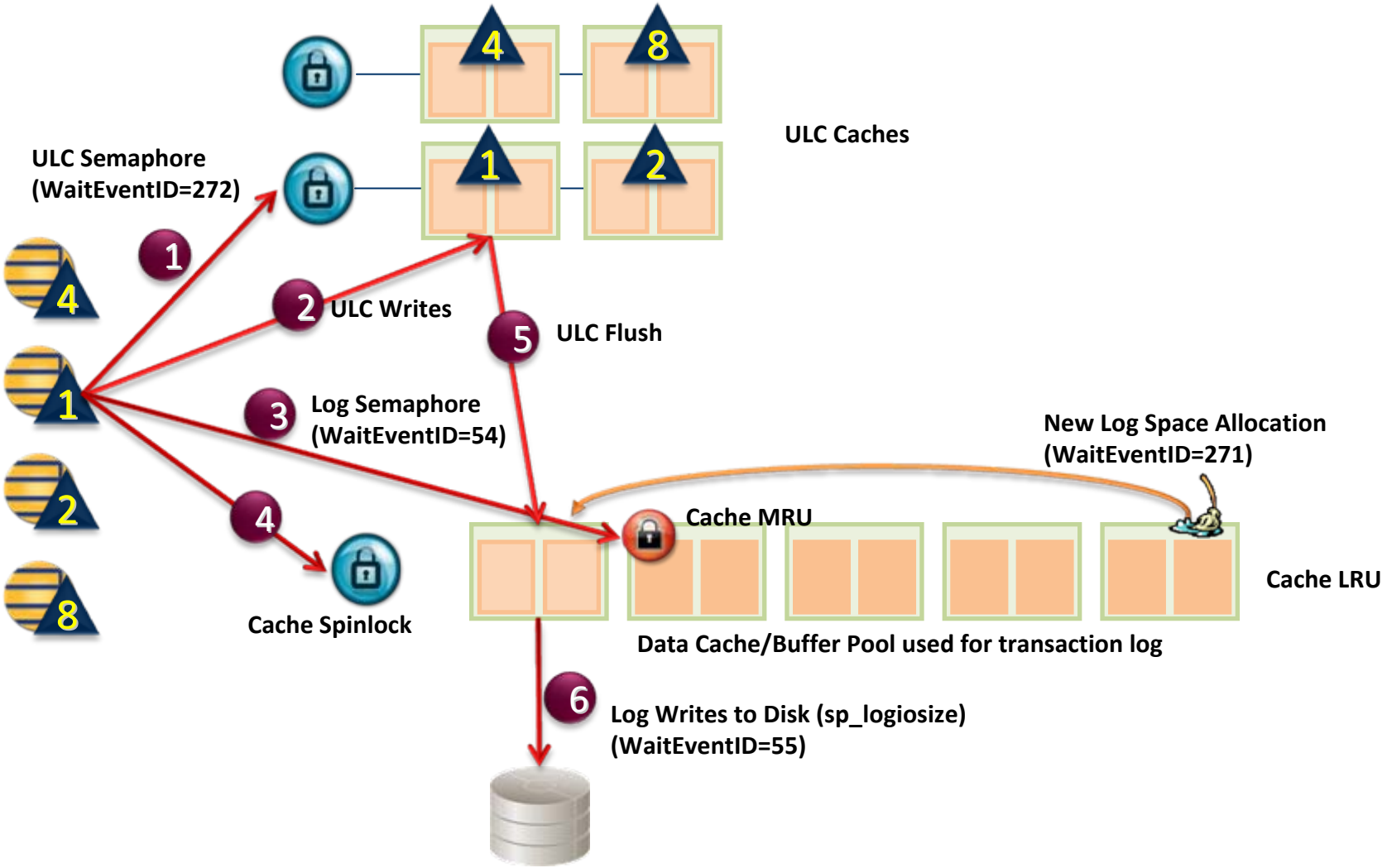
- **The trade-off:**
  - Reduces LRU→MRU relinkage driven spinlock contention
  - Increases physical IO if a lot of inserts as victim pushes wash marker around the ring
    - Wash is much more likely to hit recently modified pages since they are not moved to MRU
  - Potential increase in time it takes to find a clean page (cache stall)
  - Can decrease cache effectiveness as page cache overwrites are not dependent on how often re-read/re-written
- **Tips:**
  - Can be used for any db if db can be fully cached
    - If using multiple tempdb's, smaller OLTP tempdbs might benefit
  - Can be used for any table if the table can be fully cached and the table does not have:
    - a lot of inserts (non-ascending)
    - updates that cause page splits
    - Consider DOL tables with exp_row_size
  - A good choice for indexes if index fully cachable and low turnover
    - Consider using a smaller fill factor though to reduce new page creations due to updates to index key values
      - Index key value updates → delete followed by insert (of key values)
    - Only if spinlock contention is a concern

REDUCING SPINLOCK CONTENTION:

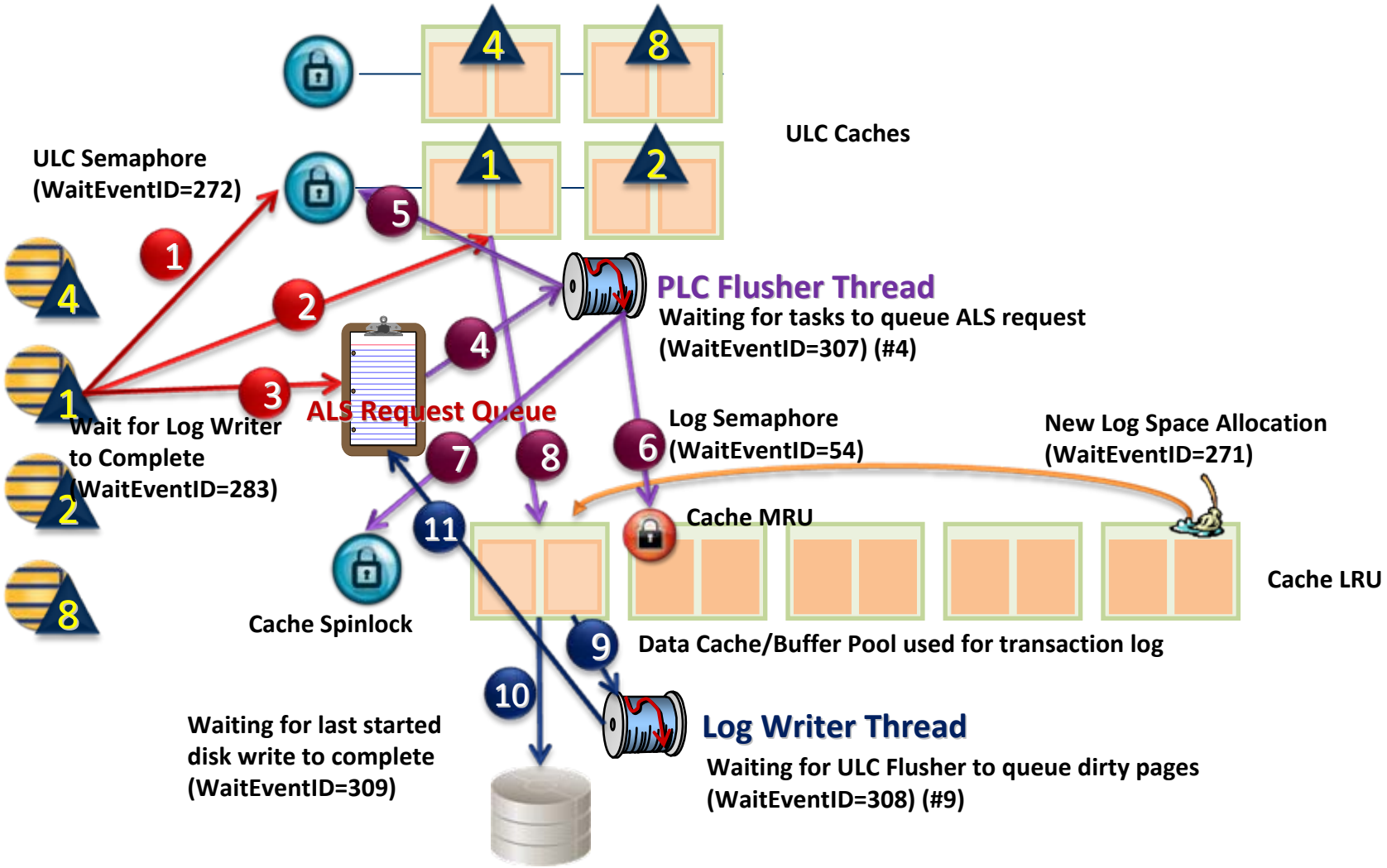
Use All 3 Choices

- **Increase number of spinlocks**
  - Decrease the spinlock ratio
  - Increase number of cache partitions (up to engine count)
- **Change cache replacement strategy**
  - Used relaxed cache replacement strategy
- **Use multiple different named caches**
  - E.g. split volatile tables and indexes into separate caches
  - Use separate caches for transaction logs
  - Use separate caches for tempdb
  - Etc.

TRANSACTION LOG CACHE



TRANSACTION LOG CACHE W/ ALS



TRANSACTION LOG IN DATA CACHE

The Need for a Dedicated Log Cache

- **Log semaphore is a lock on the last log page**
  - Therefore it moves as the log is appended to.
  - Logical lock - not a spinlock
- **Log pages in a shared data cache**
  - Log appends cause contention with cache semaphore for other cache changes
  - Cache partitioning “scatters” contiguous log allocations among partitions meaning log scans such as checkpoints, database triggers, etc. have to potentially grab multiple spinlocks during processing
  - Log appends can lead to pushing data pages out of cache
- **Other optimization considerations**
  - The HK Wash does not run in a log-only cache….therefore by having log pages in a mixed cache, the HK Wash runs unnecessarily against log pages
  - Checkpoint process will do physical reads from log and physical writes to data (it does not do physical reads from data pages)

DATA CACHE CONFIGURATION

A Recommended Starting Configuration

- **Too few DBA's configure data cache correctly**
  - Almost everything is in default data cache
  - There may be a log cache - usually oversized
  - There may be tempdb cache
- **A good starting configuration should minimally include:**

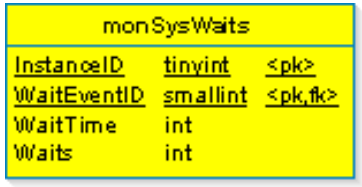
Named Cache	Sizing	Number	Partition	Cache Strategies	HK Ignore
Log cache	50-100MB (normal) 150-200MB (XXL SMP)	1-3	No	Log only, Relaxed	(implicit)
System tables	200MB-500MB	1	No or few	Relaxed	(implicit)
Tempdb caches	250MB-500MB (normal) 500MB-1GB (XXL SMP)	1 per tempdb/ tempdb group?	YES*	HK Ignore Cache or Relaxed if ~100% cached	YES
Restriction	50-100MB (normal) 256-500MB (BLOB)	1	YES*	Strict (default)	(maybe)
Reference	50-100MB	1	No or few	Relaxed	(implicit)
Hot Tables (static size – update intensive) such as key sequence tables	10-50 MB	1	Few or more*	Relaxed	(implicit)
Hot Tables/Indexes	Size of volatile data	As necessary	Few or more*	Strict (default)	NO
Application Specific	As necessary	1-3	YES*	(depends)	NO
Default Data Cache	(most of memory)	(1)	YES*	(default)	NO

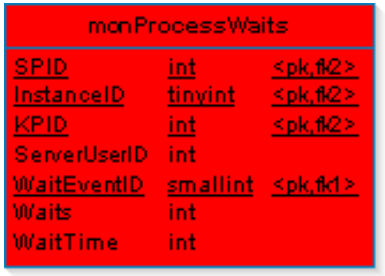
USING MONITORING DATA TO CONFIGURE CACHES

PHYSICAL WRITES

monSysWaits & monProcessWaits

- **Waits vs. WaitTime**
  - Both are important
  - "Waiting" implies process was interrupted off of the CPU - put on SLEEP queue
    - Has to wait it's turn to get back on the CPU
  - "WaitTime" is the amount of time spent waiting
    - Waiting too long is a bad thing
    - WaitTime for fast events (IO) - remember 100ms clock length
      - May take a lot of events to accurately measure
      - Slow events (locking) are more easily measured.
- **Conventional Wisdom**
  - Keep 'recovery interval in minutes' low
  - Problem: estimate is still based on 6,000 records/minute
- **MDA Quick Tip**
  - Watch Physical Writes & MASS contention
  - Use monProcessWaits/monProcessActivity
    - Separate HK wash from HK GC
    - Monitor checkpoint
  - Contention is rare, but if a lot, consider
    - increasing 'recovery interval'
    - decreasing HK free write percent
    - Changing the cache replacement strategy
  - More common
    - Checkpoint doing physical reads due to no log cache/too small (29)
    - HK writes will block data cache modifications (36)
      - Usually not a major cause, but frequently in top 10





COMMON WAIT EVENTS

Checkpoint/House Keeper Contention vs. Other SPID Contention (or Self Waiting)

Wait Event ID	Description	Common causes	PYS IO	CHK/ HK	SELF	OTH SPID
29	waiting for regular buffer read to complete	physical page read (single)	✓		✓	
30	wait to write MASS while MASS is changing	checkpoint, housekeeper is blocked by another spid changing data		✓		
31	waiting for buf write to complete before writing	blocking (synchronous) io (due to index tree maintenance/rebalancing???) (page splits in 12.5)	(✓)	?	✓	
35	waiting for buffer validation to complete	usually only seen when physical reads are swamping the system or system is cpu bound	(✓)		✓	
36	waiting for MASS to finish writing before changing	spid trying to change data is blocked by checkpoint, housekeeper or synchronous IO from another task	✓	✓		
37	wait for MASS to finish changing before changing	spid trying to change page header info is blocked by another spid modifying data in the same MASS (buffer) (pagesplits)				✓
41	wait to acquire latch	DOL index or datarows locking contention				✓
51	waiting for last i/o on MASS to complete	spid data modification waiting due to blocking (synchronous) IO (such as index tree maintenance/rebalance???) (page splits in 12.5)	✓		✓	
52	waiting for i/o on MASS initiated by another task	spid data modification waiting for physical write initiated by another task (checkpoint, housekeeper, etc.)	✓	✓		✓
53	waiting for MASS to finish changing to start i/o	hk wash, chkpt waiting for spid to finish data modification before starting write		✓		
54	waiting for write of the last log page to complete	log semaphore wait				✓
55	wait for i/o to finish after writing last log page	waiting for log flush to disk	✓		✓	
57	checkpoint process idle loop	Common to checkpoint processing (should be high)		✓		
61	hk: pause for some time	Common to all HK's (GC, Wash, Chores)		✓		
124	wait for mass read to finish when getting page	APF based physical read	✓		✓	
150	waiting for a lock	Usual blocking on page/row lock plus log semaphore or index tree rebalancing???				✓
171	waiting for CTLIB event to complete	CIS (RPC or proxy table) send or RepAgent Send event			✓	

TYPICAL MONSYSWAITS

Contention with Checkpoint & HK is pretty low

Wait EventID	WaitTime	Waits	Description
215	24,545	66,90,435,366	waiting on run queue after sleep
179	100,953	22,235,158	waiting while no network read or write is required
250	9,076,777	14,724,949	waiting for incoming network data
251	31,824	10,714,096	waiting for network send to complete
29	28,143	2,615,370	waiting for regular buffer read to complete
214	5,124	1,085,324	waiting on run queue after yield
171	1,427	1,081,419	waiting for CTLIB event to complete
51	2,694	907,929	waiting for last i/o on MASS to complete
124	5,951	667,632	wait for mass read to finish when getting page
55	1,644	586,596	wait for i/o to finish after writing last log page
52	1,190	331,100	waiting for i/o on MASS initated by another task
222	39,207	199,898	replication agent sleeping during flush
31	250	170,171	waiting for buf write to complete before writing
36	460	78,259	waiting for MASS to finish writing before changing
35	533	49,426	waiting for buffer validation to complete
178	9,371	37,270	waiting while allocating new client socket
150	3,880	23,042	waiting for a lock
54	23	3,951	waiting for write of the last log page to complete
272	32	2,689	waiting for lock on ULC
37	0	2,358	wait for MASS to finish changing before changing
41	100	1,984	wait to acquire latch
53	0	980	waiting for MASS to finish changing to start i/o

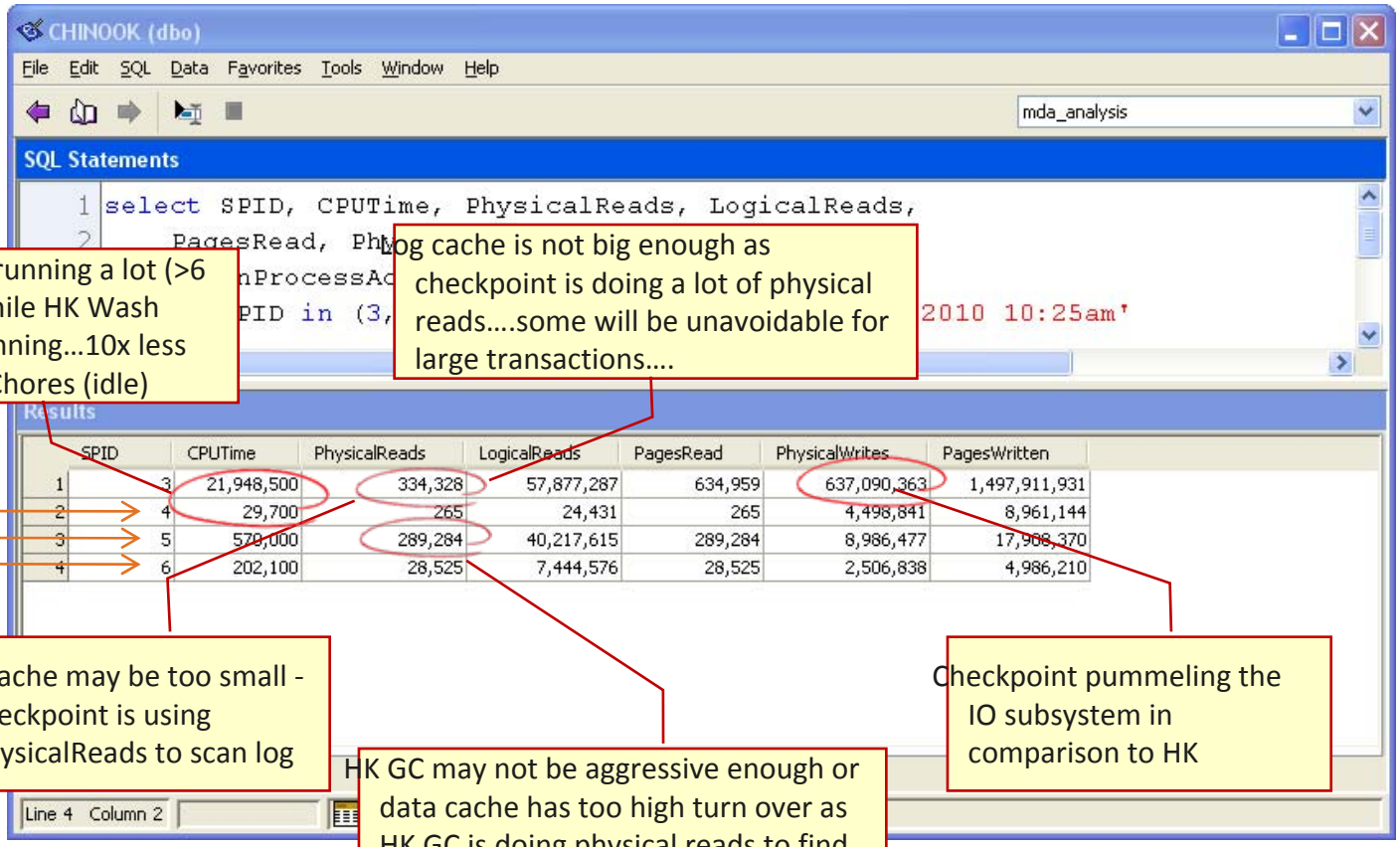
Biggest issues are network, parse/compile/optimize

Could be network if proxy tables or RPC….otherwise, it could be RepAgent

…and yet many are reluctant to change ‘housekeeper free write percent’

CHECKPOINT VS. HOUSEKEEPER TUNING

monProcessActivity Physical Reads/Writes by Checkpoint, HK Wash, GC & Chores



Checkpoint running a lot (>6 hours) while HK Wash barely running…10x less than HK Chores (idle)

Log cache is not big enough as checkpoint is doing a lot of physical reads….some will be unavoidable for large transactions….

HK Wash
HK GC
HK Chores

Log cache may be too small - checkpoint is using PhysicalReads to scan log

HK GC may not be aggressive enough or data cache has too high turn over as HK GC is doing physical reads to find pages to process deletes

Checkpoint pummeling the IO subsystem in comparison to HK

Notes:

- **Log cache may need to be configured/tuned**
- **Recovery Interval in minutes may be to high (but HK Wash needs to pick up first)**
- **Housekeeper GC needs to be made more aggressive (4 or 5)**
- **Housekeeper Free Write Percent likely could be increased (HK Wash is not keeping up vs. checkpoints)**

CACHE CONFIGURATION/SIZING:

monCachePool & monDataCache

- **Conventional Wisdom**
  - Watch cache utilization or cache hit rates
- **Common problem**
  - Most people use these (or similar metrics from sp_sysmon) as their only cache configuration/tuning aids
  - In doing so, often misinterpret cache utilization % from sp_sysmon
  - In memory table scans distort cache hit ratio
- **Limitations**
  - Doesn't help you determine if you have the correct cache configuration (number of caches)
  - Doesn't help you determine which objects should be in cache
- **Fun Facts:**
  - monCachePool.PhysicalReads will only accumulate totals from buffers which are still resident in memory so there may be discrepancies vs. monDataCache - use monDataCache for total PhysicalReads in cache
  - **monCachePool.PagesTouched is similar - reflects the number of pages *currently* in "use".**
    - This can be higher simply due to new pages created due to inserts.
    - It can be lower as pages are deallocated
    - Overtime, this is a more accurate picture of cache utilization (peak usage).
  - BuffersToMRU & BuffersToLRU seem to be measuring in pages vs. buffersize…and PagesRead = BuffersToMRU + BuffersToLRU
  - PhysicalReads vs. PagesRead should always be in IOBufferSize/@@maxpagesize units as this is the MASS size

monDataCache		
CacheID	int	<pk>
InstanceID	tinyint	<pk>
RelaxedReplacement	int	
BufferPools	int	
CacheSearches	int	
PhysicalReads	int	
LogicalReads	int	
PhysicalWrites	int	
Stalls	int	
CachePartitions	smallint	
CacheName	varchar(30)	<pk>

monCachePool		
CacheID	int	<pk,fk>
InstanceID	tinyint	<pk,fk>
IOBufferSize	int	<pk>
AllocatedKB	int	
PhysicalReads	int	
Stalls	int	
PagesTouched	int	
PagesRead	int	
BuffersToMRU	int	
BuffersToLRU	int	
CacheName	varchar(30)	<pk,fk>

MONCACHEPOOL & TEMPDB CACHE

PagesRead=PhysicalReads * (IOBufferSize/@@maxpagesize)

Results

	CacheID	InstanceID	IOBufferSize	AllocatedKB	PhysicalReads	Stalls	PagesTouched	PagesRead	BuffersToMRU	BuffersToLRU	CacheName
1	0	0	4,096	2,883,576	9,941	0	7,221	9,941	9,932	9	default data cache
2	0	0	32,768	262,144	33,096	0	64,427	264,768	33,792	230,976	default data cache
3	1	0	4,096	10,240	0	0	0	0	0	0	ec_keys_cache
4	2	0	4,096	563,200	0	0	0	0	0	0	imdb_big_cache
5	3	0	4,096	143,360	0	0	81	0	0	0	imdb_small_cache
6	4	0	4,096	51,200	0	0	0	0	0	0	log_cache
7	4	0	32,768	102,400	0	0	0	0	0	0	log_cache
8	5	0	4,096	102,400	1	0	1	1	1	0	system_cache
9	6	0	4,096	393,216	361	0	723	361	361	0	tempdb_cache
10	6	0	32,768	131,072	0	0	631	0	0	0	tempdb_cache

Results | Messages

PhysicalReads in tempdb could be system table reads or MJ spills to disk…most likely the former (system tables)

Pages added due to inserts and dropped (select/into's used 32K pool & large IO)

RecentlyUsedKB = PagesTouched*@@maxpagesize/1024

Results

	CacheID	InstanceID	IOBufferSize	AllocatedKB	PhysicalReads	Stalls	PagesTouched	PagesRead	BuffersToMRU	BuffersToLRU	CacheName
1	0	0	4,096	2,883,576	9,995	0	6,223	9,995	9,986	9	default data cache
2	0	0	32,768	262,144	46,289	0	59,536	370,312	58,160	312,152	default data cache
3	1	0	4,096	10,240	0	0	0	0	0	0	ec_keys_cache
4	2	0	4,096	563,200	0	0	0	0	0	0	imdb_big_cache
5	3	0	4,096	143,360	0	0	81	0	0	0	imdb_small_cache
6	4	0	4,096	51,200	0	0	0	0	0	0	log_cache
7	4	0	32,768	102,400	0	0	0	0	0	0	log_cache
8	5	0	4,096	102,400	1	0	1	1	1	0	system_cache
9	6	0	4,096	393,216	479	0	556	479	479	0	tempdb_cache
10	6	0	32,768	131,072	0	0	0	0	0	0	tempdb_cache

Results | Messages

Line 1 Column 27 | 10 rows

LOG CACHE CONFIGURATION/SIZING:

monDeviceIO & monIOQueue

- **Common problem**
  - Log cache undersized
  - Log cache in default data cache
- **Considerations**
  - Problem is 4K pool in default data cache (or any cache) is not reserved for log use
    - Impossible to tell if reads are occurring or not
- **MDA Quick Tip**
  - Watch Physical Reads
    - Easy if log devices are separated from others
      - SY device used for log in one DB shouldn't be used for data for another DB for optimal performance
    - May need to use monIOQueue otherwise to get fuzzy idea of whether log/data
  - Rationale
    - Prefer checkpoint and log scans to be cached
    - Dump tran speed definitely impacted if not cached
    - RepAgent latency could be a factor, but often not due to size of log cache, so log caches >100MB are usually a waste (unless dump tran dictates)

monDeviceIO		
InstanceID	tinyint	<pk>
Reads	int	
APFReads	int	
Writes	int	
DevSemaphoreRequests	int	
DevSemaphoreWaits	int	
IOTime	int	
LogicalName	varchar(30)	<pk>
PhysicalName	varchar(128)	

monIOQueue		
InstanceID	tinyint	<pk,fk>
IOs	int	
IOTime	int	
LogicalName	varchar(30)	<pk,fk>
IOType	varchar(12)	<pk>

MONDEVICEIO EXAMPLE

…from a short customer test with ASE 15.0.3

Reads	APFReads	Writes	IOTime	ms per IO	LogicalName
2766	242	15189	30200	1.7	tempdb07
2464	0	8973	25800	2.3	tempdb04
2414	0	9052	27700	2.4	tempdb06
2384	0	20899	35400	1.5	tempdb03
2286	232	11910	31600	2.2	tempdb01
2183	0	9137	29000	2.6	tempdb05
2067	0	9996	31900	2.6	tempdb02
346	242	14417	0	0.0	tempdbbatch
214	0	64192	183700	2.9	DB1_log02
20	0	8313	20300	2.4	DB2_log01
7	0	66	100	1.4	tempdbsa01
7	0	4858	24300	5.0	DB3_log01
5	0	6631	28400	4.3	DB4_log01

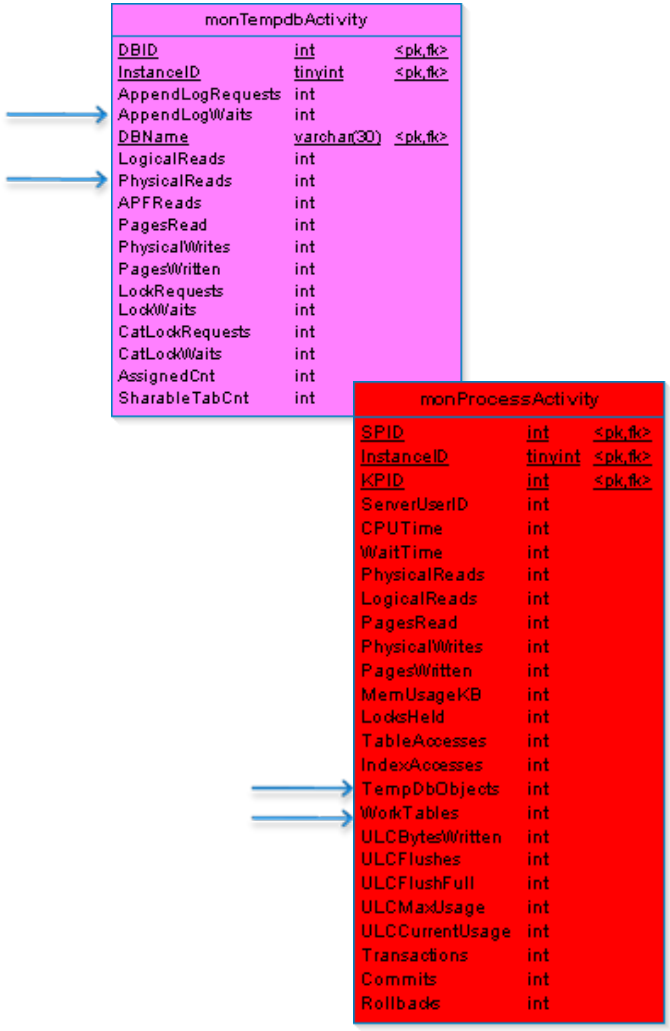
Notes:

- Tempdb probably in default data cache and it is getting pushed out of cache frequently even for small tempdb tables…or the tempdb cache is too small
  - We know this because the absence of APFReads indicates smaller tempdb tables (or APF prefetch % is too low)
  - Also, either 'session tempdb log cache size' is set too low (and tempdb log writes) or, procs are not dropping #temps as soon as done (waiting for proc exit to clean up?) as getting pushed out of cache is probably contributing to writes (sloppy coding practices will cost you hardware performance)…use monIOQueue to split out tempdb log/data ratios
- Transaction log cache is probably 'assumed' to be the 4K pool in default data cache - or the log cache is slightly too small (checkpoint reads)…or the RepAgent is lagging (increasing log cache won't help this)

MULTIPLE TEMPDB'S & CACHES

monTempdbActivity (15.5+), monProcessActivity

- **Catalog contention should be gone**
  - 15.0.2 with RLC
- **Log semaphore contention still there**
  - Contention%=AppendLogWaits/AppendLogRequests
  - Increase 'session tempdb log cache size'
    - Recommend 32KB (min) → 128KB (max)
  - …..or add another tempdb to tempdb group
- **Writes should be a lot less**
  - No more checkpoint flushes of dirty pages
  - Session tempdb log cache
  - No more SLR's or synchronous page splits
  - Make sure directio=false and dsync=false (cached UFS)
- **Configuration**
  - Turn off HK Wash in all tempdb caches
    - Add 'cache status = HK ignore cache' in cfg file
  - 3-4 small tempdbs for OLTP
    - Separate named caches for each to reduce spinlock contention during #temp creation/dropping
    - Candidates for IMDB or relaxed cache strategy
    - Watch PhysicalReads for cache sizing…ideally 0
  - 1-2 tempdbs for batch processes
    - Can share named cache with others and sa tempdb (below)
    - PhysicalReads likely due to table size vs. cache size
  - 1 tempdb for SA (update stats, etc)



CACHE CONFIGURATION/SIZING:

monOpenObjectActivity

- **Common problem**
  - Usual mash of 2-3 named caches and that is it
  - Dumping stat counters from sp_sysmon shows some cache partitions at high spinlock contention
    - Others with 0 contention results in low average
- **Conventional Wisdom**
  - Add cache partitions
- **MDA Quick Tip**
  - monOpenObjectActivity is a ***critical*** monitoring table.
    - The amount of application tuning and troubleshooting information it provides is far beyond your imagination
    - We will show how to use it to tune caches

monOpenObjectActivity		
DBID	int	<pk,fk>
ObjectID	int	<pk>
IndexID	int	<pk>
InstanceID	tinyint	<pk,fk>
DBName	varchar(30)	<pk,fk>
ObjectName	varchar(30)	<pk>
LogicalReads	int	
PhysicalReads	int	
APFReads	int	
PagesRead	int	
PhysicalWrites	int	
PagesWritten	int	
RowsInserted	int	
RowsDeleted	int	
RowsUpdated	int	
Operations	int	
LockRequests	int	
LockWaits	int	
OptSelectCount	int	
LastOptSelectDate	datetime	
UsedCount	int	
LastUsedDate	datetime	
HkgcRequests	int	
HkgcPending	int	
HkgcOverflows	int	
PhysicalLocks	int	
PhysicalLocksRetained	int	
PhysicalLocksRetainWaited	int	
PhysicalLocksDeadlocks	int	
PhysicalLocksWaited	int	
PhysicalLocksPageTransfer	int	
TransferReqWaited	int	
AvgPhysicalLockWaitTime	real	
AvgTransferReqWaitTime	real	
TotalServiceRequests	int	
PhysicalLocksDowngraded	int	
PagesTransferred	int	
ClusterPageWrites	int	
AvgServiceTime	real	
AvgTimeWaitedOnLocalUsers	real	
AvgTransferSendWaitTime	real	
AvgIOServiceTime	real	
AvgDowngradeServiceTime	real	

YOUR FIRST REAL TEST FOR TODAY

What Cache Configs are You Going to Do??? (4 hour sample)

TableName	Index ID	Logical Reads	Physical Reads	APFReads	Rows Inserted	Rows Deleted	Rows Updated	Lock Requests	Lock Waits	Used Count
site	0	110,215,414	0	0	0	0	0	0	0	0
customer	2	101,301,758	0	0	0	0	0			14311
client_event	0	86,471,323	0	0	7,235	0	35,800	186,953	20,550	0
customer_eligibility	3	56,935,200	0	0	0	0	0			0
customer_eligibility	0	46,460,814	0	0	0	0	0	0	0	0
permission_relation	2	33,331,924	0	0	0	0	0			11107604
letter_request	0	21,517,184	0	19,245,616	0	0	0	0	0	6968
permission_trans	3	13,261,549	329	0	1,569,209	1,569,210	0			616745
permission_trans	2	12,619,479	0	0	1,569,212	1,569,204	0			14168
client_steerage_detail	3	9,435,417	0	0	0	0	0			0
result_selections	0	8,646,750	0	8,307,689	0	0	0	243,215	0	14129
customer_site	3	8,123,220	0	0	0	0	0			0
site_client	2	6,371,175	0	0	0	0	0			14311
permission	0	4,167,037	0	0	0	0	0	4,167,064	0	1041762
cpt_grouping_desc	0	4,123,720	0	0	0	0	0	4,123,724	0	809656
permission_trans	0	3,453,833	2,227	0	1,569,211	1,569,213	0	13,930,857	3,133	0
customer_rules	2	3,020,451	0	0	0	0	0			0
client_search_list	0	2,570,376	0	0	0	0	0	2,599,612	0	0
client_search_list	1	2,570,376	0	0	0	0	0			2570372
customer_rules	0	2,463,521	0	0	0	0	0	4,866,200	0	0
event_status_summary	2	2,130,222	3	0	183,968	184,028	0			335729
cpt_group_addon	0	2,058,968	0	2,058,969	0	0	0	0	0	1029486
group_trans	2	2,043,431	9	0	184,183	184,184	0			2158504
permission_relation	0	1,926,847	0	0	0	0	0	13,048,693	0	0
client_event	9	1,690,384	0	0	42,897	35,800	0			15253
result_selection_comments	0	28,258	0	28,258	0	0	0	7	0	14129
call_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156
client_event_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156

ANSWER #1

Customer (Reference) Data Cache (relaxed cache strategy, ~1GB→2GB)

TableName	Index ID	Logical Reads	Physical Reads	APFReads	Rows Inserted	Rows Deleted	Rows Updated	Lock Requests	Lock Waits	Used Count
site	0	110,215,414	0	0	0	0	0	0	0	0
customer	2	101,301,758	0	0	0	0	0			14311
client_event	0	86,471,323	0	0	7,235	0	35,800	186,953	20,550	0
customer_eligibility	3	56,935,200	0	0	0	0	0			0
customer_eligibility	0	46,460,814	0	0	0	0	0	0	0	0
permission_relation	2	33,331,924	0	0	0	0	0			11107604
letter_request	0	21,517,184	0	19,245,616	0	0	0	0	0	6968
permission_trans	3	13,261,549	329	0	1,569,209	1,569,210	0			616745
permission_trans	2	12,619,479	0	0	1,569,212	1,569,204	0			14168
client_steerage_detail	3	9,435,417	0	0	0	0	0			0
result_selections	0	8,646,750	0	8,307,689	0	0	0	243,215	0	14129
customer_site	3	8,123,220	0	0	0	0	0			0
site_client	2	6,371,175	0	0	0	0	0			14311
permission	0	4,167,037	0	0	0	0	0	4,167,064	0	1041762
cpt_grouping_desc	0	4,123,720	0	0	0	0	0	4,123,724	0	809656
permission_trans	0	3,453,833	2,227	0	1,569,211	1,569,213	0	13,930,857	3,133	0
customer_rules	2	3,020,451	0	0	0	0	0			0
client_search_list	0	2,570,376	0	0	0	0	0	2,599,612	0	0
client_search_list	1	2,570,376	0	0	0	0	0			2570372
customer_rules	0	2,463,521	0	0	0	0	0	4,866,200	0	0
event_status_summary	2	2,130,222	3	0	183,968	184,028	0			335729
cpt_group_addon	0	2,058,968	0	2,058,969	0	0	0	0	0	1029486
group_trans	2	2,043,431	9	0	184,183	184,184	0			2158504
permission_relation	0	1,926,847	0	0	0	0	0	13,048,693	0	0
client_event	9	1,690,384	0	0	42,897	35,800	0			15253
result_selection_comments	0	28,258	0	28,258	0	0	0	7	0	14129
call_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156
client_event_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156

…reduces LRU→MRU relinkages by >300M…and spinlock contention as well….

ANSWER #2

Volatile Data Cache #1 (default or relaxed cache strategy, ~50MB→100MB, #partitions = #engines)….IMDB candidate (or cached UFS device + segments)

TableName	Index ID	Logical Reads	Physical Reads	APFReads	Rows Inserted	Rows Deleted	Rows Updated	Lock Requests	Lock Waits	Used Count
site	0	110,215,414	0	0	0	0	0	0	0	0
customer	2	101,301,758	0	0	0	0	0			14311
client_event	0	86,471,323	0	0	7,235	0	35,800	186,953	20,550	0
customer_eligibility	3	56,935,200	0	0	0	0	0			0
customer_eligibility	0	46,460,814	0	0	0	0	0	0	0	0
permission_relation	2	33,331,924	0	0	0	0	0			11107604
letter_request	0	21,517,184	0	19,245,616	0	0	0	0	0	6968
permission_trans	3	13,261,549	329	0	1,569,209	1,569,210	0			616745
permission_trans	2	12,619,479	0	0	1,569,212	1,569,204	0			14168
client_steerage_detail	3	9,435,417	0	0	0	0	0			0
result_selections	0	8,646,750	0	8,307,689	0	0	0	243,215	0	14129
customer_site	3	8,123,220	0	0	0	0	0			0
site_client	2	6,371,175	0	0	0	0	0			14311
permission	0	4,167,037	0	0	0	0	0	4,167,064	0	1041762
cpt_grouping_desc	0	4,123,720	0	0	0	0	0	4,123,724	0	809656
permission_trans	0	3,453,833	2,227	0	1,569,211	1,569,213	0	13,930,857	3,133	0
customer_rules	2	3,020,451	0	0	0	0	0			0
client_search_list	0	2,570,376	0	0	0	0	0	2,599,612	0	0
client_search_list	1	2,570,376	0	0	0	0	0			2570372
customer_rules	0	2,463,521	0	0	0	0	0	4,866,200	0	0
event_status_summary	2	2,130,222	3	0	183,968	184,028	0			335729
cpt_group_addon	0	2,058,968	0	2,058,969	0	0	0	0	0	1029486
group_trans	2	2,043,431	9	0	184,183	184,184	0			2158504
permission_relation	0	1,926,847	0	0	0	0	0	13,048,693	0	0
client_event	9	1,690,384	0	0	42,897	35,800	0			15253
result_selection_comments	0	28,258	0	28,258	0	0	0	7	0	14129
call_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156
client_event_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156

…rows are constantly inserted/deleted….if not IMDB, consider separate DB with delayed commit….

…HK ignore cache if not using relaxed cache strategy….

ANSWER #3

Volatile Data Cache #2 (default cache strategy, ~50MB→100MB, #partitions = #engines)….or let this remain in default data cache since it is real business xactns

TableName	Index ID	Logical Reads	Physical Reads	APFReads	Rows Inserted	Rows Deleted	Rows Updated	Lock Requests	Lock Waits	Used Count
site	0	110,215,414	0	0	0	0	0	0	0	0
customer	2	101,301,758	0	0	0	0	0			14311
client_event	0	86,471,323	0	0	7,235	0	35,800	186,953	20,550	0
customer_eligibility	3	56,935,200	0	0	0	0	0			0
customer_eligibility	0	46,460,814	0	0	0	0	0	0	0	0
permission_relation	2	33,331,924	0	0	0	0	0			11107604
letter_request	0	21,517,184	0	19,245,616	0	0	0	0	0	6968
permission_trans	3	13,261,549	329	0	1,569,209	1,569,210	0			616745
permission_trans	2	12,619,479	0	0	1,569,212	1,569,204	0			14168
client_steerage_detail	3	9,435,417	0	0	0	0	0			0
result_selections	0	8,646,750	0	8,307,689	0	0	0	243,215	0	14129
customer_site	3	8,123,220	0	0	0	0	0			0
site_client	2	6,371,175	0	0	0	0	0			14311
permission	0	4,167,037	0	0	0	0	0	4,167,064	0	1041762
cpt_grouping_desc	0	4,123,720	0	0	0	0	0	4,123,724	0	809656
permission_trans	0	3,453,833	2,227	0	1,569,211	1,569,213	0	13,930,857	3,133	0
customer_rules	2	3,020,451	0	0	0	0	0			0
client_search_list	0	2,570,376	0	0	0	0	0	2,599,612	0	0
client_search_list	1	2,570,376	0	0	0	0	0			2570372
customer_rules	0	2,463,521	0	0	0	0	0	4,866,200	0	0
event_status_summary	2	2,130,222	3	0	183,968	184,028	0			335729
cpt_group_addon	0	2,058,968	0	2,058,969	0	0	0	0	0	1029486
group_trans	2	2,043,431	9	0	184,183	184,184	0			2158504
permission_relation	0	1,926,847	0	0	0	0	0	13,048,693	0	0
client_event	9	1,690,384	0	0	42,897	35,800	0			15253
result_selection_comments	0	28,258	0	28,258	0	0	0	7	0	14129
call_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156
client_event_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156

ANSWER #4

Reference Data Cache (relaxed cache strategy, ~50MB→100MB)

TableName	Index ID	Logical Reads	Physical Reads	APFReads	Rows Inserted	Rows Deleted	Rows Updated	Lock Requests	Lock Waits	Used Count
site	0	110,215,414	0	0	0	0	0	0	0	0
customer	2	101,301,758	0	0	0	0	0			14,311
client_event	0	86,471,323	0	0	7,235	0	35,800	186,953	20,550	0
customer_eligibility	3	56,935,200	0	0	0	0	0			0
customer_eligibility	0	46,460,814	0	0	0	0	0	0	0	0
permission_relation	2	33,331,924	0	0	0	0	0			11,107,604
letter_request	0	21,517,184	0	19,245,616	0	0	0	0	0	6,968
permission_trans	3	13,261,549	329	0	1,569,209	1,569,210	0			616,745
permission_trans	2	12,619,479	0	0	1,569,212	1,569,204	0			14,168
client_steerage_detail	3	9,435,417	0	0	0	0	0			0
result_selections	0	8,646,750	0	8,307,689	0	0	0	243,215	0	14,129
customer_site	3	8,123,220	0	0	0	0	0			0
site_client	2	6,371,175	0	0	0	0	0			143,11
permission	0	4,167,037	0	0	0	0	0	4,167,064	0	1,041,762
cpt_grouping_desc	0	4,123,720	0	0	0	0	0	4,123,724	0	809,656
permission_trans	0	3,453,833	2,227	0	1,569,211	1,569,213	0	13,930,857	3,133	0
customer_rules	2	3,020,451	0	0	0	0	0			0
client_search_list	0	2,570,376	0	0	0	0	0	2,599,612	0	0
client_search_list	1	2,570,376	0	0	0	0	0			2,570,372
customer_rules	0	2,463,521	0	0	0	0	0	4,866,200	0	0
event_status_summary	2	2,130,222	3	0	183,968	184,028	0			335,729
cpt_group_addon	0	2,058,968	0	2,058,969	0	0	0	0	0	1,029,486
group_trans	2	2,043,431	9	0	184,183	184,184	0			2,158,504
permission_relation	0	1,926,847	0	0	0	0	0	13,048,693	0	0
client_event	9	1,690,384	0	0	42,897	35,800	0			15,253
result_selection_comments	0	28,258	0	28,258	0	0	0	7	0	14,129
call_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7,156
client_event_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7,156

ANSWER #5

Key Sequence Data Cache (relaxed cache strategy, ~5MB)

TableName	Index ID	Logical Reads	Physical Reads	APFReads	Rows Inserted	Rows Deleted	Rows Updated	Lock Requests	Lock Waits	Used Count
site	0	110,215,414	0	0	0	0	0	0	0	0
customer	2	101,301,758	0	0	0	0	0			14,311
client_event	0	86,471,323	0	0	7,235	0	35,800	186,953	20,550	0
customer_eligibility	3	56,935,200	0	0	0	0	0			0
customer_eligibility	0	46,460,814	0	0	0	0	0	0	0	0
permission_relation	2	33,331,924	0	0	0	0	0			11,107,604
letter_request	0	21,517,184	0	19,245,616	0	0	0	0	0	6,968
permission_trans	3	13,261,549	329	0	1,569,209	1,569,210	0			616,745
permission_trans	2	12,619,479	0	0	1,569,212	1,569,204	0			14,168
client_steerage_detail	3	9,435,417	0	0	0	0	0			0
result_selections	0	8,646,750	0	8,307,689	0	0	0	243,215	0	14,129
customer_site	3	8,123,220	0	0	0	0	0			0
site_client	2	6,371,175	0	0	0	0	0			143,11
permission	0	4,167,037	0	0	0	0	0	4,167,064	0	1,041,762
cpt_grouping_desc	0	4,123,720	0	0	0	0	0	4,123,724	0	809,656
permission_trans	0	3,453,833	2,227	0	1,569,211	1,569,213	0	13,930,857	3,133	0
customer_rules	2	3,020,451	0	0	0	0	0			0
client_search_list	0	2,570,376	0	0	0	0	0	2,599,612	0	0
client_search_list	1	2,570,376	0	0	0	0	0			2,570,372
customer_rules	0	2,463,521	0	0	0	0	0	4,866,200	0	0
event_status_summary	2	2,130,222	3	0	183,968	184,028	0			335,729
cpt_group_addon	0	2,058,968	0	2,058,969	0	0	0	0	0	1,029,486
group_trans	2	2,043,431	9	0	184,183	184,184	0			2,158,504
permission_relation	0	1,926,847	0	0	0	0	0	13,048,693	0	0
client_event	9	1,690,384	0	0	42,897	35,800	0			15,253
result_selection_comments	0	28,258	0	28,258	0	0	0	7	0	14,129
call_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7,156
client_event_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7,156

CACHE CONFIGURATION CHANGES

Putting it All Together….

TableName	Index ID	Logical Reads	Physical Reads	APFReads	Rows Inserted	Rows Deleted	Rows Updated	Lock Requests	Lock Waits	Used Count
site	0	110,215,414	0	0	0	0	0	0	0	0
customer	2	101,301,758	0	0	0	0	0			14311
client_event	0	86,471,323	0	0	7,235	0	35,800	186,953	20,550	0
customer_eligibility	3	56,935,200	0	0	0	0	0			0
customer_eligibility	0	46,460,814	0	0	0	0	0	0	0	0
permission_relation	2	33,331,924	0	0	0	0	0			11107604
letter_request	0	21,517,184	0	19,245,616	0	0	0	0	0	6968
permission_trans	3	13,261,549	329	0	1,569,209	1,569,210	0			616745
permission_trans	2	12,619,479	0	0	1,569,212	1,569,204	0			14168
client_steerage_detail	3	9,435,417	0	0	0	0	0			0
result_selections	0	8,646,750	0	8,307,689	0	0	0	243,215	0	14129
customer_site	3	8,123,220	0	0	0	0	0			0
site_client	2	6,371,175	0	0	0	0	0			14311
permission	0	4,167,037	0	0	0	0	0	4,167,064	0	1041762
cpt_grouping_desc	0	4,123,720	0	0	0	0	0	4,123,724	0	809656
permission_trans	0	3,453,833	2,227	0	1,569,211	1,569,213	0	13,930,857	3,133	0
customer_rules	2	3,020,451	0	0	0	0	0			0
client_search_list	0	2,570,376	0	0	0	0	0	2,599,612	0	0
client_search_list	1	2,570,376	0	0	0	0	0			2570372
customer_rules	0	2,463,521	0	0	0	0	0	4,866,200	0	0
event_status_summary	2	2,130,222	3	0	183,968	184,028	0			335729
cpt_group_addon	0	2,058,968	0	2,058,969	0	0	0	0	0	1029486
group_trans	2	2,043,431	9	0	184,183	184,184	0			2158504
permission_relation	0	1,926,847	0	0	0	0	0	13,048,693	0	0
client_event	9	1,690,384	0	0	42,897	35,800	0			15253
result_selection_comments	0	28,258	0	28,258	0	0	0	7	0	14129
call_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156
client_event_id	0	14,312	0	14,312	0	0	7,156	14,312	0	7156

PUTTING IT ALL TOGETHER EXPLAINED

The Benefits of Actively Managing Your Cache vs. Not Managing It

- **Significant Reduction in MRU→LRU relinkages**
  - >300M in 4 hours (23,000/second)
  - Should see a big drop in cache spinlock contention
  - Regardless, the tasks will complete slightly quicker each
- **Volatile data no longer pushing others out of cache**
  - Tables with a lot of insert/delete pairs - especially DOL - will allocate new pages (which grab LRU and go to MRU)
    - Also applies to volatile indexes on actively updated tables
  - DOL tables need to have 'enable housekeeper GC=5'
    - See next slide
- **What to use dbcc tune(desbind) on….**
  - Any table …
    - …we bound to a named cache
    - …top 20 tables in default data cache by LogicalReads
  - Not forgetting the triggers, defaults, rules on those tables

ENABLE HOUSEKEEPER GC = [4 | 5]

Are you using datarows or datapage locking anywhere???

ObjectName	Index ID	Rows Updated	Rows Deleted	Hkgc Requests	Hkgc Pending	Hkgc Overflows
BOOKING_JOURNAL	0	3,683,307	466,035	72,805	1,565	3
BOOKING_JOURNAL_TERM_SESS	4	0	471,650	27,969	411	2,967
IND_BOOK_JOUR_STOC_NUM	3	0	4,107,290	17,537	177	2,170
PK_BOOKING_JOURNAL	2	0	471,650	24,526	229	3,118

The choice is yours:

a)Do reorgs all the time…and whine about it

b)Set exp_row_size and enable housekeeper GC

PROCEDURE CACHE

JUST A BIT OF A DISCUSSION

PROCEDURE CACHE

Oh, My - the Headaches Begin

- **Recent spate of proc cache fragmentation**
  - Fragmentation is a real slap in the face that proc cache is likely too small
    - …or someone is doing something real goofy such as statement cache without literal autoparameterization
    - …or constantly recreating fully prepared statements vs. re-using them
    - …or something else causing a lot of cache turnover
  - Hint: If you ever get told to use dbcc proc_cache(free_unused)….
    - Find the application problem or increase your proc cache…
    - dbcc proc_cache(free_unused) is just a band-aide so you don't bleed to death until you do.
- **Proc reads from disk**
  - Ideally, <1-2/sec per engine excluding procs recompiled
    - This is best visible from sp_sysmon
  - Higher usually indicates proc cache is too small
    - Could be index stats or sorts pushing procs out of cache
    - This is best diagnosed with MDA

PROC CACHE SIZING:

monProcedureCache, monProcedureCacheModuleUsage, and monCachedProcedures

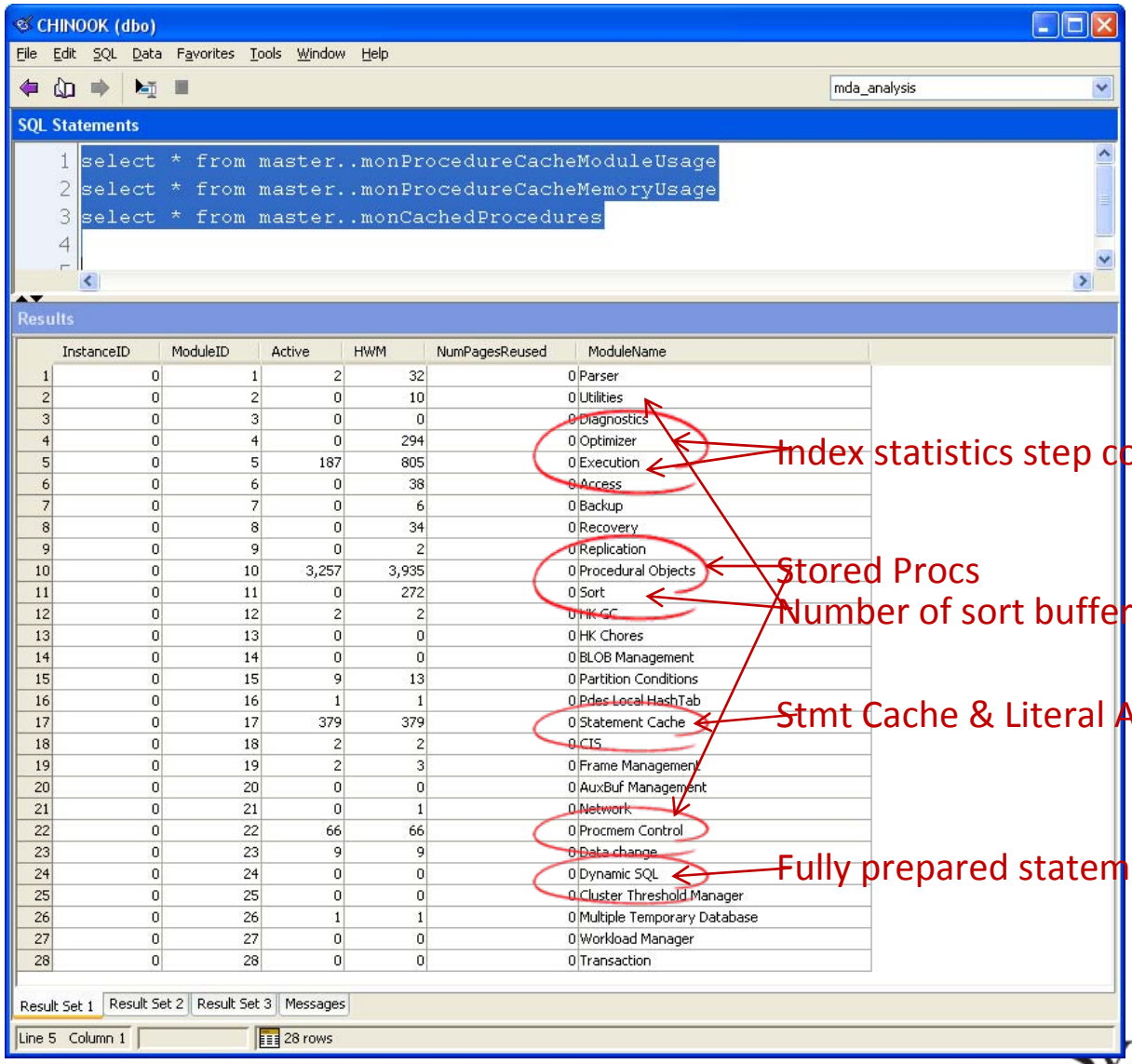
- **monProcedureCache**
  - Loads < 1-2/second per engine
  - If higher, check for procs with recompile
    - Not needed as often with deferred compilation in 15.x unless drastic changes in execution
- **monProcedureCacheModuleUsage**
  - Watch HWM for key modules (next slide)
  - If Optimization/Execution too high, check number of statistics
  - If Sort is too high - check number of sort buffers
    - Creating indexes on #temp in stored procs with a value > default for number of sort buffers will use more proc cache than time saved
- **monCachedProcedures**
  - See how many distinct PlanID's are in cache for any given stored procedure
    - If it fluctuates a lot, consider increasing proc cache
    - Use dbcc tune(desbind) on top 50 procs

monProcedureCache		
Requests	int	
Loads	int	
Writes	int	
Stalls	int	
InstanceID	tinyint	<pk>

monProcedureCacheModuleUsage		
InstanceID	tinyint	<pk>
ModuleID	int	<pk>
Active	int	
HWM	int	
NumPagesReused	int	
ModuleName	varchar(30)	

monCachedProcedures		
ObjectID	int	<pk>
InstanceID	tinyint	<pk>
OwnerUID	int	<pk>
DBID	int	<pk>
PlanID	int	<pk>
MemUsageKB	int	
CompileDate	datetime	<pk>
ObjectName	varchar(30)	<pk>
ObjectType	varchar(32)	
OwnerName	varchar(30)	<pk>
DBName	varchar(30)	<pk>
RequestCnt	int	
TempdbRemapCnt	int	
AvgTempdbRemapTime	int	

MONPROCEDURECACHEMODULEUSAGE



TYPE YOUR QUESTION IN THE ONLINE Q&A BOX OR CALL

1-866-803-2143 UNITED STATES

001-866-888-0267 MEXICO

0800-777-0476 ARGENTINA

01800-9-156448 COLOMBIA

PASSWORD: **SYBASE**

PRESS *1

THANK YOU

FOR MORE INFORMATION

WWW.SYBASE.COM/ASE


SYBASE®
An SAP Company