Customer

Replication Server Performance Tuning & Roadmap

Jeff Tallman jeff.tallman@sap.com
SAP ASE Product Management

Agenda

- **Quick Review of Recent Internals**
  - Due to impacts on performance tuning
  - Both normal (LTL) SRS and CI (HADR) SRS
- **Sizing and OS tuning**
  - T-Shirt Sizing
  - OS & memory allocation considerations
- **Common Performance Issues/Solutions**
  - RepAgent
  - Inbound queue
  - DSI w/ HVAR
- **Roadmap**

Disclaimer

NOTE: Let’s be reasonable

If you are running high volume, then you need Advanced Services Option features

- If you don’t have it, then don’t expect miracles
- Do expect to be told you need it
- It comes in ASE Platform Edition or Replication Server Premium Edition

Expect to do some tuning

- This means collecting and analyzing the RS MC stats
  - ✓ The tools to do this are free - no excuse
- It also means understanding the internals
- …and running several iterations to get a good configuration

However, there still are trade-offs

- You may need a whole lot more resources for SRS than you ever used to use
- Some configs won’t work with others (e.g. incremental apply and distributed apply)

Quick Internals Review

·Changes that improve performance

UKSUG

SAP DATABASE & TECHNOLOGY USER GROU

There are two techniques for scaling with multi-threading

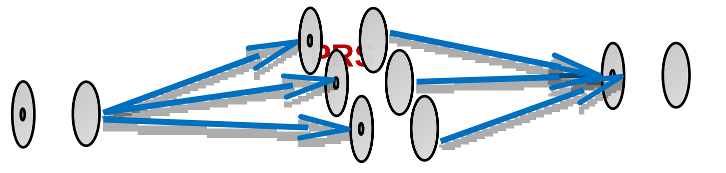
·Parallel Tasks

- Split workload across parallel execution threads
- Scales extremely well when there is no dependency between work units
  - ✓E.g. DBMS with concurrent users/queries
- Work units may need synchronization if serialization (e.g. transaction commit sequencing) is necessary
  - ✓This can impact the effectiveness of parallelization

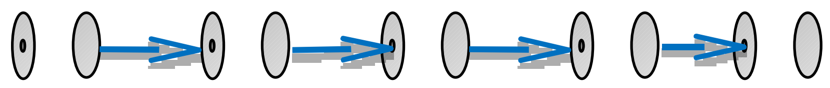
·Pipelining

- Longer tasks are split into smaller chunks
- Each thread does a small chunk and sends rest to next thread
- Scales well when sequencing of work units is important
  - ✓Common in Event Stream Processing

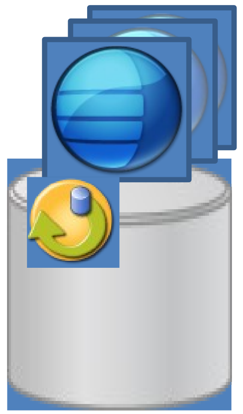
Parallel Tasks



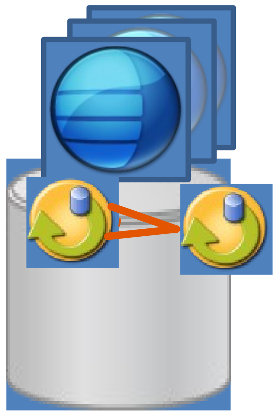
Pipelining



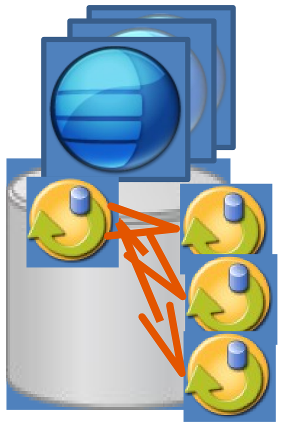
ASE 15.7 approach to multi-threaded RepAgents



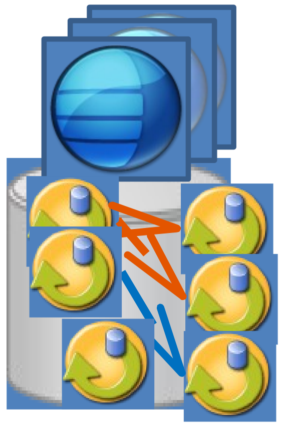
ASE 15.x – single thread (default)



ASE 15.7 – multi-threaded agent
(single scanner & single sender)



ASE 15.7 – multi-threaded agent with MPR
(single scanner & multi-sender)

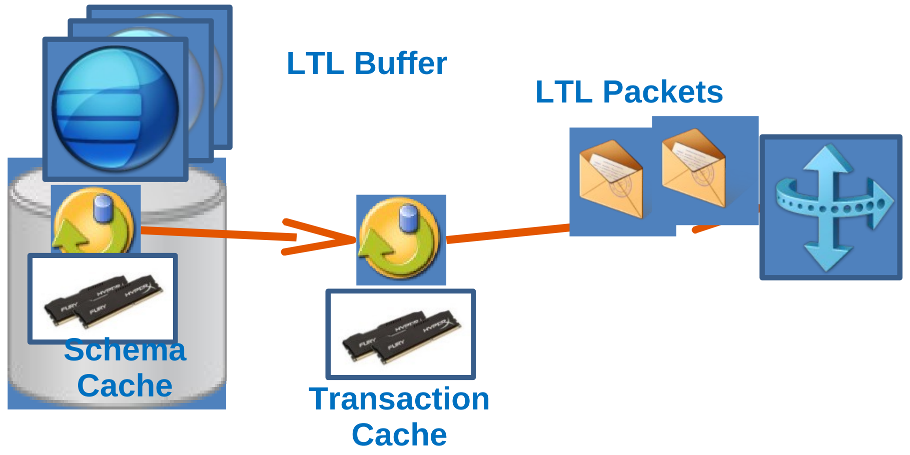


ASE 15.7 sp100 multi-threaded agent with MPR
(multi-scanner & multi-sender)

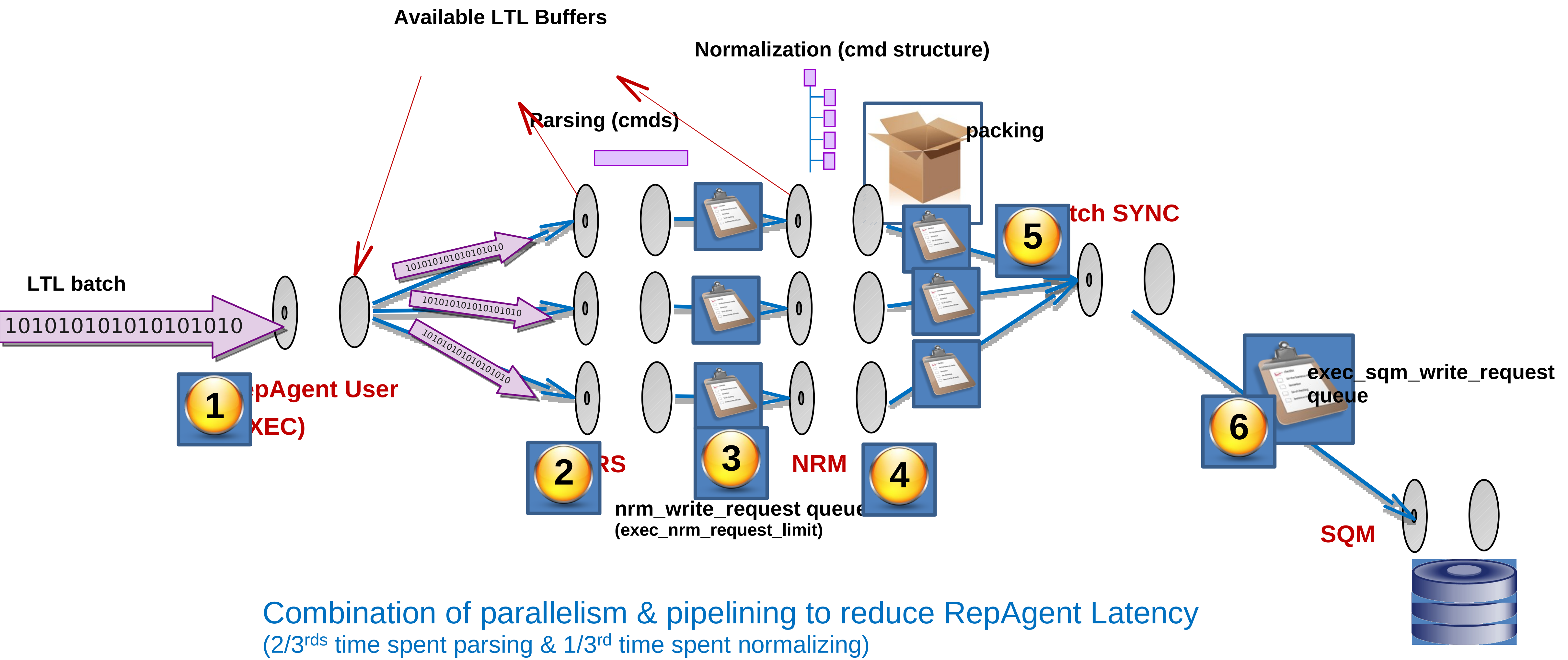
LTL RepAgent Scan/Send Processing

- **Scanner scanned log records**
  - Used schema cache to construct LTL format
  - Log records sent in LTL format to RepAgent sender thread/SPID in ASE

- **Sender**
  - Update transaction cache as necessary
    - ✓ e.g. for proc replication
  - Added LTL log record to LTL buffer
    - ✓ Size controlled by ‘ltl batch size’
  - If LTL buffer full (or batching disabled), flushed LTL buffer to SRS using multiple packets
    - ✓ Packet size controlled by ‘send buffer size’
    - ✓ For example if ‘ltl batch size’ set to 64KB and ‘send buffer size’ set to 8KB, then RepAgent would send 8 8KB packets
  - Sender waits at end of each LTL buffer flush for ack from SRS before clearing buffer



SRS 15.7.1 combination of parallelism & pipelining



Combination of parallelism & pipelining to reduce RepAgent Latency
(2/3rds time spent parsing & 1/3rd time spent normalizing)

SRS 15.7.1 DIST & DSI worker threads

·**Native OS thread pool**

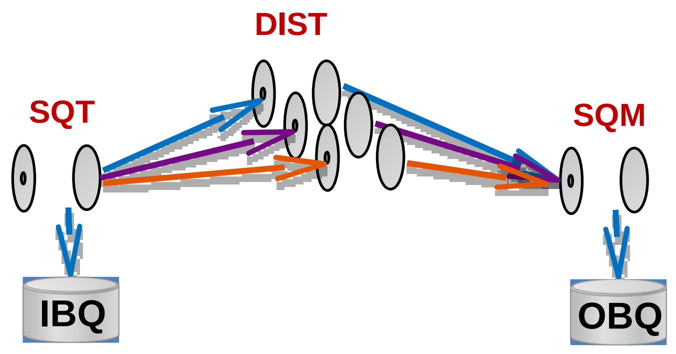
- RS config worker_thread_num

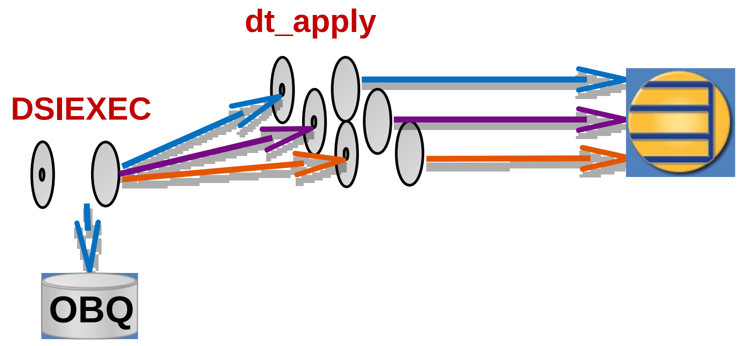
·**Parallel Distribution**

- Uses threads from worker pools
- Either on or off - no way to control threads

·**Distributed Apply**

- Works with HVAR
- Uses parallel threads to apply CDB to RDB
  - ✓Each thread does separate tables
  - ✓Dt_apply config controls number of threads used
- Requires ASE to use XA/DTM
  - ✓Maint user needs DTM role





That was then…..the future is here

- **One of the issues that plagued SRS users was RepAgent latency**
  - Part of the issue was earlier problems with process kernel and deferred async event queue
  - Part of the issue was the async nature and waiting for acks/truncation points
  - Part of the problem also was the time to construct the LTL format

- **Although LTL mode is still supported, future RepAgent uses streaming mode**
  - First implemented in ASE 16sp02 for HADR
  - Also known as CI mode - Canonical Interface
  - Mainstream adoption likely after PL05 (Q4’16)/PL06(Q2’17)…likely the latter (see roadmap)
  - While LTL mode is still supported for non-HADR, it only makes sense in the future to have a single Replication Agent code path to reduce engineering/testing
    - ✓ In addition, future ASE enhancements may only be implemented in streaming mode

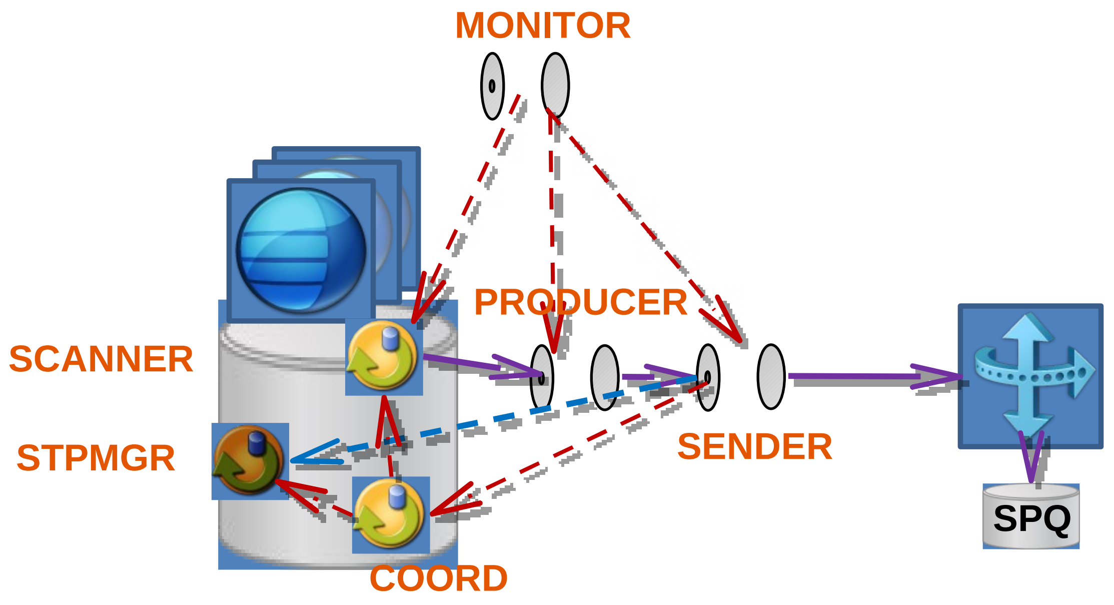
Re-Implementation of RepAgent - CI Mode RepAgent

- **Mix of ASE & Native threads**

- **3 RepAgent ASE tasks (spids) per DB**
  - RepAgent Coordinator - stop/start RepAgent
  - RepAgent Scanner - reads log
  - RepAgent STP Manager - moves STP

- **+ 2 Native threads per DB**
  - Implements stream packaging & sending functions
    - ✓Producer thread(s)
    - ✓Sender thread
  - Spawned by ASE (so uses ASE shared memory)
  - Not part of ASE thread pools

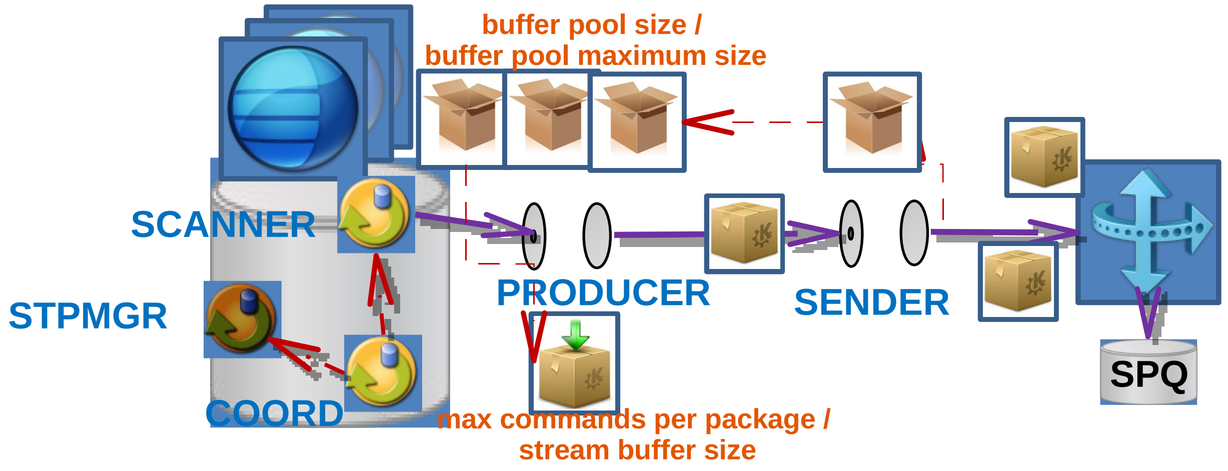
- **+ 1 Native thread per server for monitoring**



Note: ASE 16 still supports LTL mode RepAgent. CI mode is only supported for streaming connections and ASE must be using threaded kernel.

CI Mode RepAgent Configuration & Processing

- **Scanner**
  - Sends log records in binary form to Producer (native OS thread)
- **Producer**
  - Packs commands in a package (buffer) until either max commands per package reached or stream buffer size reached
  - Sends package to sender
- **Sender**
  - Sends package to SRS
- **Benefits**
  - Removes stop/start of single buffer
  - Improved pipeline scalability

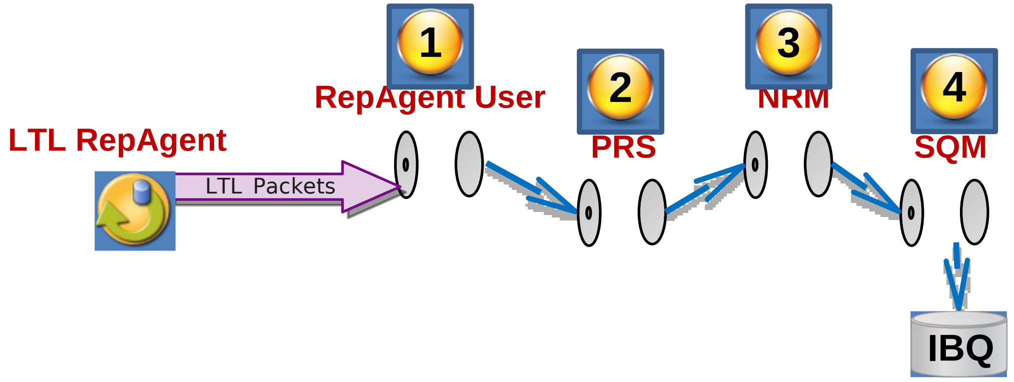


SRS Sizing	Default	Recommended
stream replication	false	true
stream mode	async	sync
max stream retry	-1	-1
buffer pool size	8	20
stream buffer size	1048576	1048576
max commit wait	10000000	5000000
max commands per package	20	20
peak transaction threshold	5	5
peak transaction timer	300	300
buffer pool maximum size	8	100

SRS 15.7.1 sp300+ with SPQ

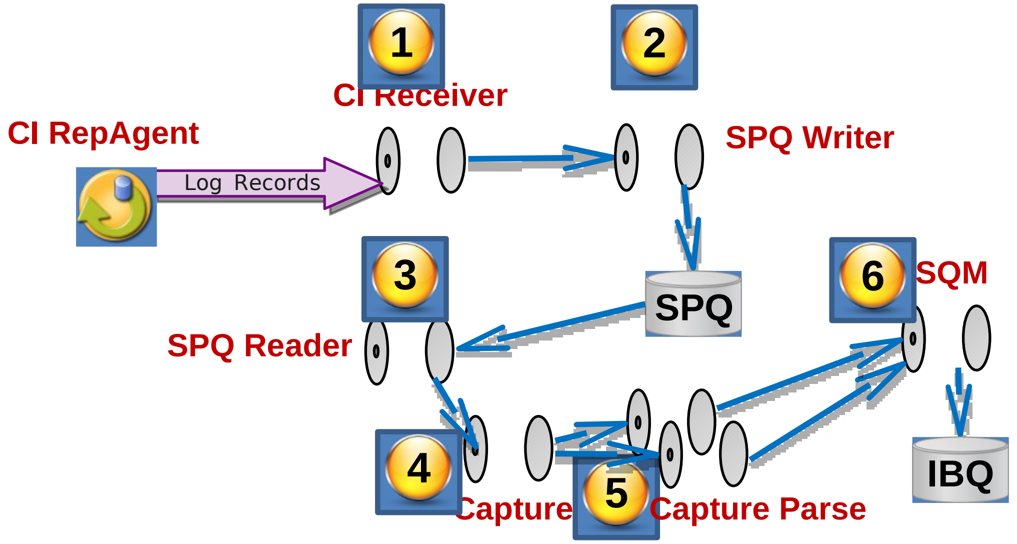
·Traditional LTL interface

- LTL packets are received by RepAgent User thread
- Packets are parsed & normalized before being written to inbound queue
  - ✓Inbound queue is space allocation on possibly shared stable queue/partition



·CI mode interface

- Records are received by CI interface and immediately written to SPQ
  - ✓Each database in HADR cluster will have its own SPQ as a *separate* file
  - ✓Minimum recommended size is 2GB
- New threads read from SPQ and do parse/normalization and write records to normal Inbound queue
  - ✓Can be parallel



ASE SPIDS/OCS Thread vs. Native OS Threads

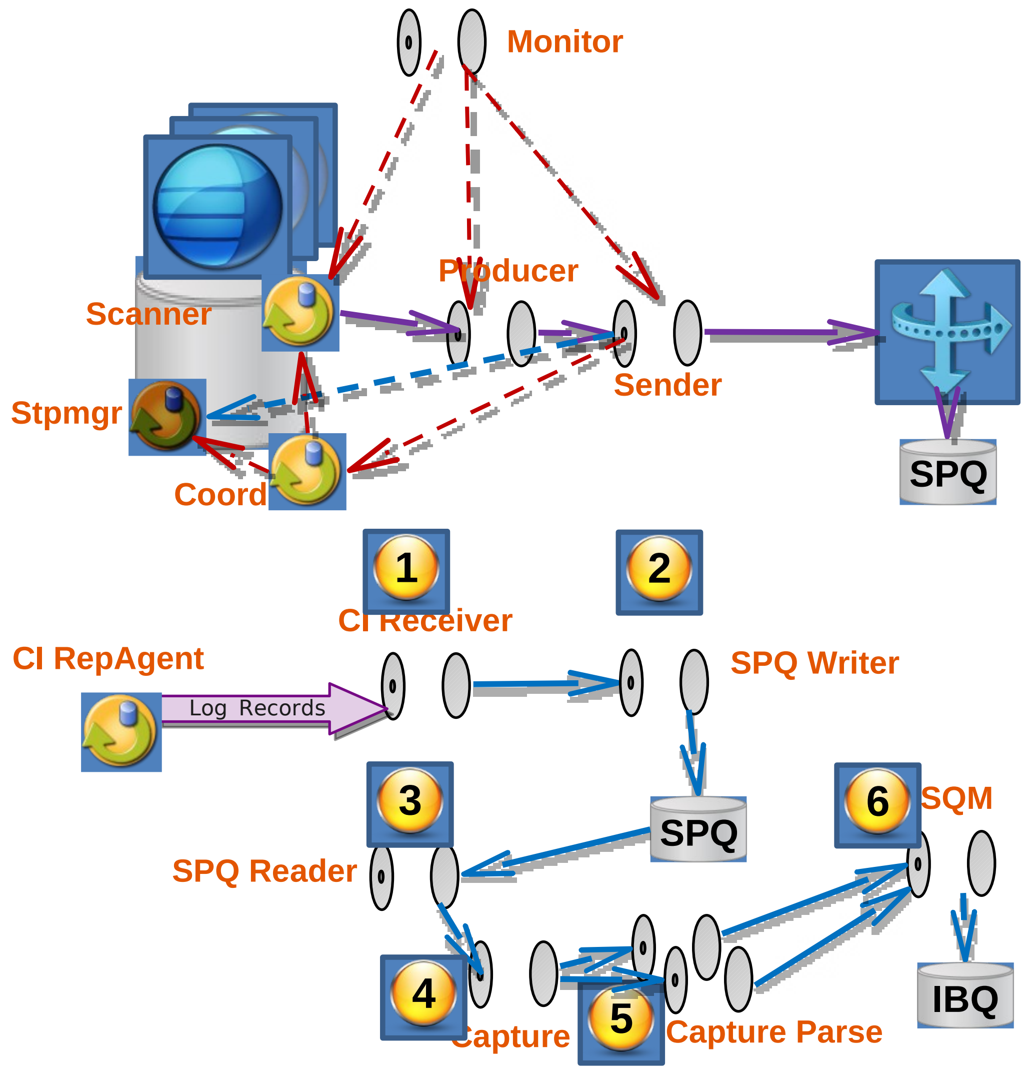
·**Most of the new CI mode leverages native threads**

·**ASE Spid's**

- Problem is that scheduling is up to ASE scheduler
- A spid could be waiting for access to engine to run
- No real signaling between threads
  - ✓ No real way for thread 1 to tell thread 2 to wake up as it now has work to do

·**OCS threads**

- Were implemented as a way of providing threading back when OS's didn't
- OCS provided a lot of infrastructure for thread scheduling and interthread communications… but…
- Isn't as efficient at thread scheduling and sequencing as native threads
  - ✓ E.g. thread signaling
  - ✓ CPU scheduling



The life of a package

Process

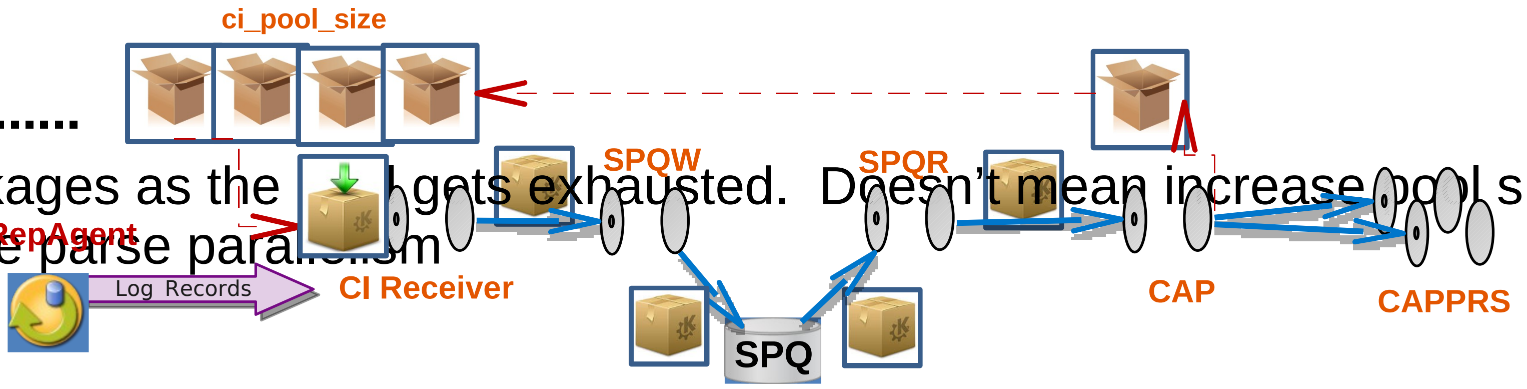
- CI Receiver takes an empty package buffer from pool, fills with stream data and sends to SPQ
- SPQW stores the package in serialized form on disk
- SPQR sends package to Capture thread
- Capture thread parses package into commands and frees package buffer for pool

Rationale

- Just like for RepAgent, this eliminates the stop/start of processing when only working with a single buffer

If CAPPRS is sloww.......

- We run out of packages as the pool gets exhausted. Doesn't mean increase pool size - you may need to increase parse parallelism



CI package layout & schema management

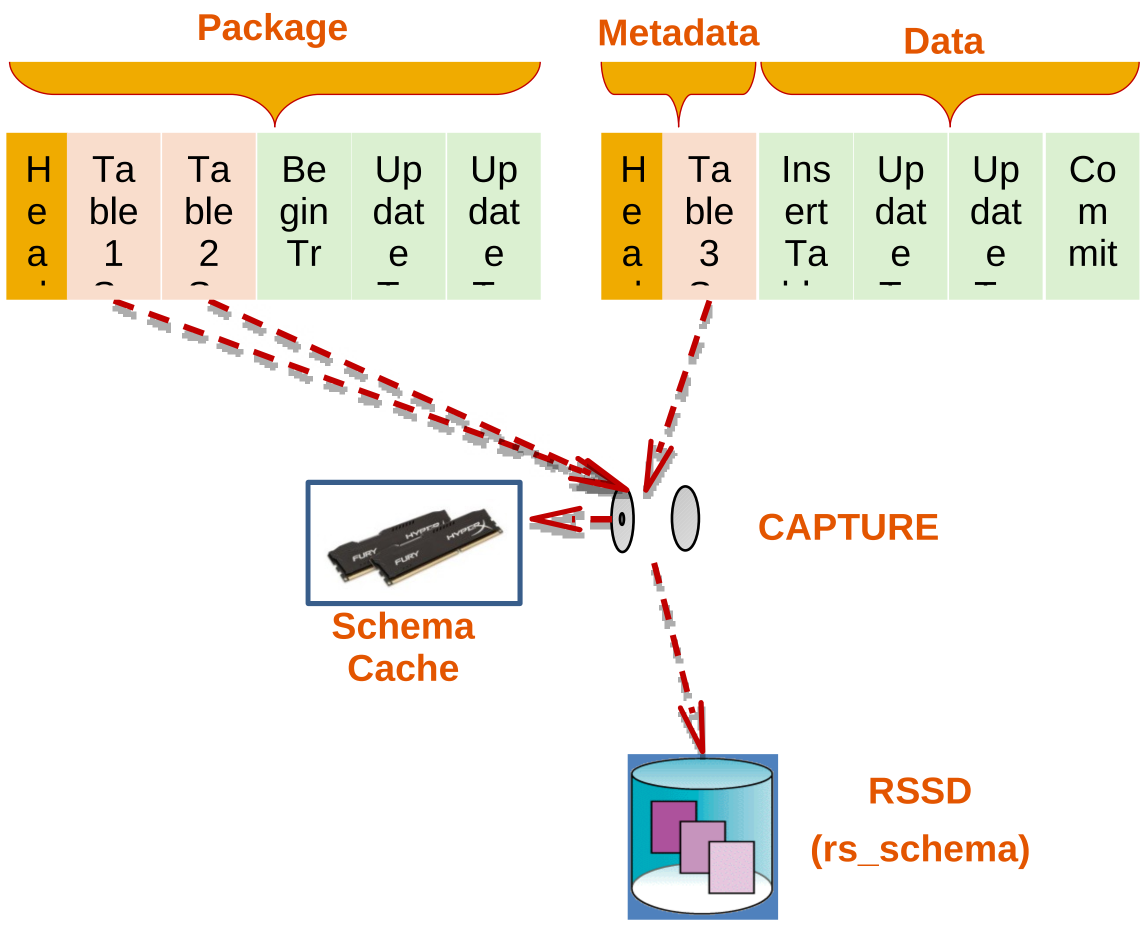
·CI RepAgent sends schema info in package

- Remember, in LTL mode, the repagent had a schema cache and put schema info in LTL
  - ✓ So, 10000 updates on same table, each LTL record has schema info (schema repeated 10000x)
- In CI mode, RATCI simply sends raw log records
- RepAgent only sends schema once when a row of a table is processed for the first time. (like LTL metadata reduction)

·Capture parses schema and stores in RSSD

- Capture adds schemas to schema cache and persists into RSSD after serialized.
- Schema is reloaded from RSSD when it misses from cache.
- Schema with different versions are different schemas.
- Removing of unused schema is not currently supported.

·



Simple Persistent Queue (SPQ)…..isn't so simple

·**CI Receiver**
- Receives stream packages from CI RepAgent
- Puts packages on message queue for SPQ

·**SPQ Logger**
- Reads packages from message queue
- Sends to SPQ Writer for writing

·**SPQ Writer (SPQW)**
- Serializes data for writing
- Puts package on queue for SPQ Grouper to write
- Notifies SPQ Dispatch when write completes

·**SPQ Grouper**
- Groups stream packages together for write efficiency

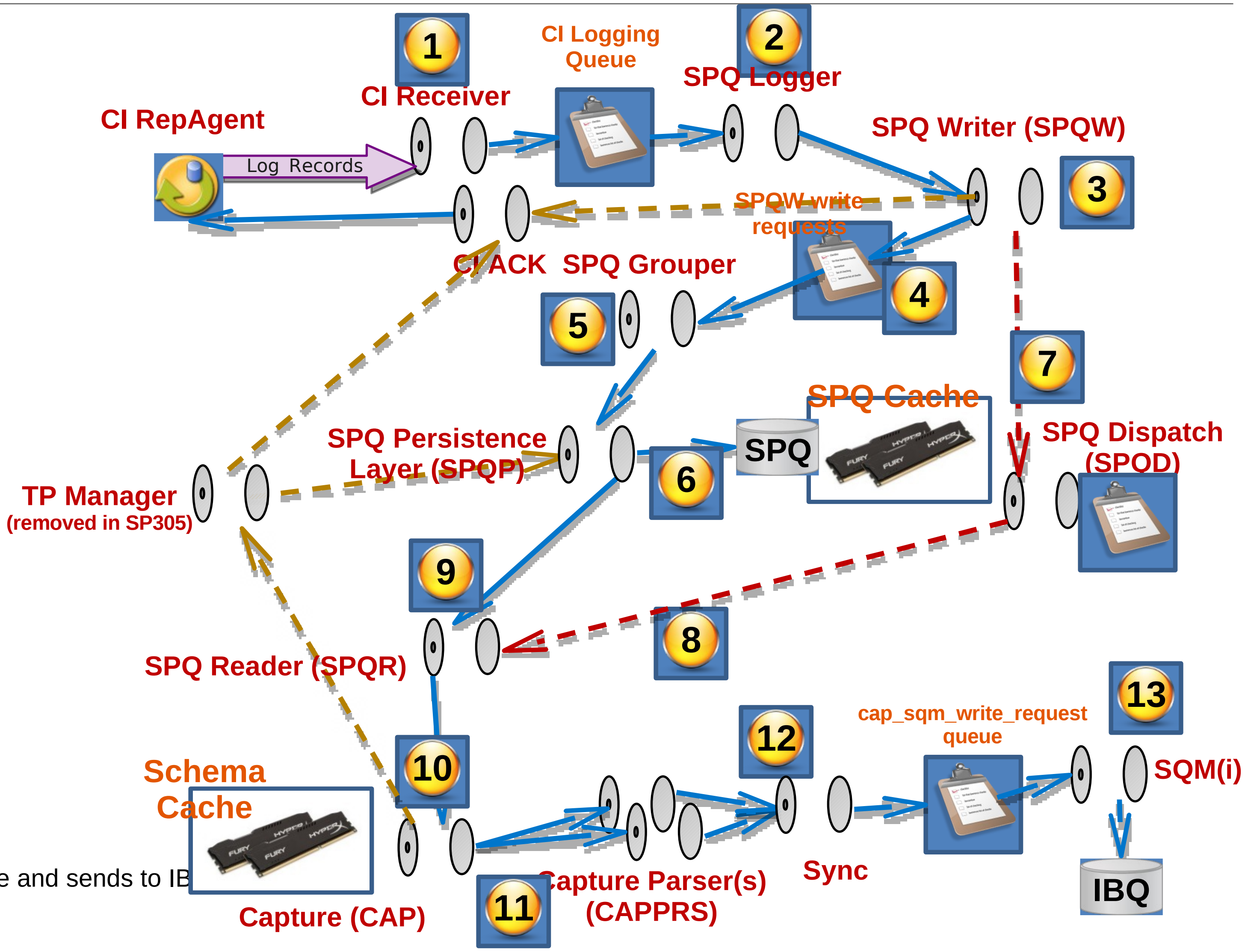
·**SPQ Persistence Layer (SPQP)**
- Performs actual disk reads/writes

·**SPQ Dispatch (SPQD)**
- Notifies reader when a new package is available

·**SPQ Reader (SPQR)**
- Deserializes the package

·**Capture (CAP)**
- Breaks apart the package into commands
- The only OCS thread in chain until SQM

·**Capture Parsers (CAPPRS)**
- Parses and normalizes the command into standard SRS data structure and sends to IB



So what is the point to all of this…

- **ASE RepAgent & SRS transitioning to more pipelining & native OS threads**
  - Earlier attempts of combining low parallelism with low pipelining wasn't too successful
  - As is typical of data movement, pipelining has better returns than parallelism

- **Reducing RepAgent latency by switching to streaming mode…**
  - Requires that some functions of RA processing now has to move to SRS (e.g. schema association)
  - Adds another queue (SPQ) for binary log record images (with schema info in package)

- **SRS will need more CPU and memory resources**
  - More caches to attempt to reduce need to read from disk
  - Larger pools of buffers for pipelined operations
    - ✓ If a replicated command/object remains in same buffer throughout the pipeline, then a lot of buffers will be needed for the volume
    - ✓ If each step of the pipeline allocates new structure due to command transformation (e.g. package C cmds parsed cmd), then a lot of new allocations will be happening from buffer pool

Sizing and OS Tuning

·T-Shirt Size, OS considerations, etc.

UKSUG
SAP DATABASE & TECHNOLOGY USER GROU

HADR HW Requirements & OS Tuning

- **HW Requirements**
  - SPQ must be on a dedicated SSD
  - If file system for SPQ & SRS partitions, appropriate file system tuning (no journaling, noop IO sched, mount options (e.g. cio), device queue limits, etc.)
  - 10GbE highly recommended (or separate subnet & NIC for HA components)
  - 8-32GB of memory & 4-12 cores depending on sizing on host for HADR components
    - ✓ This is in addition to ASE requirements

- **OS Tuning**
  - Network TCP send/receive buffer should be set
  - NIC interface tuning for queue size
  - File system tuning
    - ✓ No file system journaling, noop IO scheduler, request queue=1024, etc.

SRS & Memory

- **RS 15.x needs a lot more memory**
  - Starting in 15.2, there have been a lot of caches added to aid in transition to in-memory processing
  - ASO option in 15.5 added more memory requirements for large volume/low latency configurations

- **The problem**
  - Long time users were not expecting the memory increases
  - Mostly familiar with 32-bit 2GB limitation and tuning as per 12.6
  - Estimated memory at 2x SQT cache sizes * number of connections
  - Ran a large number of connections per SRS
  - …and usually configured everything at the server level vs. tuning individual connections (except SQT)
  - ….and promptly ran out of memory after upgrading to 15.2+

Always-On sizing

- **T-Shirt sizing**
  - See chart
  - Remember that you may need to aggregate all the databases involved.
- **T-Shirt sizing is a starting point**
  - Merely a suggestion based on anecdotal evidence
  - Your conditions may vary and need less or more
- **Two sites replicate 2GB per hour**
  - Site A - single txn with really wide table
  - Site B - tons of small txns with narrow tables
  - Site B’s config is not optimal for Site A

SRS Sizing	Small	Med	Large	XOLTP
Log growth rate (GB/Hr)	1	3.5	35	>>35
Cmds/Sec	1K	5K	10K	20K+
ASO/HVAR enabled	No	Yes	Yes	Yes
Memory Needed (GB)	4	8	16	24+
Cores for SRS	2	4	8	8+

Note:

The next few slides have a lot of detailed configuration information.

We will not be discussing them in any detail.   You can review them at your leisure back in your office. The goal for this session is to:

- Provide you with a reference for a starting point for configuring your system
- Demonstrate the impact on resources that these configurations may have

LTL RepAgent User configurations for T-Shirt Sizes

SRS configuration parameter	Default	Small	Med	Large	XOLTP
async_parser	off	off	off	on	on
ascii_pack_ibq	off	off	off	on	on
cmd_direct_replicate	off	off	off	on	on
exec_max_cache_size	1048576	1048576	1048576	1048576	1048576
exec_nrm_request_limit	(n/a)	(n/a)	8388608	8388608	8388608
exec_prs_num_threads	(n/a)	(n/a)	(n/a)	2	2
exec_sqm_write_request_limit	1048576	1048576	8388608	8388608	8388608
nrm_thread	off	off	on	on	on
RepAgent User memory consumption	2MB	2MB	17MB	17MB	17MB

CI Receiver configurations for T-Shirt Sizes

SRS configuration parameter	Default	Small*	Med	Large	XOLTP
cap_prs_num_threads	2	1	1	1	1
cap_sqm_write_msg_limit	5000	5000	5000	5000	5000
cap_sqm_write_request_limit	8388608	8388608	8388608	8388608	8388608
ci_batch_packages	1	1	1	1	1
ci_max_cmds	80	80	80	80	80
ci_package_size	1048576	1048576	1048576	1048576	1048576
ci_pool_size	50	50	200	200	400
ci_thread_pool_size	0	0	0	0	0
spq_cache_size	10485760	10485760	10485760	20971520	33554432
spq_data_file_size	1024	1024	1024	1024	1024
spq_save_interval	0	0	0	0	0
spq_write_flush	on	on	on	on	on
CI Receiver memory consumption	68MB	68MB	218MB	228MB	440MB

*(Small) õ In this case, assumption is a small database within HADR or using streaming rep

SQM (inbound & outbound) configurations for T-Shirt Sizes

SRS configuration parameter	Default	Small	Med	Large	XOLTP
block_size	16	16	256	256	256
init_sqm_write_delay	100	25	25	25	25
init_sqm_write_max_delay	1000	100	100	100	100
sqm_async_seg_delete	on	on	on	on	on
sqm_cache_enable	on	on	on	on	on
sqm_cache_size	16	128	16	32	48
sqm_cmd_cache_size	20971520	20971520	33554432	67108864	100663296
sqm_max_cmd_in_block	320	320	2560	2560	2560
sqm_page_size	4	4	4	4	4
sqm_recover_segs	1	10	10	10	10
sqm_write_flush	on	off	off	off	off
SQM memory consumption	1MB	28MB	48MB	96MB	144MB

SQT configurations for T-Shirt Sizes

SRS configuration parameter	Default	Small	Med	Large	XOLTP
dist_sqt_max_cache_size	0	0	0	0	0
sqt_init_read_delay	2000	25	25	25	25
sqt_max_cache_size	20971520	(64MB)	(256MB)	(512MB)	(1024MB)
sqt_max_read_delay	10000	100	100	100	100
sqm_read_timeout	10000	10000	10000	10000	10000
sqm_reader_first	off	off	off	off	off
Inbound SQT memory consumption	20MB	64MB	256MB	512MB	1024MB

DIST & RSI configurations for T-Shirt Sizes

SRS configuration parameter	Default	Small	Med	Large	XOLTP
dist_cmd_direct_replicate	off	on	on	on	on
dist_direct_cache_read	off	off	on	on	on
md_sqm_write_request_limit	1048576	8388608	8388608	8388608	8388608
sts_cachesize	1000	10000	20000	20000	20000
rsi_batch_size	262144	16777216	16777216	16777216	16777216
rsi_fadeout_time	-1	-1	-1	-1	-1
rsi_packet_size	4096	16384	16384	16384	16384
rsi_sync_interval	60	10	10	10	10
DIST & RSI memory consumption	2MB	48MB	96MB	96MB	96MB

DSI configurations for T-Shirt Sizes

SRS configuration parameter	Default	Small	Med	Large	XOLTP
db_packet_size	512	8192+	8192+	8192+	8192+
dsi_bulk_copy	off	on	on	on	on
dsi_bulk_threshold	20	10	10	10	10
dsi_cdb_max_size	1024	(n/a)	1GB	1.5GB	1.5GB
dsi_cmd_batch_size	8192	65536	65536	65536	65536
dsi_cmd_prefetch	off	(n/a)	off	off	off
dsi_compile_enable	off	(n/a)	on	on	on
dsi_compile_max_cmds	10000	(n/a)	100000	150000	150000
dsi_compile_retry_threshold	100	(n/a)	500	500	500
dsi_large_xact_size	100	100000	100000	10000	10000
dsi_max_cmds_in_batch	100	100	100	100	100
dsi_max_xacts_in_group	20	20	20	20	20
dsi_non_blocking_commit	0	10	10	10	10

DSI configurations for T-Shirt Sizes

SRS configuration parameter	Default	Small	Med	Large	XOLTP
dsi_row_count_validation	on	on for standby databases/off otherwise			
dsi_sqt_max_cache_size	0	256MB	1024MB	1536MB	2048MB
dsi_xact_group_size	65536	8388608	8388608	8388608	8388608
dynamic_sql	off	on	on	on	on
dynamic_sql_cache_management	mru	mru	mru	mru	mru
dynamic_sql_cache_size	100	1000	1000	1000	1000
sqt_max_prs_size	(2GB)	(256KB)	(2GB)	(2GB)	(2GB)
dt_apply	1	1	1	2	4
dsi_num_threads	1	1	3	4	4
dsi_serialization_method	1	1	Commit	Commit	Commit
dsi_large_xact_threads	0	0	0	1	1
DSI & DSIEXEC memory consumption	24MB	300MB	4096MB	8192MB	8192MB

Adding up the memory….per connection!!

SRS configuration parameter	Default	Small	Med	Large	XOLTP
RepAgent User memory consumption	2MB	2MB	17MB	17MB	17MB
CI Receiver memory consumption	68MB	(68MB)	218MB	228MB	440MB
SQM memory consumption	1MB	28MB	48MB	96MB	144MB
Inbound SQT memory consumption	20MB	64MB	256MB	512MB	1024MB
DIST & RSI memory consumption	2MB	48MB	96MB	96MB	96MB
DSI & DSIEXEC memory consumption	24MB	300MB	4096MB	8192MB	8192MB
total memory consumption	50MB	500MB	4800MB	9300MB	10240MB

It is best to think of SRS as merely a container - the real tuning is done on the replication 'paths'

- Need to tune the inbound path
- Need to tune the outbound path

Controlling Process Size vs. Memory Configuration (1)

·Many customers have reported ‘memory leaks’ with SRS

- Some were legitimate…..but most were simply due to not understanding OS handling of malloc calls and multi-threaded programming

·All OS’s manage virtual memory using heaps per process

- If there is a single heap for a multi-threaded process, then you could get a lot of mutex contention if the process frequently allocates and deallocates memory
- This is clearly seen on AIX with SAP Replication Server and SAP IQ
  - ✓ On AIX, the default is 1 heap per process. You have to use MALLOCOPTIONS environment variable to configure more.
  - ✓ ASE is generally unaffected by this as it preallocates nearly all memory upfront in a single shared memory segment and then manages the memory internally
    - …..but then you get to see sort of the same problem with certain spinlocks
- To avoid this, most OS’s (except AIX) allocate multiple pools - OS specific how this is done
  - ✓ Since multiple heaps are allocated, this may mean that the process size in the OS could be substantially larger than the DBMS/SRS program configurations
  - ✓ It is not uncommon for the process size to be 10x the SRS configuration

Red Hat Enterprise Linux 6 features version 2.11 of glibc, providing many features and enhancements, including... An enhanced dynamic memory allocation (malloc) behaviour enabling higher scalability across many sockets and cores. This is achieved by assigning threads their own memory pools and by avoiding locking in some situations. The amount of additional memory used for the memory pools (if any) can be controlled using the environment variables MALLOC_ARENA_TEST and MALLOC_ARENA_MAX. MALLOC_ARENA_TEST specifies that a test for the number of cores is performed once the number of memory pools reaches this value. MALLOC_ARENA_MAX sets the maximum number of memory pools used, regardless of the number of cores.

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/6.0_Release_Notes/compiler.html

Controlling Process Size vs. Memory Configuration (Linux)

- **On RHEL, this is controlled via 2 environment variables**
  - ❖ MALLOC_ARENA_TEST
    - ✓ Once the number of memory “arenas” (memory pools) reaches this value, the OS will test for the number of cores.
  - ❖ MALLOC_ARENA_MAX default is 8x cores
    - ✓ The maximum number of memory pools allowed per process
  - ❖ MALLOC_PER_THREAD=1 was used in RHEL 5 (deprecated?)

- **SRS uses a large number of threads**
  - ❖ A typical SRS data path uses 10+ threads, but with CI, it reaches ~20 threads
  - ❖ As the number of connections increase, this increases the number of threads proportionately and likely SRS is running close to MALLOC_ARENA_MAX

- **SRS Tuning**
  - ❖ If you can afford the memory, don’t worry about it
  - ❖ If you want the process size to be closer to the SRS configured size, then you will need to tune this - **BEWARE - DOING THIS MAY DEGRADE SRS PERFORMANCE**
  - ❖ If we assume HT is enabled, we know we have ~2 hardware threads
    - ✓ Optimistic) set MALLOC_ARENA_MAX in RUN file to 2x cores
    - ✓ Pessimistic set MALLOC_ARENA_MAX in RUN file to 1x cores

Not all OS's do things equally

- **Linux pre-allocates heaps up to 8x the number of cores**
  - You might say it is tuned for multi-threaded apps out of the box
  - However, it suggests that a lot of processes that use virtual memory will increase swapping

- **AIX only allocates a single heap**
  - This is more friendly for multiple processes - e.g. a more fair/balanced approach
  - But then multi-threaded servers suffer mightily

By default, the malloc subsystem uses a single heap, or free memory pool.

However, it also provides an optional multiheap capability to allow the use of multiple heaps of free memory, rather than just one.

The purpose of providing multiple-heap capability in the **malloc** subsystem is to improve the performance of threaded applications running on multiprocessor systems. When the **malloc** subsystem is limited to using a single heap, simultaneous memory-allocation requests received from threads running on separate processors are serialized. The **malloc** subsystem can therefore only service one thread at a time, resulting in a serious impact on multiprocessor system performance.

With malloc multiheap capability enabled, the **malloc** subsystem creates a fixed number of heaps for its use. It will begin to use multiple heaps after the second thread is started (process becomes multithreaded). Each memory-allocation request will be serviced using one of the available heaps. The **malloc** subsystem can then process memory allocation requests in parallel, as long as the number of threads simultaneously requesting service is less than or equal to the number of heaps.

If the number of threads simultaneously requesting service exceeds the number of heaps, additional simultaneous requests will be serialized. Unless this occurs on an ongoing basis, the overall performance of the **malloc** subsystem should be significantly improved when multiple threads are making calls to the **malloc** subroutine in a multiprocessor environment.

https://www.ibm.com/support/knowledgecenter/ssw_aix_61/com.ibm.aix.genprogc/malloc_multiheap.htm

AIX Mandatory Tuning Settings

- **See previous discussion about virtual memory and process size**

- **On AIX, you must add the following ENV settings to the SRS RUN_SERVER file:**

```
AIXTHREAD_MUTEX_FAST="ON"
YIELDLOOPTIME=0
SPINLOOPTIME=500
MALLOCOPTIONS="threadcache,multiheap:8,considersize"
export AIXTHREAD_MUTEX_FAST YIELDLOOPTIME SPINLOOPTIME MALLOCOPTIONS
```

- **Notes**

- This will make the SRS process look up to 10x higher in OS memory vs. configuration
- The MALLOCOPTIONS multiheap setting should be close to the number of database connections in SRS.
  - ✓ As a result, anything less than 4 is not recommended. 4 may be used with BusinessSuite on ASE or other similar situations where there are only 1-2 databases being replicated (other than master and the CID)
  - ✓ On really high volume situations, you may need more heaps than connections - up to 2x the number of processor cores
  - ✓ Any value between 1 and 32 is supported for multiheap - does not have to be even power of 2, etc.
- Considersize attempts to reduce the process size by using the next heap that can handle the mallocrequest. However, as a result, it can be slower.
  - ✓ If memory is not a factor, might not want to use this option and just use the default allocation algorithm.
- Threadcache preallocates multiple same sized chunks along with a larger chunk for future if malloc <4K.
  - ✓ This is key to reducing contention on global mutex- so don't disable.

Impact of Virtual Machines/LPAR’s

- **VM’s/LPARS can have a significant impact on SRS throughput**
  - Comparison between a single LPAR using a shared processor pool and a dedicated processor pool showed a 30% gain in performance with dedicated processor pool
  - The point is that if the SRS host image is sharing resources at all - the hypervisor still has to run to check to see if anything needs those resources

- **On the flip side, tests with VM’s have shown greater performance in some cases**
  - E.g. when SRS running on large numbers of cores, it may end up running across a bunch of different cores/sockets vs. running on the same socket

- **Lessons:**
  - Using virtualization to constrain SRS does have some benefits
  - But if doing so, best technique is to dedicate some number of processors vs. sharing
  - A lot will depend on which virtualization software you use

Common Performance Issues

- ….and solutions

UKSUG
SAP DATABASE & TECHNOLOGY USER GROU

Common Performance Issues

·**Missing Primary Keys vs. HVAR**

- Impacts inserts and updates on ALL tables without primary keys
- SRS is forced to use dsi_top1_enable
- RS MC will show a lot of TPF failures as to why HVAR was not used.
    - ✓5121 DSI: Transactions non-compilable for TPF (TranNonHQForTPF)

·**SPQ latency w/ Always-On**

- Typically caused by IBQ/OBQ being full - so check this first
- If IBQ/OBQ not involved, issue is SPQ reader not keeping up with writer
    - ✓Most likely cause is CAPPRS threads.
        - By default there are 2 per connection (parallel)
        - This is a CPU intensive routine - you may need more threads with wider tables on high volume connections
    - ✓Second cause is SQM Write Waits
        - 74015 CAPTURE: Write wait time (WriteWaitsTime) – The amount of time XLAT waiting for the SQM Writer thread to drain the number of outstanding write requests to get the number of outstanding bytes to be written under the threshold.
        - This points to a slow SQM Writer - IBQ needs to be sped up somehow (depends on implementation what the cause is)

Common Performance Issues

·Inbound Queue latency (IBQ)

- Typically due to OBQ being full
- Use RS MC to verify where DIST is spending the time
  - ✓If time is mostly reading, then dist_direct_cache_read is off or it is reading from disk (txn removed from cache)
  - ✓If time is spent parsing, then direct replication is disabled
  - ✓If time is spend in TD/MD, then problem is outbound queue (SQM) or OS memory contention
- By default the dist_sqt_max_cache_size isn't set, so it inherits the default which isn't set, so it inherits the default for sqt_max_cache_size
  - ✓Check sqt_max_cache_size
  - ✓Check for large transactions/transactions removed
  - ✓Consider increasing the connection's dist_sqt_max_cache_size if txns are removed

·Outbound Queue latency (OBQ)

- The main issue is that the DSI/DSIEXEC have defaults set for a lot of really small transactions on a small schema (e.g. TPCC).
- Real world sizing for DSI SQT cache, CDB sizes, etc. for real world requirements.
  - ✓

If latency persists

Inbound queue (due to DIST latency w/ ASO features enabled)

- Enable parallel distribution

Outbound queue (due to RDB slowing HVAR)

- Enable parallel apply (dt_apply) for the connection

Both of these require some SRS reconfiguration

- First we have to configure some worker threads

  ```
  Configure replication server set worker_thread_num to 'n'   -- n between 2 and 32 - suggest 10
  ```

  ✓ Reboot SRS
- Then we can configure parallel distribution

  ```
  Suspend distribution from <source_ds>.<source_db>
  Alter connection to <source_ds>.<source_db> set parallel_dist to 'on'
  Resume distribution from <source_ds>.<source_db>
  ```
- Distributed Apply on next slide
- Only do either of these *after* careful consultation with SAP wrt any considerations

  ✓ See next slide and impact of incremental apply vs. distributed apply as an example of why

  ✓

  ✓

Distributed Apply (dt_apply)

- **Configure distributed apply**
  - SRS configs
    - **`Suspend connection to <dest_ds>.<dest_db>`**
    - **`Alter connection to <dest_ds>.<dest_db> set dt_apply to ‘4’`**
    - **`Resume connection to <dest_ds>.<dest_db>`**
  - ASE configs
    - **`sp_configure 'enable dtm', 1`**
    - **`sp_role "grant", dtm_tm_role, <maintenance user>`**

- **Note that with distributed apply, you can not use incremental compilation**
  - This is on by default for HADR/Always-on
    - ✓turn on rsfeature, rsfeature_hq1 and watch for “INCREMENTAL CDB flush”
    - ✓This presents a problem - if incremental apply was being used, then you will need to size the DSI SQT cache and CDB to be large enough to hold the largest txn from the source….if you don’t, it will go language mode (ouch ouch ouch ouch ouch ouch!!!!)
  - Enabling incremental apply with trace
    - **`trace {"on"|"off"},rsfeature,rsfeature_hq_incr_cmpl_on`**
  - …or in the config file
    - **`trace=rsfeature,rsfeature_hq_incr_cmpl_on`**

SRS Roadmap

- …for the next 18 months

UKSUG
SAP DATABASE & TECHNOLOGY USER GROU

Long term direction

·**Move from LTL to CI interface**

- ❖ Resolve RA latency and allow sync/near sync/async replication
  - ✓Initial focus was for HADR
- ❖ CI interface needs completion to match LTL for custom replication
  - ✓Request & Applied Functions
  - ✓SQLDML replication
  - ✓Multi-Path Replication (MPR)

·**Focus will be on ASE replication**

- ❖ ASE ASE
- ❖ ASE Hetero and Hetero – ASE
- ❖ Hetero Hetero is SAP Enterprise Information Management (EIM)

·**Align releases/schedules with ASE**

- ❖ Approximately 1 release per year with PL’s every 6 months
- ❖ SRS 15.7.1 sp1xx support for ASE 15.7 with EOL 31 Dec 2020
- ❖ SRS 15.7.1 sp3xx C becomes SRS 16.x and supports ASE 16 through EOL (2025+)

SAP Replication Server (SRS)

Product road map overview - key themes and capabilities

Today

·SAP ASE Support
- SAP ASE 16 support
- Zero data loss (ZDL) synchronous replication for custom apps, including ability to handle schema changes
- SAP Business Suite on SAP ASE Disaster Recovery
- Support for SAP ASE Cluster Edition with virtual IP support

·Cloud and Big Data Support
- Certification on AWS

·Core Enhancements Support
- RAH – Multi-tenant support
- SAP HANA Replication for Business Suite on Oracle, DB2 and MSSQL with Cluster and Pool table support
- HANA to HANA out-bound replication for real-time reporting (including BS and non-BS)
- Heterogeneous DDL replication support to SAP HANA
- Enhanced management and monitoring support with Replication Management Agent (RMA)
- Data Assurance

Planned Q3/2016

·ASE Support
- Zero data loss (ZDL) synchronous replication for Business Suite apps, including ability to handle schema changes

Planned Q4/2016

·ASE Support
- Improve materialization for HADR with concurrent dump
- **External replication into and out of HADR Cluster**
- TCO optimizations – memory/disk
- SAP ASE/SRS Compatibility and interoperability

·Core Enhancements Support
- Certification of latest/newer versions of third party database (i.e. – SAP HANA, Oracle)

Planned Q2 2017

·SAP ASE Support
- Support for SAP HANA and SAP ASE interoperability
- Performance enhancements (T-shirt sizing, large transactions)
- **CI mode support for stored proc & SQLDML replication**

·Cloud and Big Data Support
- Replication in AWS Marketplace and other providers
- Replication for SAP ASE in HCP

·Core Enhancements Support
- CCL support

Red - key HADR/CI mode enhancements

Green – may move to 2017

Blue – under investigation

This is the current state of planning and may be changed by SAP at any time.

Questions???
Follow-up…
Discussion…
Clarifications…

UKSUG

SAP DATABASE & TECHNOLOGY USER GROUP

THANK YOU

For more information on SAP ASE 16 visit:

www.sap.com/ase

http://help.sap.com/ase1602/

https://ideas.sap.com/SAPASE