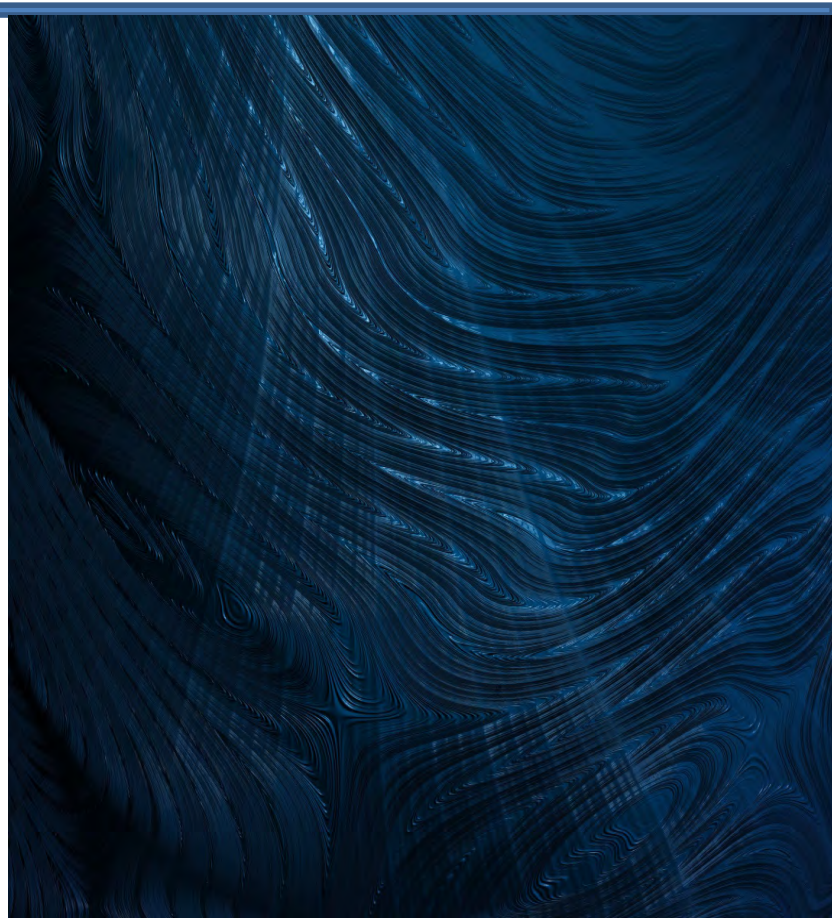
ASE Tempdb Performance Tuning
The definitive reference

UKSUG
SAP DATABASE & TECHNOLOGY USER GROUP

Joe Woodhouse
29 July 2020

PRIMA DONNA CONSULTING

Contents



About the author

- Sybase Australia 1996 – 2003
- Freelance consultant via Prima Donna Consulting for 17+ years
- Works exclusively with ASE, IQ, and Replication Server for 24+ years
- Based in London, UK and Melbourne, Australia
- International Sybase User Group Board of Directors since 2010
- UK Sybase User Group Board of Directors since 2019
- Not a lawyer – no charge for emails!

About this presentation

- Sybase User Group masterclasses in three countries, 2009 - 2018
- International Sybase/SAP conferences in 2011, 2012, 2013, 2014, 2015
- Now revised and current as of ASE 16.0 SP03 PL08 HF2 (July 2020)
- This is intended to be the definitive reference for tempdb performance
  - Prizes awarded if you have an idea that doesn't appear here!
    - Prizes awarded in 2009, 2010, and 2013 ☺
    - No-one has managed it since (including JT, PT, and RV)
    - Maybe you'll be the one
- This version now has 73 different ways to make tempdb go faster

Why tempdb?

- Tempdb is often taken for granted
- It is common for tempdb to be the busiest database
  - Take a look at I/Os per database, you may be surprised
  - tempdb might do more I/O than everything else combined
    - Perhaps even 10 or 100x more than everything else combined
- It's more complicated than that though
  - Tempdb I/Os can flood caches and storage systems
  - Making tempdb faster (or doing less work) has a flow-on effect everywhere

Performance & tuning philosophy

- Usually we think of tuning as making something go faster
  - And yes certainly that's an important part of it
- Sometimes it is about doing less work
  - Sometimes it is about not doing the work at all
    - Can't get faster than zero elapsed time!
- Sometimes it is about removing a blocker or reducing contention
- Almost any tuning for I/O benefits tempdb
- There are also some tempdb-specific tuning measures
- The world of DevOps: you need to know hardware and OS now

Some conventions used here

Do something faster

Do less work or fewer I/Os

Unblock something blocked, or reduce contention

Needs a separate license (or ASE Platform Edition)

Storage hardware & disk layout

14 techniques

Storage hardware & disk layout (1 of 14)

- (#1) Update all storage hardware firmware & drivers
  - Yes, really
  - Excellent odds that this needs to be done
  - Cases exist where firmware updates increased performance

Storage hardware & disk layout (2 of 14)

- (#2) Review controller cache settings
  - Yes, really this too
  - Excellent odds that this also needs to be done
  - Some HBA and/or SAN controllers default to disabled write cache

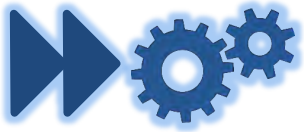
Storage hardware & disk layout (3 of 14)

- (#3) Dedicated storage for tempdb% devices
  - At a minimum, tempdb% devices should be on their own LUNs
  - Don't share tempdb% devices with data, log, binaries, or dump area
  - Reduces contention at LUN and mount point levels
  - Allows tempdb-specific tuning at LUN and mount point levels
  - Allows SSD for tempdb even if can't afford it for all ASE devices
  - Allows NVMe/PCIe for tempdb even if can't afford it for all ASE devices

Storage hardware & disk layout (4 of 14)

- (#4) Local storage for tempdb% devices
  - Extend #1 above and keep all tempdb% devices on local disks
  - Reduces load on the SAN
  - Reduces network traffic to/from the SAN
  - Reduces contention on HBAs
  - No recoverability issues for tempdb
    - But there **are** availability issues, so we want this at least mirrored
  - Allows possibility of NVMe/PCIe motherboard storage for tempdb%
    - Probably can't afford 100% NVMe for entire ASE; best ROI

Storage hardware & disk layout (5 of 14)

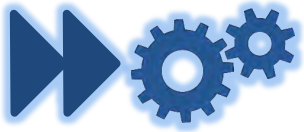


- (#5) Optimise RAID settings
  - RAID level is critical for database performance
    - RAID 10 is best, then 0+1, then 5, then 6
    - Significant write penalties with RAID 5 and 6
    - … and tempdb is **always** writing
  - RAID block size *must* be aligned with ASE page size and filesystem block size
    - Make an integer multiple of those
    - Too small or not aligned = I/O explosion
    - Too large = hotspots where multiple ASE I/Os directed to same RAID block
  - Changing either of these is a physical rebuild

Storage hardware & disk layout (6 of 14)

- (#6) Disable SAN replication for tempdb%
  - Disk replication imposes significant write penalty if synchronous
    - But there are no recoverability concerns for tempdb%, so this is wasted
    - At a minimum, make asynchronous for tempdb% LUNs
    - Better: disable completely for tempdb% LUNs
    - … not possible unless tempdb% has dedicated LUNs

Storage hardware & disk layout (7 of 14)



- (#7) Disable snapshot backups for tempdb%
  - Small performance penalty to use disk snapshot backups
    - But there are no recoverability concerns for tempdb, so this is wasted
    - Disable completely for tempdb% LUNs
    - … not possible unless tempdb% has dedicated LUNs

Storage hardware & disk layout (8 of 14)

- (#8) Disable filesystem backups for tempdb%
  - Moderate performance penalty to take file system backups while in use
    - But there are no recoverability concerns for tempdb, so this is wasted
    - Disable completely for tempdb% LUNs
    - … not possible unless tempdb% has dedicated LUNs
  - Eliminating file backups of file devices also reduces contention on files & LUNs
  - Arguably we don't want filesystem backups of **any** ASE devices
    - You can't restore them as they weren't taken while devices were quiesced

Storage hardware & disk layout (9 of 14)

- (#9) Logically separate tempdb data & log
  - It's common to have tempdb% databases mix data & log
  - This creates several issues
  - Pre 11.9.2: no user log cache (ULC) in any database where data & log are mixed
    - Tempdb is always writing log records, so this is a significant issue
  - Post 11.9.2: still has implications for I/O streaming
  - Post 16.0: user log queue behaviour improves this but still worthwhile doing
  - Interleaved data & log pages in cache frustrate large/sequential reads
  - Be 100% mixed or 100% separated
    - CR 781100, not fixed, slows tempdb recovery if these are combined

Storage hardware & disk layout (10 of 14)

- (#10) Physically separate tempdb data & log
  - It's not enough simply to logically separate tempdb data & log
  - Should be isolated to separate LUNs and/or drives
  - Reduces contention at the LUN and mount point levels
  - Allows dedicated disk tuning for tempdb data vs. tempdb log
    - e.g. dsync off for both; directio on for data; directio off for log

Storage hardware & disk layout (11 of 14)

- (#11) Remove tempdb segments from master device
  - This is **not** the same as deleting the fragments! DO NOT DO!
  - Instead update sysusages set segmap=0 where dbid=2 and lstart=0
  - And reboot ASE
  - Fragment of tempdb on master will still exist but be unused
  - Important because master device will not be optimised for tempdb
    - First fragment will have all system tables on it; we want these to be fast
    - Older versions of ASE using dsync=true for master device!
    - Won't be on dedicated tempdb hardware
    - Won't benefit from LUN, mount point, and device tuning

Storage hardware & disk layout (12 of 14)

- (#12) Separate tempdb system and default segments
  - Advanced technique: use multiple tempdb data devices
  - Use sp_dropsegment for dedicated devices each for system and default
  - System segment is used for ASE internal worktables (+ system tables)
  - Default segment is used for all user-created tables (#temp & tempdb..temp)
  - Logical separation reduces device contention
  - Physical separation reduces LUN and mount point contention

Storage hardware & disk layout (13 of 14)

- (#13) Multiple tempdb devices per data segment
  - Advanced technique: use multiple tempdb data devices **per segment**
  - Some ASE parallelism decisions are made by default based on # of devices
    - Parallel create index
    - Parallel query
    - Repartitioning
  - Multiple ASE devices per system and default segment = more parallel degree
  - Of course ASE must be configured for parallel correctly (and at all)
  - ASE 16.0+: avoid_implicit_data_repartitioning optimiser criterion partial fix
    - Not on by default in any optimization goal! Must manually set!

Storage hardware & disk layout (14 of 14)

- (#14) Raw vs. file; choice of filesystem
  - This argument goes back 25+ years in Sybase
  - Difference today between raw partition and well tuned f/s is < 5%
  - In some scenarios f/s can outperform raw (e.g. buffered writes for tempdb log)
  - Just use file, it's easier to manage and allows some tuning tricks
  - Windows: Always take NTFS
  - RHEL: Pros and cons for ext4 vs XFS
    - Maybe XFS for all ASE devices except tempdb; ext4 for tempdb?
    - ZFS has licensing issues on Linux, not quite as performant as ext4 or XFS
    - btrfs and f2fs probably not ready for PROD yet

Operating system tuning

8 techniques

OS tuning (1 of 8)

- (#15) DON'T DO: tempdb on tmpfs
  - Ancient advice, has not been true since 1996 (SQL Server 11.0)
  - Intention was to nail tempdb into memory
  - Far better ways of doing this today
  - DON'T DO THIS EVER
  - The memory is **always** used better within ASE
  - If you're not convinced yet: all I/O to tmpfs is synchronous

OS tuning (2 of 8)

- (#16) DON'T DO: tempdb on RAMdisk
  - Ancient advice, has not been true since 1996 (SQL Server 11.0)
  - Intention was to nail tempdb into memory
  - Far better ways of doing this today
  - DON'T DO THIS EVER
  - The memory is **always** used better within ASE
  - If you're not convinced yet: all I/O to RAMdisk is synchronous
  - (Encountered this in July 2020 so it still needs to be said)

OS tuning (3 of 8)

- (#17) Update the OS with all patches
  - Yes, really
  - Excellent odds that this needs to be done
  - Cases exist where OS updates **significantly** increased performance
    - Multiple known Solaris bugs with async I/O
    - Multiple known RHEL bugs with POSIX and KAIO
  - Ensure you have every OS package recommended by the ASE install docs

OS tuning (4 of 8)

- (#18) Tune the OS kernel
  - Documentation doesn't emphasise performance
  - Platform-specific, but here are some critical RHEL recommendations:
    - fs.aio-max-nr = *max online engines* * (*disk i/o structures* + *max async i/os per engine*)
      - (*disk i/o structures* = 1048576)
      - (*max async i/os per engine* = 8192 or 16384)
    - fs.file-max = 6291456 # double default
  - If using a SAN remember that network tuning is effectively also disk I/O tuning

OS tuning (5 of 8)

- (#19) I/O elevators (RHEL)
  - RHEL has three methods to schedule I/Os
    - cfq = Completely Fair Queuing (default)
    - deadline = Don't allow anything to take too long
    - noop = no scheduling, do them in the order they were issued
  - SANs will already reorder I/Os so there is no sense in reordering twice
    - Use noop for all devices & LUNs on a SAN
  - For local disks, deadline outperforms cfq and often outperforms noop
  - Can be tuned per host, or per LUN (all lvols and mount points must be the same)
    - Per LUN is not persistent and must be scripted after every host reboot

OS tuning (6 of 8)

- (#20) Disk queue depths (RHEL)
  - RHEL limits the number of outstanding I/O requests per disk
  - Default of 128 is too small for any real ASE
  - Set it instead to 1024
    - echo 1024 > /sys/block/sdb/queue/nr_requests
  - Must be tuned per LUN (all lvols and mount points must be the same)
    - Per LUN is not persistent and must be scripted after every host reboot
    - Now you see why we want tempdb on dedicated storage hardware…

OS tuning (7 of 8)

- (#21) Filesystem tuning per mount point
  - ext4
    - noatime, nodiratime
    - Disable filesystem journaling, yes really, for all ASE devices
      - Not needed in ASE and can give a 400% boost to I/O
      - Sysadmins do not believe it is unneeded and hobble ASE on ext4
      - tune2fs -O ^has_journal
    - If you cannot win the journal fight, add barrier=0, data=writeback
  - XFS
    - noatime, nodiratime, nobarrier, logbufs=8

Time out – more on journaling

- Sysadmins **really** don't like disabling ext4 journaling
  - Truly it isn't needed, and benchmarks show 400% faster without
  - Sysadmins probably OK with barrier=0, data=writeback
    - These disable data journaling and use journaling only for file metadata
  - ASE preallocates all devices, so file sizes cannot change because of an I/O
    - (Unless you use skip_alloc = true in disk init)
  - File metadata will therefore never change or be out of sync with file contents
  - Therefore ext4 journaling is never needed for any ASE device
    - Not just tempdb! Any ASE device!
  - Refer your sysadmins to me, maybe I can convince them

OS tuning (8 of 8)

- (#22) Filesystem block size
  - Should be the same as the ASE page size
  - If smaller, then one ASE I/O becomes many
  - If larger, then multiple ASE I/Os are sent to the same I/O
    - If these weren't sequential then some of the I/O was unnecessary
  - This is an argument for a larger ASE page size BTW
  - Changing filesystem block size is destructive and needs a rebuild
  - Changing ASE page size is a pain in the neck, no dump & load
    - I saw two 2020 clients run out of space in a 2Kb page ASE (max 8Tb)

ASE server-level tuning

22 techniques

ASE server tuning (1 of 22)

- (#23) dbcc tune (deviochar, [...])
  - An oldie but still a goodie (SQL Server 11)
  - Increases batch size of I/Os the ASE housekeeper writes to this device
  - Default is 3, can go much higher if your storage system is up to the job
  - Not very valuable on its own, but useful when combined with other tricks
    - i.e. if using local NVMe for tempdb log using file system buffer cache
    - Doesn't make sense to combine with HK ignore cache (more on this later)

ASE server tuning (2 of 22)

- (#24) *number of sort buffers*
  - SQL Server 11.0, set with sp_configure
  - Makes some sorts faster
    - create index, update statistics
    - But hopefully you do both of those on all your temp tables…
    - … more on this later
  - Caution: setting above 10,000 increases procedure cache requirements
  - But if you have plenty of memory, set to maximum value 32,767

ASE server tuning (3 of 22)


- (#25) *user log cache size*
  - SQL Server 11.0, set with sp_configure
  - Batches up writes to transaction logs, reduces contention on last log page
  - All ASE benefits from this, and tempdb is no exception
  - Tempdb transaction logs run very hot, think about it
    - Every row written to tempdb must be logged at least minimally
    - Every row written to tempdb will also be deallocated at least minimally
    - Every write will also be logged
  - sp_sysmon will help you size this, but for sure you want larger than default

ASE server tuning (4 of 22)

- (#26) logiosize
  - SQL Server 11.0, set with sp_logiosize per database
  - ASE will issue writes to transaction logs in this size
  - … if a buffer pool of that size exists in the cache that syslogs is using
  - Set to maximum for tempdb, as its logging is mostly batch rather than OLTP
    - Maximum = 8 pages = 8 * ASE page size
    - Make sure there is an 8-page buffer pool in cache used by tempdb%..syslogs

ASE server tuning (5 of 22)

- (#27) named cache for tempdb%
  - SQL Server 11.0, use sp_cacheconfig and sp_bindcache
  - This is old news but worth repeating because few get it right
  - Fundamental principle of caching: any one objects can only use one cache
    - If it appears in memory twice we tend to call that corruption
    - Therefore splitting one large default data cache into d.d.c. + tempdb_cache
    - … means every object now has less cache memory to use

Time out (1 of 2)

- So you want to cache tempdb
  - Named cache for tempdb is the luxury of "enough memory"
    - Suggest you don't even think about it if under 32G ASE max memory
    - Default data cache should usually have at least 75% of all data cache
    - But Joe, didn't you say tempdb% might have 10x more I/O than all else?!
  - Tempdb doesn't really tend to see a lot of reuse
    - Most tempdb use is private to one session
  - Caching is all about reuse
    - Most tempdb cache use is "MRU replacement strategy"
    - Meaning the cached pages probably won't be reused anyway

Time out (2 of 2)

- So you want to cache tempdb
  - tempdb% cache(s) maybe shouldn't be relaxed (ASE 11.5)
    - Hit rates are unlikely to be very high, and there is a lot of turnover
  - If not relaxed, then not lockless either (ASE 16.0), for the same reasons
  - Probably do want tempdb% cache(s) partitioned though (ASE 12.5)
    - Suggest next power of 2 greater than *max online engines*
  - Probably want 35%-40% in 8-page buffer pool
    - Set tempdb%..logiosize to this size also
    - No point having a third or fourth buffer pool size
  - Set cache status = HK ignore cache in .CFG file (ASE 15.0)

ASE server tuning (6 of 22)

- (#28) named cache for tempdb%..syslogs
  - Advanced technique: can't directly bind any tempdb% system table
    - So bind model..syslogs to this cache, and restart ASE
  - Can be small, 1Gb almost certainly enough
  - Definitely want cache set to logonly (SS 11.0)
  - Probably want it relaxed (ASE 11.5) or lockless (ASE 16.0, supersedes relaxed)
  - Probably want it partitioned (ASE 12.5), next power of 2 > *max online engines*
  - Definitely only want two pool sizes, and give 95% to 8-page pool
    - Don't forget to set logiosize to this size!!
  - Set cache status = HK ignore cache in .CFG file (ASE 15.0)

ASE server tuning (7 of 22)

- (#29) dsync and directio
  - dsync introduced in ASE 12.0, and directio in ASE 15.0
  - Definitely want dsync=false on every ASE device
  - Usually want directio=true on every ASE device, but…
    - … if separating tempdb% data and log, maybe set directio=false for log
  - The idea is when we don't care about recoverability, leverage filesystem cache
  - Probably not for tempdb% data since we are reading that a lot
  - But tempdb% log is write and forget
  - We really wish we could set sp_dboption "delayed commit" for tempdb
    - But we can't, not even using the model trick

ASE server tuning (8 of 22)

- (#30) Multiple tempdb databases (ASE 12.5.0.3)
  - Allows logical and physical separation of users' and applications' tempdb use
  - Caution: don't just carve up one large tempdb into smaller ones
    - Any one session can only use one tempdb database
    - Don't make each one too small
    - Rule of thumb: if dividing tempdb users by 2, set each tempdb to 75% size
    - If dividing by 3, set each tempdb to 67% size
  - Tip: reserve one tempdb just for DBAs in case users blow up tempdb%
  - Tip: shrink built-in tempdb to something tiny and don't ever use it!
    - Some options cannot be set for dbid=2 but can be set for other tempdb%

ASE server tuning (9 of 22)

- (#31) *max buffers per lava operator* (ASE 15.0)
  - Similar to *number of sort buffers* but affects other operations
    - Affects all other sorts, and hashes too
  - Memory is stolen from whatever cache tempdb uses
    - Default 2,000 is too small
    - Increase to 16K, 32K, or 64K (maximum) if you have plenty of cache

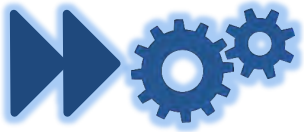
ASE server tuning (10 of 22)

- (#32) Disable compatibility mode (ASE 15.0.2)
  - I thought this was dead like disco but I found it at a 2020 client
  - If enabled, ASE uses the 12.5.4 optimiser
  - Yes even in ASE 16.0
  - There is no place for it anywhere anymore
  - Check whether you have it on, and if so, start testing with it off
  - This is a general ASE issue but absolutely it affects tempdb too

ASE server tuning (11 of 22)

- (#33) Speaking of ASE 15.0.2…
  - These are internals and not configurable, but worth mentioning
  - Row level locking for all system tables, very important for tempdb
  - APL page splits are asynchronous now, also important for #temp tables
    - … since you probably don't set a locking scheme; more on this later

ASE server tuning (12 of 22)

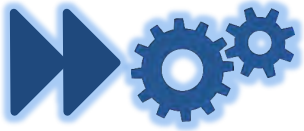


- (#34) Relocated joins on CIS proxy tables
  - Set using sp_serveroption
  - CIS proxy tables are like a shortcut or symlink to a table in another server
  - Joining local tables to proxy tables pulls the remote data and does a local join
  - If remote table is also in ASE, set the relocated joins server option
  - This does as much as possible in the remote server to reduce rows first
  - It will reduce network traffic
  - It will **significantly** reduce tempdb I/Os

ASE server tuning (13 of 22)

- (#35) *session tempdb log cache size*
  - Set using sp_configure, as of ASE 15.0.2
  - Replaces *user log cache size* for tempdb% databases
  - Generally want this set high; recommend minimum size 16 pages
    - So that's 32K in a 2K page ASE, etc.
    - Can go to 32 or even 64 pages if tempdb used heavily

ASE server tuning (14 of 22)

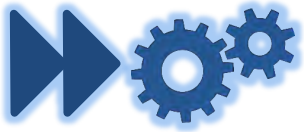


- (#36) Deferred compilation (ASE 15.0.3)
  - Better answer to problem solved by "... with recompile"
  - Allows procedures and triggers to know details of #temp tables
  - Enabled by default; keep it enabled
  - Can be enabled/disabled per procedure as of ASE 16.0 SP03 PL06
  - Very useful when combined with other tricks we'll see later

ASE server tuning (15 of 22)

- (#37) ASE optimizer level (ASE 15.0.3 ESD#1)
  - Similar to compatibility mode, but finer grained
  - A shortcut for enabling/disabling individual criteria by version they were added
  - Default setting of "ase_default" means "run like ASE 15.0.3 ESD#1"
  - Misses out on many optimiser improvements; many affect tempdb
  - Strongly recommend "ase_current" which means "the version running"
  - Also worth mentioning: can be saved as part of a custom opt_goal

ASE server tuning (16 of 22)

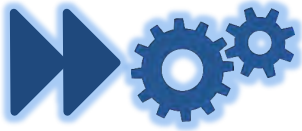


- (#38) *number of pre-allocated extents* (ASE 15.5)
  - Set with sp_configure
  - This goes back to SS 11.0, where legal values were 1-31; used only for bcp
  - Extended in ASE 15.5 to add value 32
  - 32 has a special meaning: will now be used for other commands
  - … including select into
  - Significantly improves select into performance
  - Will require more tempdb space (minimum allocation is now 256 pages)

ASE server tuning (17 of 22)

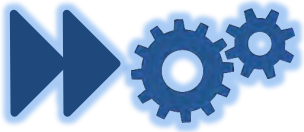
- (#39) *optimize temp table resolution* (ASE 15.7 SP100)
  - Formerly trace flag 299, now set with sp_configure
  - Prevent stored procedure recompiles when they mention an outside #temp table
  - They now look up #temp tables by name, not ID
  - This will cause issues if you reuse #temp table names for different schemas

ASE server tuning (18 of 22)



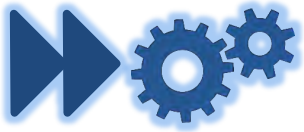
- (#40) *max utility parallel degree* (ASE 15.7 SP100)
  - Set with sp_configure
  - Server-wide limit on parallel degree for update stats and create index
  - Useful as some optimiser techniques create indexes on worktables
  - Useful as we should explicitly do both on most #temp tables
    - More on this later

ASE server tuning (19 of 22)



- (#41) *scavenge temp objects* (ASE 15.7 SP100)
  - Set with sp_configure
  - 1 = disabled, 0 = enabled, default = 1 (disabled)
  - Enable (set to 0) to remove temp tables from cache once no longer used
  - Greatly improves tempdb cache performance

ASE server tuning (20 of 22)

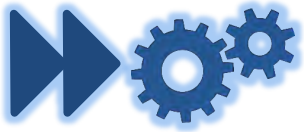


- (#42) Bulk inserts (ASE 15.7 SP130)
  - Set with sp_configure "enable bulk inserts", 1
  - Can also be set per session
  - Use bulk inserts (minimally logged) if criteria are met; but many restrictions
  - Partial list of restrictions:
    - » Must insert more than 8 pages of data
    - » Must be DOL table
    - » Cannot be to a #temp table (!!)
  - Restrictions can be addressed using other tricks we haven't seen yet

ASE server tuning (21 of 22)


- (#43) *user log cache queue size* (ASE 16.0)
  - Set with sp_configure
  - When enabled, upgrades ULC behaviour to use multiple ULC cachelets
  - Reduces contention and increases batching
  - Still necessary to tune *user log cache size* and *session tempdb log cache size*

ASE server tuning (22 of 22)

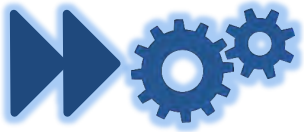


- (#44) *enable select into in tran* (ASE 16.0 SP03)
  - Set with sp_configure
  - Does what it says on the tin: enables select into in tran
  - Useful because procs previously disqualified for replication can now replicate
  - … like select into #temp table
  - Anything that allows more select into improves performance
  - Recall also that as of ASE 16 this doesn't necessarily break dump tran

ASE database-level tuning

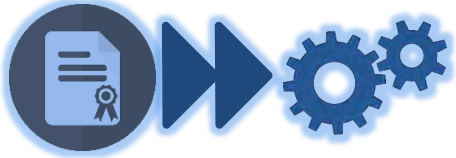
6 techniques

ASE database tuning (1 of 6)



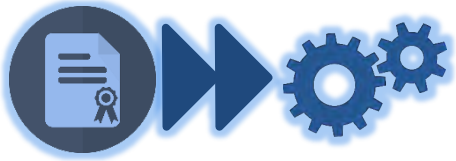
- (#45) dml_logging = minimal (ASE 15.5)
  - Set per database with alter database
  - Forces all inserts/updates/deletes to be minimally logged
  - Cannot be set on built-in tempdb dbid=2
    - No, not even by setting on model
    - Can hack sysdatabases..status3 but I can't confirm behaviour changes
  - **Can** be set on all manually created temporary databases!
    - Now you see why I suggest just not using built-in tempdb for any real work
  - Briefly requires database to be in single-user mode
    - Can also be set per session, more on this later

ASE database tuning (2 of 6)



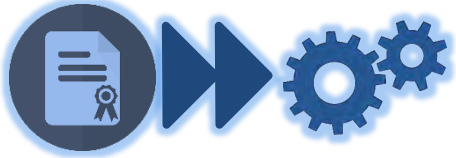
- (#46) Relaxed durability database (ASE 15.5)
  - Set per database with alter database
  - Automatically on for all temporary databases including dbid=2
    - Cannot be unset for temporary databases
    - Useful for scratch and staging databases also
    - However requires IMDB or Memscale license for non-temporary databases
    - Briefly requires database to be in single-user mode

ASE database tuning (3 of 6)



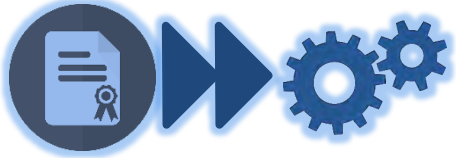
- (#47) IMDB tempdb (ASE 15.5)
  - Set per database with create database
  - This is not the same as fitting 100% of tempdb% database in named cache
  - Many internal differences; most good, some maybe not
    - Important one: large I/O disabled in IMDB (because there are no PIOs)
    - Important one: bulk inserts disabled in IMDB
  - Requires IMDB/Memscale license
  - Test carefully, you win some, you lose some, you may lose overall for tempdb%
  - Improved in ASE 15.7 SP100
    - IMDB tempdb sorting and update stats now done in default data cache

ASE database tuning (4 of 6)



- (#48) Data compression (ASE 15.7)
  - enable with sp_configure
  - Set per database with create or alter database
  - Two forms, row and page compression, page is better
  - Can be set on every tempdb% database
  - Yes, there is a small CPU overhead…
  - … but I/O is the slowest part of any computer, and compression = fewer I/Os
  - Requires license
  - Should also sp_configure "compression info pool size" to at least 128Mb
    - Go as high as 1Gb for VLDB

ASE database tuning (5 of 6)



- (#49) Index compression (ASE 16.0)
  - enable with sp_configure
  - Set per database with create or alter database
  - Only page compression available for indexes
  - Can be set on every tempdb% database
  - Again, fewer I/Os = faster, also fewer pages in cache
  - Requires license
  - Should also sp_configure "compression info pool size" to at least 128Mb
    - Go as high as 1Gb for VLDB

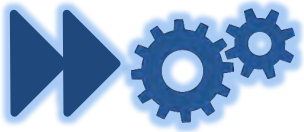
ASE database tuning (6 of 6)

- (#50) NVcache for tempdb (ASE 16.0 SP02 PL02)
  - Requires Memscale license
  - Requires dedicated hardware – fast disk
  - Tiered fast (and non-volatile!) storage to be treated as extension to data cache
  - Intended for scenarios where entire database cannot fit on the fast storage
    - If it fits, just create the database there normally
  - Useful for tempdb if tempdb cannot fit entirely on fast storage
  - Note: large I/O is disabled in NVcache

ASE object-level tuning

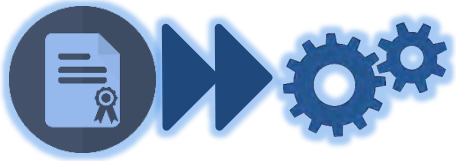
5 techniques

ASE object tuning (1 of 5)



- (#51) exp_row_size (ASE 11.9.2)
  - DOL tables only
  - Usually wouldn't create #temp tables with DOL since they are single-user
  - Consider explicitly creating as datapages (don't need datarows)
  - exp_row_size on #temp table is important for variable length rows
  - Remember any nullable columns = variable length!
  - If updating #temp tables, forwarded rows and other garbage can be painful
  - Set exp_row_size when creating #temp table
    - Yes, even with select into (plus most other table attributes)

ASE object tuning (2 of 5)



- (#52) Partitioned temp tables (ASE 15.0)
  - All temp tables can be partitioned
    - Even #temp tables
    - Even when created with select into
  - Why? Allows better/faster parallel query
  - Needs Partitions license

ASE object tuning (3 of 5)

- (#53) Precomputed result sets
  - First created in ASE 15.0, extended in 16.0 SP03 PL04
  - May prevent the need for a temp table at all
  - Similar to views
  - In ASE 16.0 SP03 PL04 they can be made temporary
    - Create on top of a view
    - Will only materialise when queried
    - Will auto-refresh if & when underlying data changes

ASE object tuning (4 of 5)

- (#54) Global shareable temp tables, version 1
  - Main overhead of any temp table is the metadata
    - Updating the various system tables during create and then drop
  - Advanced technique: don't constantly create and drop
  - Create a "permanent" tempdb%..temp table (note: no "#")
  - Users will share it
  - Leading column of PK should be spid
  - Add WHERE spid = @@spid to every statement using it
  - Can also use FGAC (fine-grained access control) for built-in WHERE clause
  - FGAC needs a license

ASE object tuning (5 of 5)

- (#55) Global shareable temp tables, version 2
  - ASE 16.0 SP03 introduced a way to do this directly
  - Users still share it, and only see their own rows
  - No license or additional management required
  - Can considerably reduce tempdb% I/O on system tables
  - Also reduces system table contention

ASE session-level tuning

6 techniques

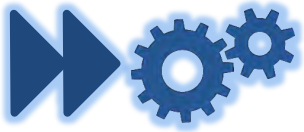
ASE session tuning (1 of 6)

- (#56) Delayed commit (ASE 15.0)
  - This can be set per database… but not for any temporary database
    - Even using the model trick
  - We can however set it in a session:
    - set delayed_commit on
  - This is worth setting in any code that populates and writes to any tempdb%
    - Including #temp tables

ASE session tuning (2 of 6)

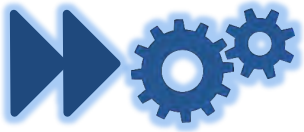
- (#57) dml_logging = minimal (ASE 15.5)
  - Can set per session
    - set dml_logging = minimal
  - Can use this for create & populate #temp tables even in dbid=2
  - This is worth setting in any code that populates and writes to any tempdb%

ASE session tuning (3 of 6)



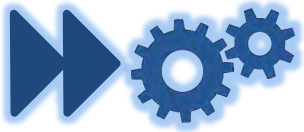
- (#58) Bulk inserts (ASE 15.7 SP130)
  - Can be set per session as well as per server
    - set ins_by_bulk on
  - Cannot be used for #temp tables, among many other restrictions
    - Can't be used for global shareable temp tables, because they already exist
    - But **can** be used for tempdb%..temp tables, if you can keep names unique
  - This is worth setting in any code that populates and writes to any tempdb%
    - Including #temp tables
  - Note: cannot be combined with minimal logging
    - Test which helps you more (probably minimal logging)

ASE session tuning (4 of 6)



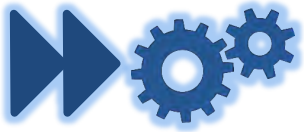
- (#59) conserve_tempdb_space (ASE 15.0.3 ESD#2)
  - set conserve_tempdb_space on
    - Is set when optlevel is ase_current or at least ase1503esd2
  - When on, limits tempdb usage according to *resource granularity*
    - Important to stop queries blowing up tempdb
  - Usually want this on
  - Might occasionally deliberately set it off in a session
    - If allowing more tempdb usage results in a faster query

ASE session tuning (5 of 6)



- (#60) auto_temptable_stats (ASE 15.0.3 ESD#3)
  - set auto_temptable_stats on
    - Is set when optlevel is ase_current or at least ase1503esd3
  - When on, automatically generates stats on #temp tables
    - Sounds good right?
    - But probably simplest form of update stats, with no options or attributes
  - If manually creating stats on #temp tables (and you should be)
    - No need to generate stats twice
    - Might consider switching this off **only** if manual stats are always generated

ASE session tuning (6 of 6)

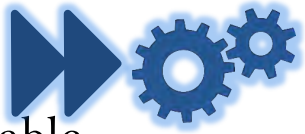


- (#61) unptn_pllscan (ASE 16.0)
  - set unptn_pllscan on
    - Is not set by any opt level or opt goal
    - Must be manually set
  - When on, allows parallel query on unpartitioned tables
  - Set before using large #temp tables
  - Ensure other parallel settings are correct

ASE statement-level tuning

12 techniques

ASE statement tuning (1 of 12)

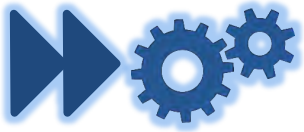


- (#62) Use temp tables more than once!
  - If you only use a #temp table once, you didn't need a #temp table
  - Could have used a view instead
  - Could have used a derived table instead (ASE 12.5.1)
  - Could have used a precomputed result set instead (ASE 15.7 SP100)

ASE statement tuning (2 of 12)

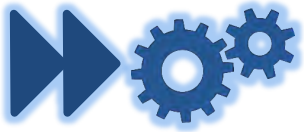
- (#63) Only write to a #temp table once
  - See this all the time: select into #temp, insert #temp, update #temp
  - You can use UNION and UNION ALL in select into
    - Can often prevent need for repeated insert statements
  - You can use CASE in updates
    - Can often prevent need for repeated update statements
  - Can use MERGE command (ASE 15.7)
    - Often prevents needs for separate insert/update statements

ASE statement tuning (3 of 12)



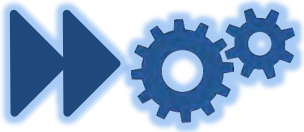
- (#64) Keep #temp tables as small as possible
  - Only write rows into the temp table that will be used
  - Use WHERE clauses to eliminate rows going into #temp table in the first place
  - Far better than populate #temp table and then delete rows
  - If you find yourself ever deleting from a #temp table you probably have bad code
  - Only write columns into the temp table that will be used
  - Be suspicious of any select *
  - Every column should be justified and actually used

ASE statement tuning (4 of 12)



- (#65) Set-based SQL always wins
  - Be suspicious of any WHILE loop
  - Be suspicious of any cursor
  - Almost always these can be rewritten as single pass set-based statements

ASE statement tuning (5 of 12)



- (#66) Truncate #temp table before explicit drop
  - Modern versions of ASE are supposed to do this anyway
    - As of ASE 15.0 ESD2
    - Which means if you're running compatibility mode you **don't** have this
    - Can use boot trace flag 3706 to force this in compatibility mode
    - Trace flag 3706 is obsolete as of ASE 15.0 ESD2 without compatibility mode
  - It is always good practice
  - Manually truncate #temp tables before end of last code that uses them
  - Manually drop #temp tables before end of last code that uses them

ASE statement tuning (6 of 12)

- (#67) Let the optimiser know about temp tables
  - A problem as old as Sybase… a proc doesn't know what a #temp table looks like
  - Old, bad answer: create proc … with recompile
  - Old, bad answer: exec proc … with recompile
  - Old, OK answer: create #temp in one proc, use it in another
  - Better answer (ASE 12.5): exec (<SQL text>)
  - Better answer (ASE 15.0.3): deferred compilation
  - Better answer (ASE 16.0): select … with recompile
  - Better answer (ASE 16.0 SP03 PL06):
    - Enable/disable deferred compilation per proc

ASE statement tuning (7 of 12)

- (#68) Let the optimiser know about temp tables
  - Building on the previous… create indexes on temp tables
    - Surprising how often this isn't done
  - ASE 16.0 now allows create index … with statistics
    - Can also pass all useful update stats attributes to create index
  - Particularly important if indexes created to let optimiser know
  - Even the humble sp_recompile works here

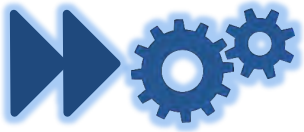
ASE statement tuning (8 of 12)

- (#69) Let the optimiser know about temp tables
  - Separate to previous: stats on all temp tables
  - auto_temptable_stats doesn't generate the best stats
  - Better to generate higher quality stats:
    - update index statistics … (ASE 11.9.2)
    - using 50 values (ASE 11.9.2)
    - with sampling_percent = [NN] (ASE 15.0)
    - with statistics histogram_tuning_factor (ASE 15.7)
    - with statistics hashing=[full | partial] (ASE 15.7 ESD#2)

ASE statement tuning (9 of 12)

- (#70) *text* as procedure parameter (ASE 15.7)
  - *text* parameters can be passed in and out of procedures
  - This may prevent need for some #temp tables at all
  - Test and time, it may not actually be faster
  - But even if slower it may cause less contention
  - With high concurrent user counts that probably matters more

ASE statement tuning (10 of 12)

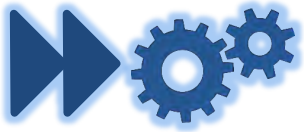


- (#71) Star join hint (ASE 16.0)
  - New abstract query plan hint for when there is a star join
    - Many dimension tables joined to a single central fact table
  - Hint significantly reduces rows passed to parallel hash joins
  - This uses far less space in tempdb% and makes the query faster
  - Specify AQP hints at end of statement
    - (use fact_table [...])
  - Extended in ASE 16.0 SP02 to allow UNION derived tables as dimension tables

ASE statement tuning (11 of 12)

- (#72) Tables as datatype (ASE 16.0 SP02)
  - Can now use tables as a datatype for user-defined functions and local variables
  - Might negate need for many #temp tables
  - Extended in ASE 16.0 SP03
    - Allows tables as parameters (in and out) to procedures
  - Extended in ASE 16.0 SP03 PL03
    - Allows inline table UDFs to become parameterised views

ASE statement tuning (12 of 12)



- (#73) Select … intersect/except (ASE 16.0 SP03)
  - UNION is like a logical OR of two result sets
  - INTERSECT is like a logical AND of two result sets, all rows that appear in both
  - EXCEPT is all the rows of one result set that don't appear in a second
  - Can negate need for many #temp tables now

Thank you

Contact information:

Joe Woodhouse
joe.woodhouse@primadonnaconsulting.com

PRIMA DONNA CONSULTING