SECRETS OF LOCKS AND BLOCKING IN SYBASE ASE

NA ASE VIRTUAL USER GROUP WEBCAST

EDWARD STANGLER
DIRECTOR OF R&D,
BRADMARK TECHNOLOGIES, INC

SYBASE ASE WEBCAST: SEPTEMBER 26, 2012

SECRETS OF LOCKS AND BLOCKING IN SYBASE ASE

NA ASE VIRTUAL USER GROUP WEBCAST

GUEST SPEAKER:

ROB VERSCHOOR, TECHNICAL DIRECTOR
DATA MGMT EVANGELISM FOR SYBASE

ABOUT BRADMARK

- Bradmark produces monitoring tools for databases, operating systems, and networks.
- Also has an online reorg tool for SAP R/3 on Oracle.
- Bradmark Surveillance is the top-tier end-to-end monitoring solution for SAP.
- Monitors SAP Sybase ASE, ASE CE, IQ, IQ MPX, RS, and more.

LOCKS AND BLOCKING IN ASE

- We will talk about several types of locks in Sybase ASE:
  - **Regular locks**
  - **Spinlocks**
  - **Internal locks**

- A **latch** is another kind of short-duration lock.

- A **semaphore** is generally a regular lock (except ULC semaphore).

LOCKS AND BLOCKING IN ASE

- We'll cover some MDA tables and just touch on sp_sysmon().

- These MDA tables are documented, but how locks and blockers look like is not well-documented.

- Lots of caveats when determining blockers in all scenarios.

COMING UP…

- Spinlocks
- Lock Owner ID (or, the notorious LockID)
- Distributed transaction locks (such as for XA transactions)
- Blockers
- Internal locks and miscellaneous

SPINLOCKS

WHAT ARE SPINLOCKS?

- A **spinlock** is lightweight (but high-CPU), low-latency lock for access to a shared resource (like an in-memory data structure) in a concurrent environment (i.e. between CPU engines).

- Spinlocks typically guard shorter operations such as memory access. Regular locks typically guard longer operations.

- Lower latency comes at the price of higher CPU usage if called too often when contention is high.

WHAT ARE SPINLOCKS?

-
  ```
  loop forever
          if quick non-atomic test
                  then if atomic test and acquire shared resource
                          break
          spin on the CPU (i.e. for i in 1..1000 do nothing)
  end loop
  access shared resource
  relinquish shared resource
  ```

- This is the “classical” spinlock algorithm.

WHAT ARE SPINLOCKS?

- Used everywhere:
  - Data caches
  - Procedure cache
  - User Log Cache (ULC)
  - Run queues
  - Lock hash tables
  - Internal structures (i.e. DBTABLE) and many more

- Three counters per spinlock: Hits, Waits, # of spins

WHAT ARE SPINLOCKS?

- Stefan Karlsson's oldie-but-goodie podcast on spinlocks:

  http://video.sybase.com/podcasts/sybase-podcast-ase-tech07-071007.mp3

- Spinlocks becoming more important as traditional wait events, such as disk I/O, are shortened or eliminated.

- IMDB, flash memory, SSDs, higher CPU core counts

WHEN GOOD SPINLOCKS GO BAD

- Sybase CR 632207 (fixed in ASE 15.5 ESD #3) is a rare example of spinlocks go wrong.
- Spinlocks used during query of MDA table monCachedObject.
- With many data cache sizes, works perfectly fine. But with very large data caches (i.e. 75 GB), the CPU went to 100% and brought the data server to its knees.
- Fixed by reducing the number of spinlocks held.

SPINLOCK CONFIGURE OPTIONS

- Some of the spinlock sp_configure() options:
  - lock hashtable size
  - lock spinlock ratio
  - lock table spinlock ratio
  - And many more

- Great discussions of these options in performance tuning papers put out by Sybase.

NO SPIN ZONE

- Prior to ASE 15.7 ESD #2, the only way to get spinlock information was through sp_sysmon().

- But spread across many different sysmon sections. And some spinlock information is still not available without custom hacks (see sp_sysmon_spinlock() in Mr. Tallman's Opus).

- Spinlock metrics are available in these sp_sysmon() sections: locks, dcache, mdcache, xactmgmt (ULC Semaphore), cache wizard

NEW KID ON THE BLOCK

- In ASE 15.7 ESD #2, Sybase introduced the new MDA table: monSpinlockActivity

- Shows the three counters (hits = grabs - waits, waits, # of spins) for each spinlock.

- Other great information available.

LIVE DEMO

LOCK OWNER ID

(OR, THE NOTORIOUS LOCKID)

WHAT IS A LOCK OWNER ID?

- A **Lock Owner ID** (loid) identifies the owner of a regular lock.
- Initially, a process is assigned an loid, and that loid is assigned to every regular lock that the process creates.
- It is *not*:
  - An ID for a lock
  - An ID for a transaction
  - An ID for a process

WHAT IS A LOCK OWNER ID?

- A lock owner ID can be shared by several spids in a family (in parallel queries). See “Viewing locks with sp_familylock” in the documentation for an example.

- A lock owner ID can be shared by several spids—though only one at a time—for distributed transactions, such as XA transactions.

WHEN IS A LOCKID NOT A LOCK ID?

- In non-MDA tables, the lock owner ID is identified with a column name “loid” or “xloid” or some variant (i.e. “block_xloid”).

- Lock owner ID column name is not consistent (loid vs. xloid).

- But in MDA tables, the column name is inexplicably called “LockID”. Further, the documentation is extremely misleading.

WHERE TO GET LOCK OWNER ID VALUES?

- Finding all of the outstanding lock owner IDs is not easy.
- Not in monProcess / sysprocesses.
- Some blockers won't have rows in systransactions (i.e. SELECT can grab locks without starting a transaction).
- monLocks / syslocks is a better source of lock owner IDs (when talking about blockers). But multiple locks will share the same loid.

SAMPLE (REGULAR) LOCKID VALUES

select distinct SPID, LockID from master..monLocks order by SPID

SPID	LockID
27	54
28	56
29	58
31	62

These are regular lock owner ID (not distributed transaction lock owner ID) values from an ASE 15.0.3 database, without families.

Not guaranteed patterns and could change in a future release.

IN SEARCH OF…

select SPID, LockID, LockType, LockLevel, DBID, ObjectID, PageNumber, RowNumber from master..monLocks order by SPID

SPID	LOCKID	LOCKTYPE	LEVEL	DBID	OBJECTID	PAGE	ROW
27	54	exclusive intent	TABLE	1	896719216	NULL	NULL
27	54	exclusive page	PAGE	1	896719216	2562	NULL
28	56	exclusive intent	TABLE	1	896719216	NULL	NULL
28	56	exclusive intent	TABLE	1	928719330	NULL	NULL
28	56	update page	PAGE	1	896719216	2562	NULL
28	56	exclusive page	PAGE	1	928719330	2593	NULL
29	58	exclusive intent	TABLE	1	896719216	NULL	NULL
29	58	update page	PAGE	1	896719216	2562	NULL
30	60	exclusive intent	TABLE	1	928719330	NULL	NULL
30	60	update page	PAGE	1	928719330	2593	NULL
31	62	exclusive intent	TABLE	1	928719330	NULL	NULL
31	62	update page	PAGE	1	928719330	2593	NULL

PK FOR MONLOCKS AND MORE

- LockID is obviously *not* a PK for monLocks.
- One way to make a PK for monLocks:
  (KPID, LockLevel, DBID, ObjectID, LockType, PageNumber, RowNumber)
- PK for monProcess: (KPID)
- *Limited* PK for systransactions: (loid)
  Unique to a SPID or distributed transaction at a point-in-time.

DISTRIBUTED TRANSACTION LOCKS

(SUCH AS FOR XA TRANSACTIONS)

WHY IS THERE A LOCK OWNER ID?

- The loid is assigned at process creation time.

- But a distributed transaction--such as an XA transaction--will assign a new lock owner ID, so that the transaction and locks can be shared between processes.

- Support for distributed transaction lock owner IDs was added in ASE 12.0. That’s when the “loid” column was added to tables such as syslocks.

DISTRIBUTED TRANSACTION

- A distributed transaction survives beyond the lifetime of any particular process. An XA transaction is an example of a distributed transaction.

- The Global Transaction ID (xa_gtrid) and the Branch Qualifier (xa_bqual) are important in order to identify the XA transactions.

- So, typically want to get the xa_gtrid(xactname) and xa_bqual(xactname) from systransctions for XA transactions which are blockers.

DETACHED PROCESS

- Processes attach and detach from a distributed transaction.
- When no process is attached, a lock is sometimes referred to as a *detached process lock*.
- In systransactions, the row for the detached process now has connection = 2 and spid = 0.

DETACHED PROCESS

- The locks and transaction for a distributed transaction persist after the process has detached (SPID = 0 and KPID = 0).

- Need to look at more than just SPID and KPID in order to handle multiple distributed transaction blockers that have detached processes (they will all have SPID = 0 and KPID = 0).

- The loid differentiates between these cases. The locker owner ID was uniquely created for the distributed transaction.

BLOCKERS

BLOCKER SCENARIOS

- Three blocker scenarios:
  1. Blocker is a normal process.
  2. Blocker is a process that has started a distributed transaction.
  3. Blocker is a detached process (that had started a distributed transaction). In other words, the distributed transaction itself is the blocker.

- Most scripts only handle scenario #1, which is for (attached) processes blocking other processes on regular locks.

SCENARIO #1

Blocker is a normal process.

- Only BlockingSPID is set in monProcess for the waiting processes.
- Will find the blocker process in monProcess.
- Blocker may or may not have a row in systransactions.

SCENARIO #2

Blocker is a process that has started a distributed transaction.

- Both BlockingSPID and BlockingXLOID may be set in monProcess for the waiting processes.
- The blocker process suddenly has a BlockingXLOID because it receives a new lock owner ID as part of starting the distributed transaction.
- Will find the blocker process in monProcess.
- Blocker will have a row in systransactions.

SCENARIO #3

Blocker is a detached process.

- Then only BlockingXLOID is set in monProcess for waiting processes.
- Will *not* find the blocker process in monProcess (since it's detached, after all).
- Blocker will have a row in systransactions.
- In systransactions, the row for the detached process now has connection = 2 (was: 1) and spid = 0.

MONLOCKTIMEOUT

- New in ASE 15.7.

- Shows blocking information *after* the locks have timed out, one row per waiting process.

- Not helpful for blocking scenarios that are *currently* happening, but still a great post-event diagnostic tool.

MONLOCKTIMEOUT

- sp_configure ‘lock timeout pipe active’
  sp_configure ‘lock timeout pipe max messages’

- Little-known fact:
  Pipe sizes are *per engine*!

- If want to set global lock timeout (may require ASE restart when first turning on):
  sp_configure ‘lock wait period’

MONLOCKTIMEOUT

```
1> set lock wait 10
2> go
1> update blocked_tab set a=a
2> go
Msg 12205, Level 17, State 2:
Server ‘MYSERVER157', Line 1:
Could not acquire a lock within the specified wait period.
SESSION level wait period=10 seconds, spid=25, lock
type=update page, dbid=1, objid=217048778, pageno=2913,
rowno=0. Aborting the transaction.
```

CURRENTLY BLOCKING

- To see blocking scenarios that are *currently* running, then need to use a custom script.

- Find all blockers by looking for (BlockingSPID, BlockingXLOID) in monProcess. This will include blockers from all three scenarios.

- Difficult to get BlockingInstanceID or BlockingKPID, however.

- Take into account the three different blocker scenarios (different values for BlockingSPID and BlockingXLOID).

CURRENTLY BLOCKING

- Join to build a block tree.
- Correlate with monLocks to grab all of the blocker process and waiting process information that's needed for real-time alerting.
- Grab any post-build information that's needed, such as from systransactions.

LIVE DEMO

INTERNAL LOCKS AND MISCELLANEOUS

ERROR 8233

- Trying to use system functions like object_name() on a table that is undergoing REORG or ALTER TABLE will throw Error 8233 in your query.

- Trace flag 2792 (in ASE 15.0.3 ESD #3 and ASE 15.5 ESD #1) can convert the error to a sleep, but may result in deadlock (see Sybase CR 620045).

- Hard to detect, but may have some success by looking at monLocks.SourceCodeID or looking for certain commands, such as “$ALTER TABLE”, in systransaction.xactname.

ERROR 936

- If you try to access the “model” database while any database recovery is going on, then you get Error 936 (“The Model database is unavailable. It is being used to create a new database.”).

- Can use DBCC DBTABLE(model) and check if dbt_lock = 1. See Sybase Case 11525932.

CLUSTER EDITION

- Prior to ASE CE 15.5, systransactions only showed the current instance.

- On ASE CE, for an attached process which has started a distributed transaction, only BlockingXLOID is set unless the blocker process and waiting process are on the same node (then BlockingSPID is also set). Seems like a bug.

CLUSTER EDITION

- Wait event 454 - "waiting for cluster logical lock lookup“

- Caused by insufficient “lock hashtable size”.

- Search the documentation for “Increase in lock hashtable size” (with the double quotes) to see the wacky formula for calculating the correct “lock hashtable size” (including the mysterious “divide-by-8”).

- Check with sp_sysmon() that the Avg. Chain Length for Page & Row Lock HashTable is not too high.

DETACHED PROCESS AND MONLOCKS

- monLocks.BlockState may show "Detached“ for a lock held by a process that has just started a distributed transaction, but since connection = 1 in systransactions, the process is very much still attached.

- monLocks.BlockedBy has issues (at least in ASE 15.5):
  - It refers to the lock owner ID (LockID) and not the SPID (which is apparently by design; see Sybase CR 446521). Documentation says SPID.
  - Sometimes shows the wrong value.

TALLMAN'S OPUS

- A great resource for further tuning your system is the (huge) whitepaper, "Managing Workloads with ASE: Techniques for OLTP Scaling and Performance Management with Large SMP and Shared Disk Clusters".
- Anything you could possibly imagine tuning in ASE is in that whitepaper.
- Nearly completely correct.

BRADMARK AT SAP TECHED 2012

- Bradmark is a Silver sponsor and exhibitor – Booth 111.

- Please visit us if you have questions or want to see more.

- Bradmark will be introducing a new product for an exciting SAP database.

- And will be showing the latest in monitoring for SAP Sybase ASE, IQ, RS, and more.

CONTACT

- Further questions:

  estangler@bradmark.com

  http://www.bradmark.com

bradmark technologies

Q&A

SYBASE®

An SAP® Company

Questions And Answers