Microsoft

CSS Academy

How to Resolve Blocking and Deadlock in SQL Server

Ying Fan

Support Engineer
Customer Service & Support
Asia Pacific & Greater China Region

Welcome!

Microsoft

CSS Academy

Microsoft

CSS Academy

Last Workshop Re-cap

- Index structure
- Reading Query Plans
- Troubleshooting Query Plans
- Optimizing Query Plans
- Plan Caching issue

Agenda

- Blocking Issue
  - Locking
  - PageIOlatch
  - Pagelatch
  - ASYNC NETWORK IO
  - Tempdb Contention
- Deadlock

BLOCKING

What is Blocking?

- An existing spid has a lock on the resource
- Your connection is requesting a lock on the same resource with an incompatible lock mode

Informatiom collection

- Sys.sysprocesses
- Sys.dm_exec_requests
- Sys.dm_os_wait_stats
- Blocking script (KB271509)
- SQL trace
- Pssdiag

Information From Sysprocesses

- Blocked – what spid is blocking yours?
- Cmd – basic type of command being processes
- Status – is the spid doing anything
- Waittype – hex value denoting what the spid is waiting on
- Lastwaittype – saved each time a wait finishes
- Waittime – how long has it been waiting (milliseconds)
- Waitresource – text description of resource
- Open_tran – number of nested transaction

DMVs

- Sys.dm_exec_requests
  - New DMV introduced from SQL Server 2005
  - Sys.sysprocesses still available for backward compatibility purpose
  - Handle of the TSQL and query plan
- Sys.dm_exec_connections
- Sys.dm_exec_sessions

Waittype in Sys.dm_os_wait_stats

- LCX_M_x
- PAGEIOLATCH_x
- PAGELATCH_x
- ASYNC_NETWORK_IO
- WRITELOG
- CXPACKET
- SQLTRACE_x

LCK_M_x

- Used to Provide Desired Level of Logical Consistency of Data
- Type of locks acquired depend on:
  - Isolation level
  - Lock cost
  - lock escalation
  - Lock hints
- Locking issue could be caused:
  - Long running query/sp
  - Improper use of transaction

Lock Resources and Hierarchy

- Common Resources
  - **RID**
  - **Key**
  - **Page**
  - **Table**
- Lock Hierarchy
  - Acquire intent locks before traversing to the level below

Resource	Format	Example
Table	DatabaseID:ObjectID	TAB: 5:1
Page	DatabaseID:File:Page	PAG: 5:1:104
Key	DatabaseID:ObjectID:IndexID (Hash value for index key)	KEY: 5:1977058079:1 (02014f0bec4e)
RID	DatabaseID:FileID:PageID:Slot	RID: 5:1:104:3

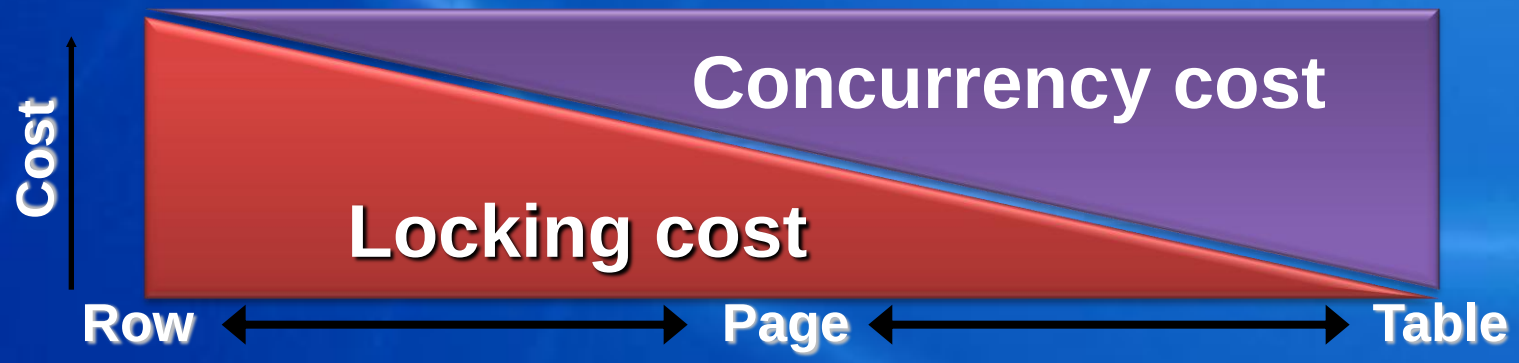
Lock Duration

- Lock Mode and Transaction Isolation Level

Lock Mode	Read Committed	Repeatable Read	Serializable
Shared	**Held until data read and processed**	**Held until end of transaction**	**Held until end of transaction**
Update	**Held until data read and processed unless promoted to Exclusive**	**Held until data read and processed unless promoted to Exclusive**	**Held until end of transaction unless promoted to Exclusive**
Exclusive	**Held until end of transaction**	**Held until end of transaction**	**Held until end of transaction**

Dynamic Locking

- Row locking is not always the right choice
  - Scanning a table with 100 million rows means 100 million calls to the lock manager
- Sometimes page or table locking is the optimal way to scan
  - Table locks don't allow much concurrency but are cheaper to acquire and manage
- SQL Server locking strategy is dynamic
  - SQL Server chooses the lowest cost locking strategy (row, page, or table) at run time based upon input from the query optimizer



Lock Escalation

- Happens in two conditions:
  1) >5,000 locks
  2) lock memory > 40 %
- Used to lower the number of locks taken by a transaction
  - Lock manager attempts to replace one transaction's many row or page locks with a single table-level lock
  - Escalation never converts row locks to page locks
  - There is no lock escalation on temporary tables or system tables.
- Lock “de-escalation” never occurs

Lock Time Out

- Application lock timeout
  - User configurable lock time-out
    - SET LOCK_TIMEOUT 10000
  - Rollback transaction
- Internal lock timeout
  - Internal lock request can time out
  - Does not rollback transaction

Locking Hints

- Can be specified using the SELECT, INSERT, UPDATE, and DELETE statements
- Direct SQL Server to the type of locks to be used
  - **Granularity hints:** ROWLOCK, PAGLOCK,TABLOCK
  - **Isolation Level hints:** HOLDLOCK, NOLOCK
    - READCOMMITTED, REPEATABLEREAD, SERIALIZABLE, READUNCOMMITTED
  - **UPDLOCK:** use update lock rather than shared lock when reading
  - **XLOCK:** use exclusive lock instead
  - **READPAST:** will “skip” rows that are currently locked
- Used when a finer control of the types of locks acquired on an object is required
- Override the current transaction isolation level for the session

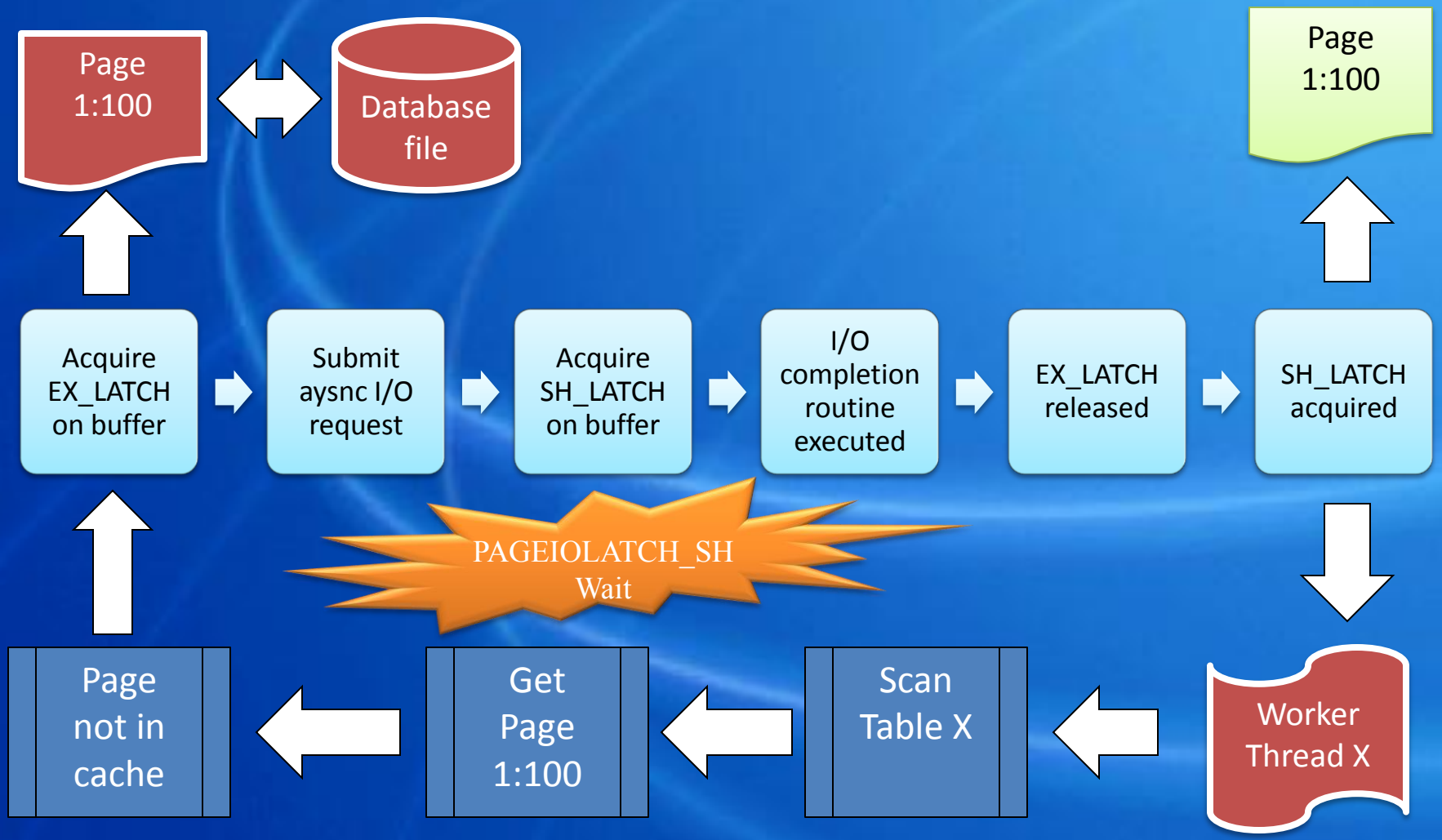
Recommendations to prevent locking issue

- Keep the transaction short and in one batch
- Separate readers from writers
- Beware of implicit transactions
- Use lowest Isolation Level required
- Process all Result Quickly
- Avoid Escalation when it is NOT necessary

PAGEIOLATCH_x

- Latches are short term synchronization objects. used to synchronize access to buffer pages. PageIOLatch is used for disk to memory transfers.
- If this is significant in percentage, it typically suggests memory pressure and disk IO subsystem issues. Check memory and disk counters.

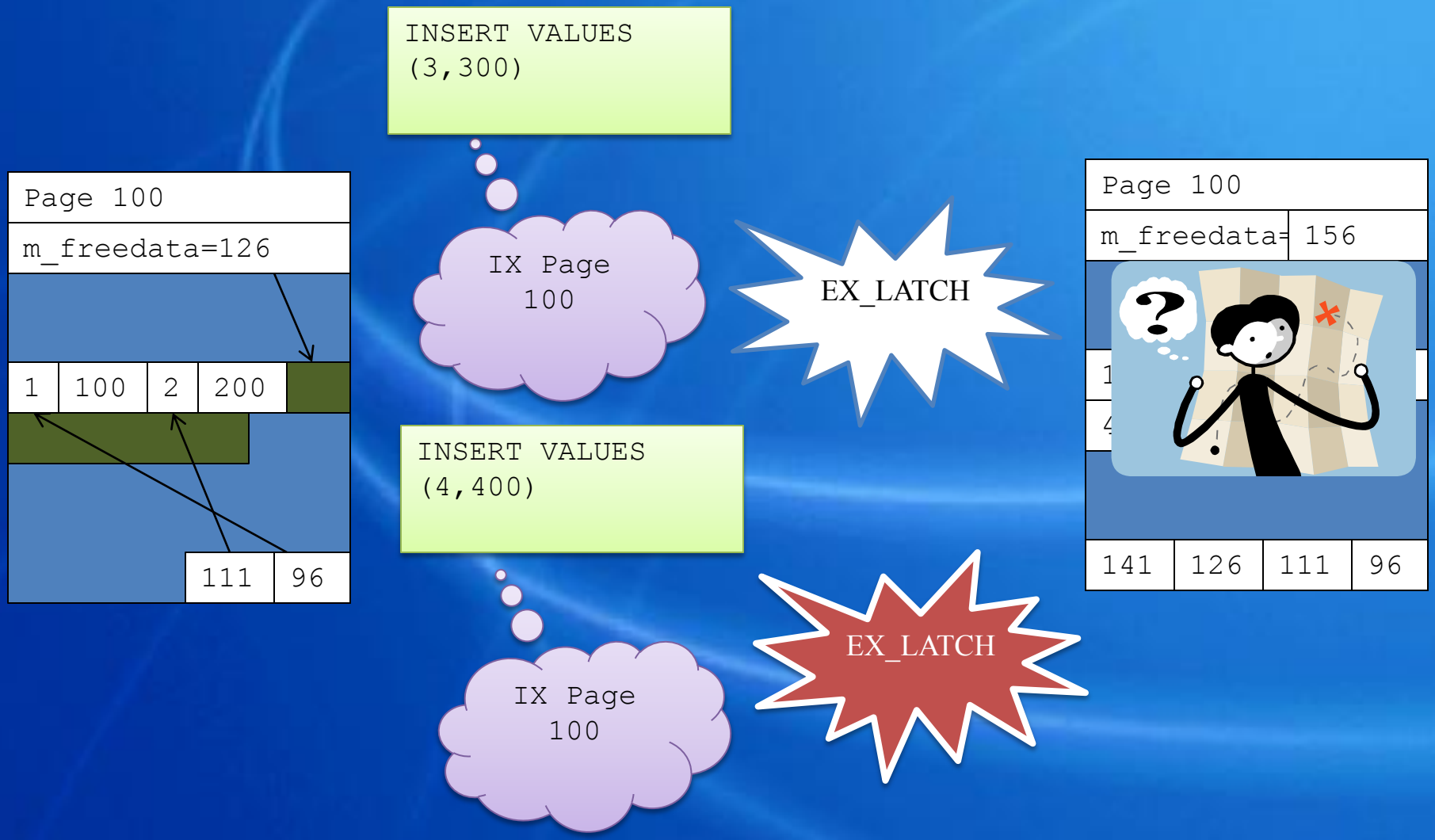
How PAGEIOLATCH_x works for Reading a page from disk



PAGELATCH_x

- Occurs when a task is waiting for a latch for a buffer that is not in an I/O request.
- If this is significant in percentage, it typically indicates cache contention, not IO or memory.
- Gather blocking script. Try to find out the work load which has caused contention.
- Could consider gathering a dump file to double confirm your suspect.
- Review the known SQL issues.
- Also could consider using partitioned table

Who Needs a PAGELATCH_x When I Have Locks



Tempdb Contention

Where will I see latch contention?

- SGAM , PFS, and GAM pages
- System catalog pages

Why is there latch contention?

- Alloc pages must be latched as part of “transaction” of allocation
- System tables pages latched due to drop/create of a table

How do I reduce the contention?

- Create multiple data files
- All files have the same size
- Avoid autogrow and reduce tempdb usage

This could happen in a user database

- If users dropped/created tables frequently.

Demo

Problem:

The end user complains application “slow”.

Questions:

What do you think the problem is?

Why contention on tempdb occurs?

Action plan?

ASYNC_NETWORK_IO

- SQL Server has finished query execution within database engine.
- SQL Network I/O is asynchronized
- Normally caused by
  - network issue
  - Client does not fetch the data quickly
- Netmon trace is useful to identify the problem

Lab 1 - Blocking

- Learn how to analyze the blocking issue with blocking script and SQL trace.

Advanced blocking issues

- TokenAndPermUserStore
- Resource semaphore
- Resource semaphore query compile

TokenAndPermUserStore

Symptoms:

- Queries that typically run faster take a longer time to finish running
- CPU utilization for the SQL Server process is more than usual.
- When you experience decreased performance when you run an ad hoc query, you view the query from the **sys.dm_exec_requests** or **sys.dm_os_waiting_tasks** DMVs. However, the query does not appear to be waiting for any resource.
- The size of the TokenAndPermUserStore cache store is in the order of several hundred megabytes (MB)

```
SELECT SUM(single_pages_kb + multi_pages_kb) AS
"CurrentSizeOfTokenCache(kb)" FROM sys.dm_os_memory_clerks
WHERE name = 'TokenAndPermUserStore'
```

TokenAndPermUserStore

Solutions

- Explicitly parameterize ad hoc queries.
- Wrap ad hoc queries within stored procedures, and use stored procedures instead of directly executing ad hoc queries.
- Add the login that executes varied ad hoc queries as a member of the sysadmin server group.
- Flush entries from the TokenAndPermUserStore cache. - DBCC FREESYSTEMCACHE ('TokenAndPermUserStore')

TokenAndPermUserStore

Solutions (cont.)

- SQL 2005 SP2: Trace Flag 4618 and 4610
- SQL 2005 SP3: Trace flag 4621 + add registry key

  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\<MSSQL.X>\MSSQLServer

  Name: TokenPermQuota

  Type: DWORD

  Value: *<QuotaSize in Hex>*

- SQL 2008: sp_configure - "access check bucket count" and "access check cache quota" option

Resource semaphore

- Memory grants were required for queries that performed sort or hash operations. These queries can be potentially blocked on RESOURCE_SEMAPHORE wait type—until the necessary grant could be acquired.

- If time-out, error 8645 is encountered.

- Acquiring the grant before execution ensures that the necessary memory will be available at runtime.

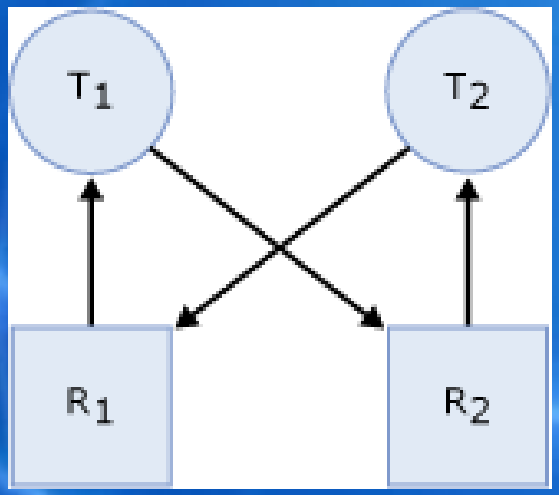
Resource semaphore query compile

- A large number of waits of type RESOURCE_SEMAPHORE_QUERY_COMPILE indicates a large number of concurrent compiles.
- SQL server has gateways for throttling individual query optimizations based on memory usage
- To prevent inefficient use of server resource.

Deadlock

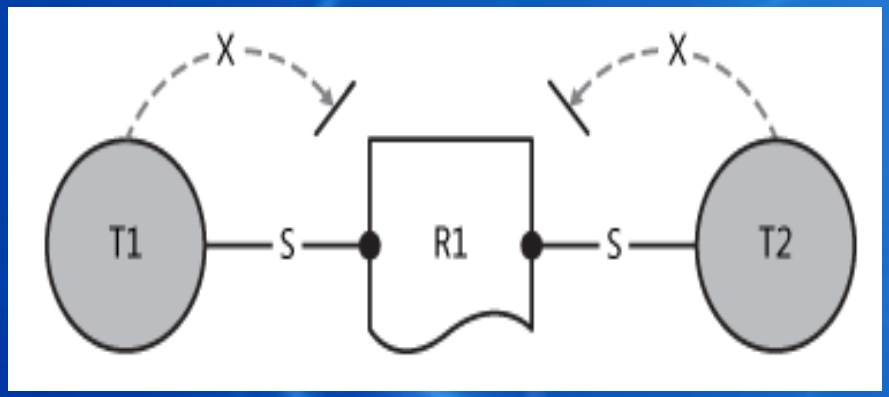
Deadlock

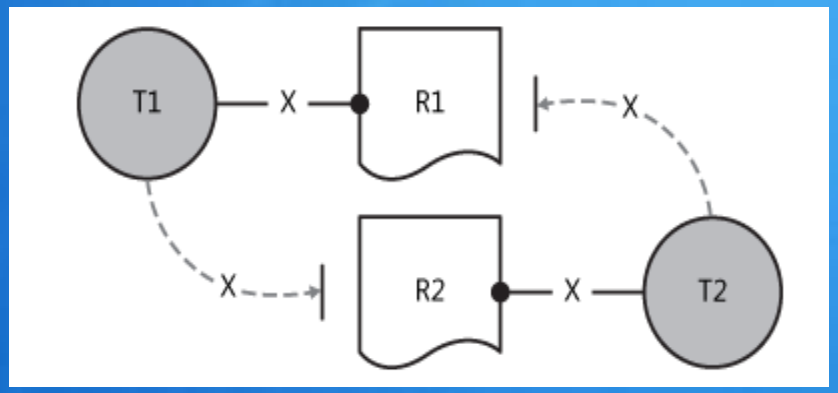
- Occurs when two connections are simultaneously waiting on a resource that the other connection holds

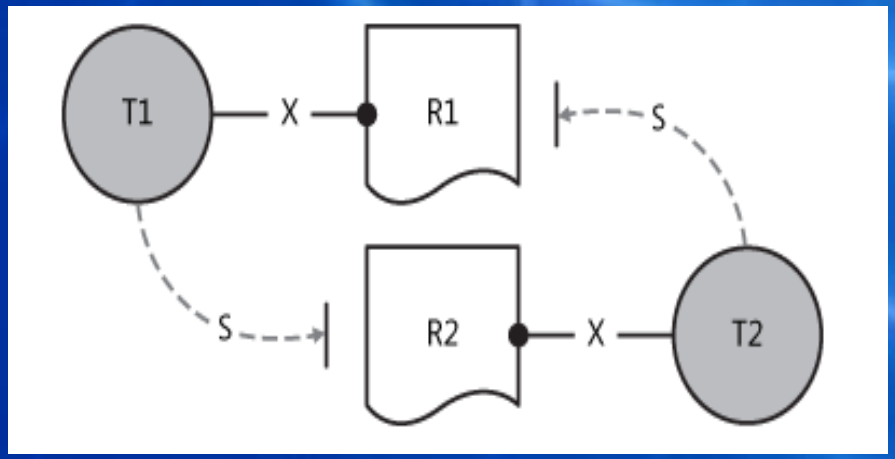


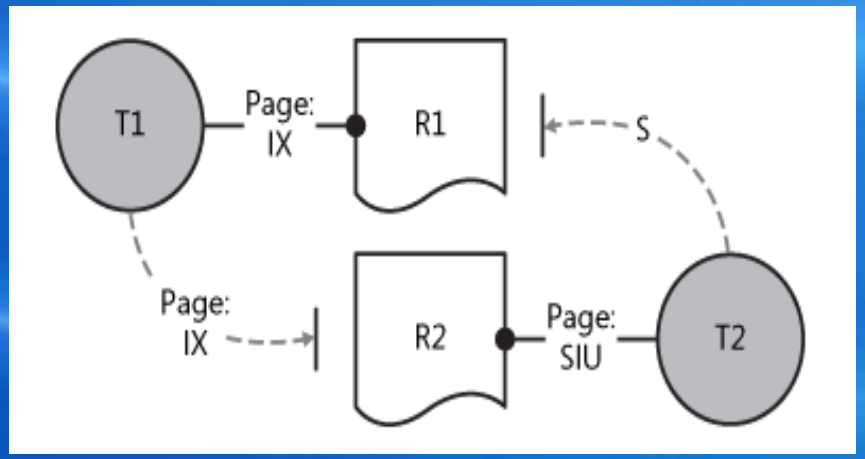
  - Tran #1 holds Resource1 and is waiting on Resource2
  - Tran #2 holds Resource 2 and is waiting on Resource1
  - Each transaction is waiting on the other, and neither can proceed

Deadlock Examples









Deadlock Detection and resolution

- Lock manager detects deadlocks automatically by means of a background process called LOCK_MONITOR
- If deadlock is detected. Select the transaction that is cheapest to rollback - the deadlock victim
- Rollback the victim's transaction
- Notify the victim by means of a 1205 error:

Error 1205: Your transaction (process ID #%d) was deadlocked with another process and has been chosen as the deadlock victim. Rerun your transaction.

Information collection for deadlock

- Enable trace flag 1204 or 1222 (new from SQL Server 2005)
- SQL errorlog
- Capture SQL trace

How to avoid the deadlock

- Add the clustered index
- Reduce the lock holding time in the transaction
- Use the small transaction instead of the long duration transaction
- Commit the implicit transaction
- Reduce the lock range
- Change the lock type

Deadlock Analysis – Example Using Trace Flag 1204

Deadlock encountered .... Printing deadlock information

Wait-for graph

Node:1

KEY: 5:2121058592:1 (a70064fb1eac) CleanCnt:1 Mode: X Flags: 0x0

Grant List::

Owner:0x19165c00 Mode: X Flg:0x0 Ref:0 Life:02000000 SPID:58 ECID:0

SPID: 58 ECID: 0 Statement Type: UPDATE Line #: 1

Input Buf: Language Event: update authors set au_id = au_id

Requested By:

ResType:LockOwner Stype:'OR' Mode: U SPID:57 ECID:0 Ec:(0x1afcd520) Value:0x19167bc0 Cost:(0/10AC)

Node:2

KEY: 5:1977058079:1 (02014f0bec4e) CleanCnt:1 Mode: X Flags: 0x0

Grant List::

Owner:0x1916c220 Mode: X Flg:0x0 Ref:0 Life:02000000 SPID:57 ECID:0

SPID: 57 ECID: 0 Statement Type: UPDATE Line #: 1

Input Buf: Language Event: update titles set type = type

Requested By:

ResType:LockOwner Stype:'OR' Mode: U SPID:58 ECID:0 Ec:(0x1936d520) Value:0x1916bb80 Cost:(0/54)

Victim Resource Owner:

ResType:LockOwner Stype:'OR' Mode: U **SPID:58** ECID:0 Ec:(0x1936d520) Value:0x1916bb80 Cost:(0/54)

Lock Resource

Lock Mode

Useful if Stored Proc is Involved

Lock Mode

Deadlock Victim

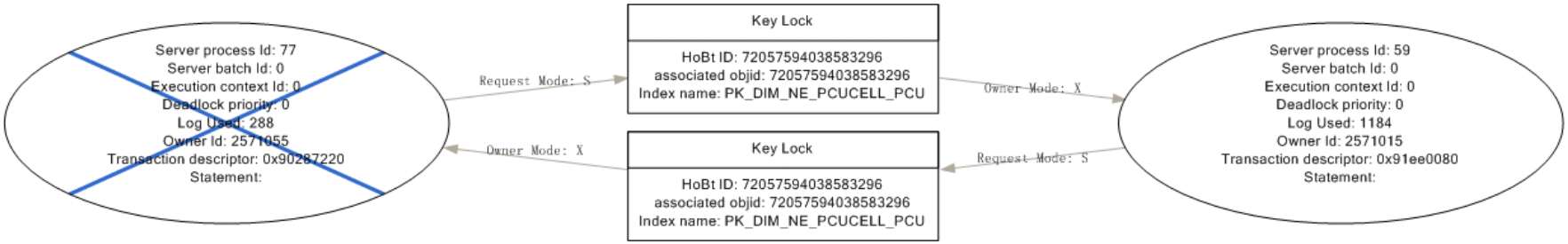
Analyze deadlock in SQL 2005

- New trace flag is introduced: T1222
- New SQL trace event: Deadlock graph

SQL Server Profiler - [F:\...\deadlock\HP-DL580_WEDNESDAY_sp_trace.trc]

File Edit View Replay Tools Window Help

EventClass	BinaryData	DatabaseID	TransactionID	NTUserNa...	NTDomainN...	HostNa...	ClientProcessID	ApplicationName	LoginName	SPID	Duration	StartTime	EndTime
SP:StmtCompleted		10	2580661			HP-DL580	24232	DataImport	sa	65	3	2008-01-09 11:18:22.310	2008-01-
SP:StmtCompleted		10	2580661			HP-DL580	24232	DataImport	sa	65	20	2008-01-09 11:18:22.293	2008-01-
SQLTransaction		10	2580661			HP-DL580	24232	DataImport	sa	65	20	2008-01-09 11:18:22.293	2008-01-
User Error Message		10				HP-DL580	24232	DataImport	sa	65		2008-01-09 11:18:22.327	
SQLTransaction		12	2580659			HP-DL580	24232	DataImport	sa	65	21	2008-01-09 11:18:22.293	2008-01-
SP:StmtStarting		12	2580652			HP-DL580	24232	DataImport	sa	65		2008-01-09 11:18:22.390	
Lock:Deadlock Chain	0X0A000...	8	2571175						sa	4		2008-01-09 11:18:23.187	
Lock:Deadlock Chain	0X0A000...	8	2571015						sa	4		2008-01-09 11:18:23.203	
Lock:Deadlock Chain	0X0A000...	8	2571055						sa	4		2008-01-09 11:18:23.203	
SQLTransaction		1	2582872						sa	16	0	2008-01-09 11:18:23.310	2008-01-
SQLTransaction		1	2582872						sa	16	0	2008-01-09 11:18:23.310	2008-01-
SQLTransaction		1	2582876						sa	16	0	2008-01-09 11:18:23.327	2008-01-
SQLTransaction		1	2582876						sa	16	0	2008-01-09 11:18:23.327	2008-01-
Deadlock graph		1							sa	16		2008-01-09 11:18:23.327	



Lab 2 - Deadlock

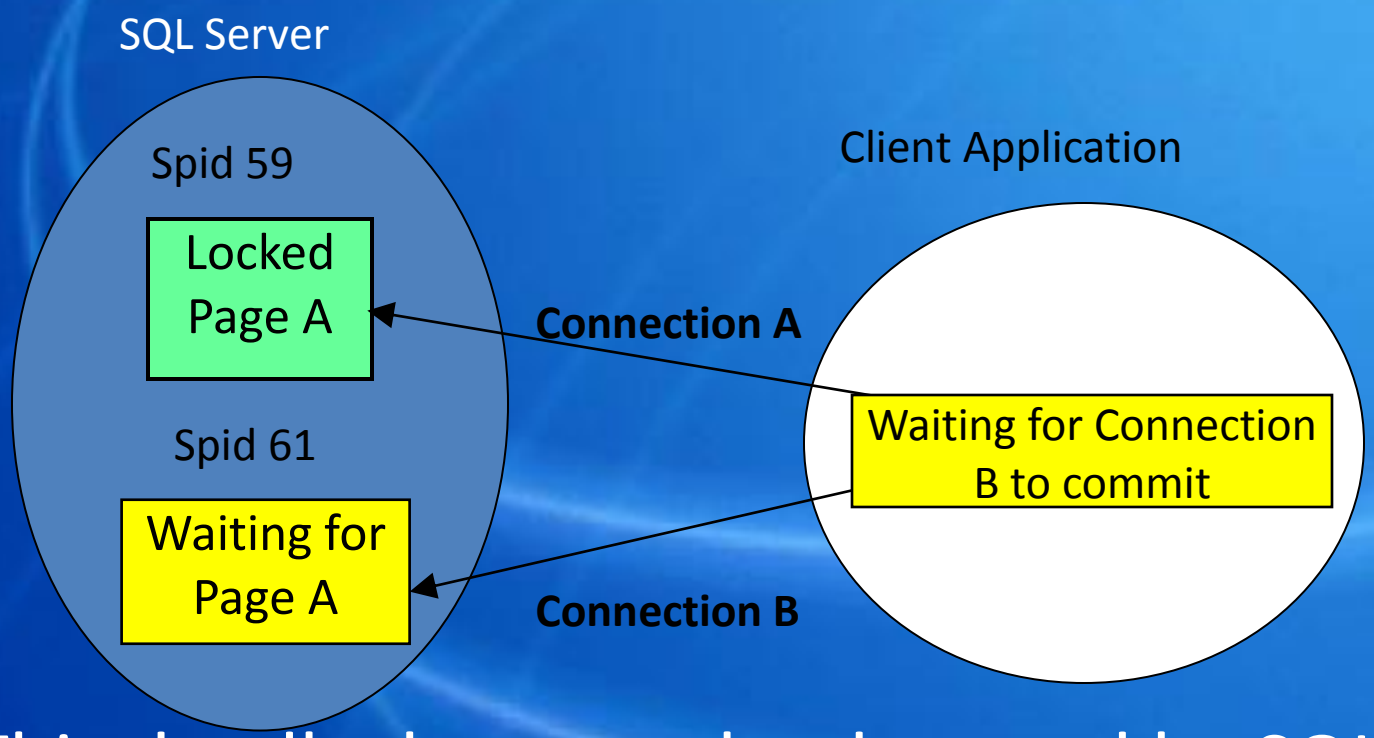
- Generate a deadlock issue manually
- Analyze the deadlock
- How to resolve it?

Distributed Deadlock

- Distributed deadlocks usually occur when one of the locked resources resides outside of SQL Server (or on another SQL Server)

Distributed Deadlock – Scenario 1

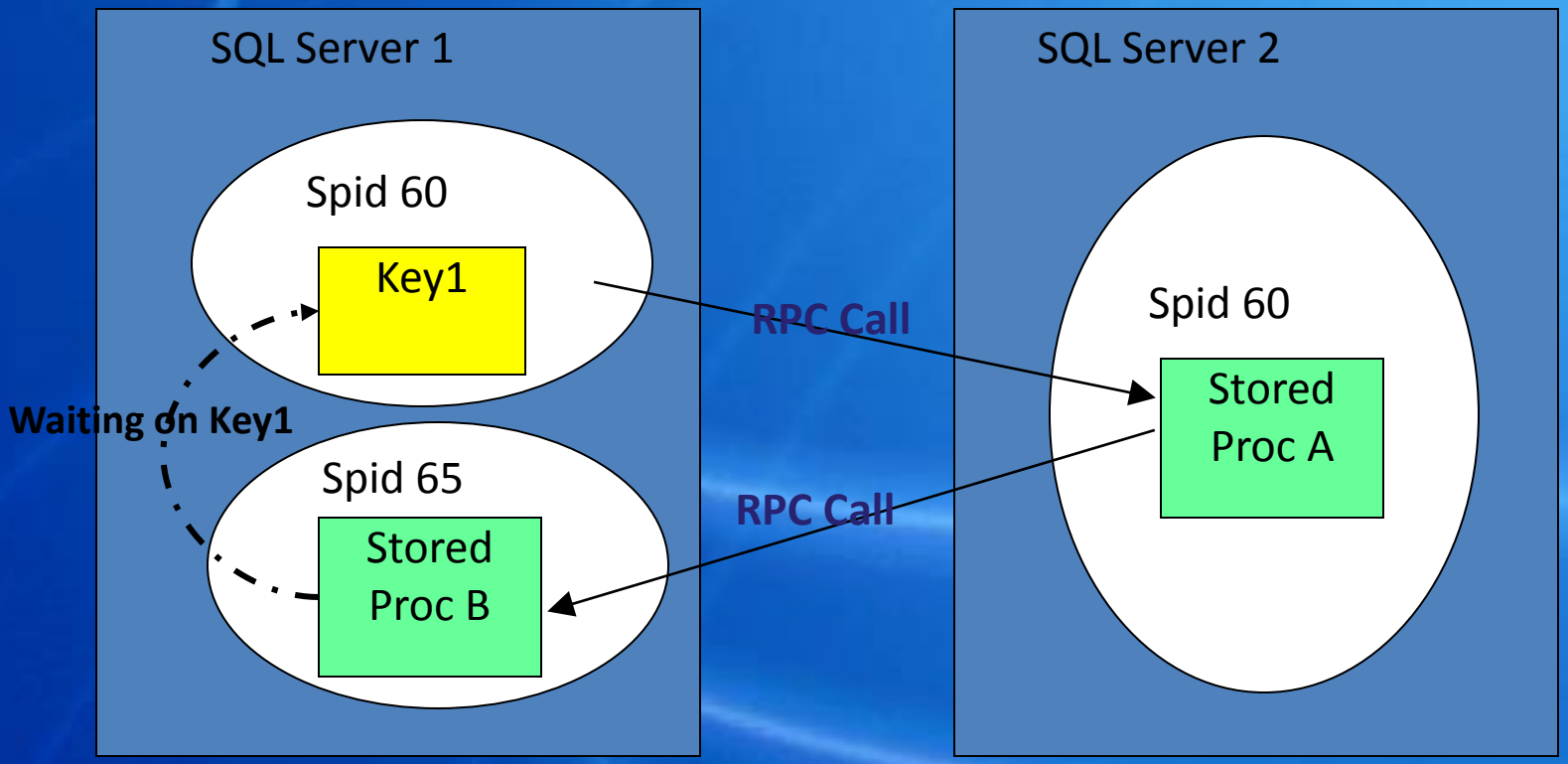
- Scenario 1



- This deadlock cannot be detected by SQL Server

Distributed Deadlock – Scenario 2

- Scenario 2



- This deadlock cannot be detected by SQL Server

Symptoms

- SQL server one or more SPIDs were waiting on ASYNC_NETWORK_IO
- Application server hang or ran slowly

Information to collect

- Netmon traces
- PSSDIAG/Blocking script
- Dumps for <application>.exe and sqlservr.exe

Review

- Blocking Issue
  - Locking
  - PageIOlatch
  - Pagelatch
  - ASYNC NETWORK IO
  - Tempdb Contention
- Deadlock

Q & A

Thank You!

Microsoft®

Your potential. Our passion.™