SQL Server Troubleshooting

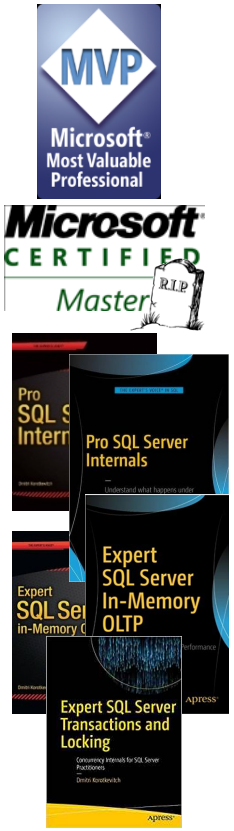
Intro Into Wait Statistics

Dmitri Korotkevitch
http://aboutsqlserver.com

Good morning!

20+ years of experience in IT
15+ years of experience working with SQL Server
Microsoft Data Platform MVP
Microsoft Certified Master
Author

- Pro SQL Server Internals (v1-2)
- Expert SQL Server In-Memory OLTP (v1-2)
- Expert SQL Server Transactions and Locking



Slides and Demos: **https://aboutsqlserver.com/presentations**
Youtube Channel: aboutsqlserver
Blog: http://aboutsqlserver.com
Email: dk@aboutsqlserver.com

Agenda

SQLOS and Wait Statistics overview

Troubleshooting approach for the typical waits and bottlenecks

Simplest Optimization Strategy

SQLOS

Layer between SQL Server and Windows

- In SQL2017+ includes PAL (Platform Abstract Layer) to support cross-platform architecture

Responsible for

- Scheduling
- I/O operations
- Memory and Resource Management

SQL Server Execution Model

SQLOS creates 1 scheduler per logical CPU

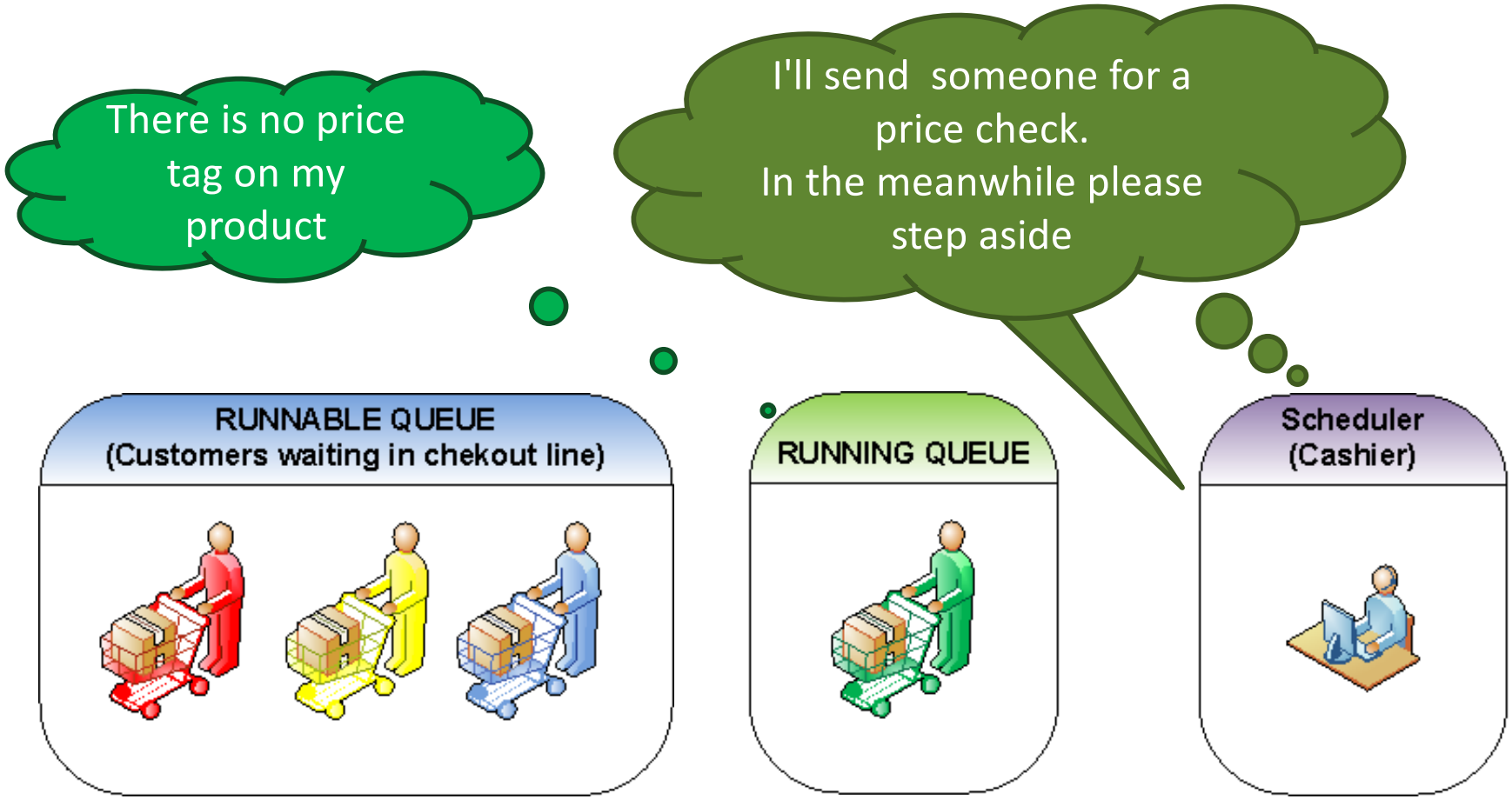
Worker Threads are created and evenly divided across schedulers

Batch is assigned to 1 or multiple workers

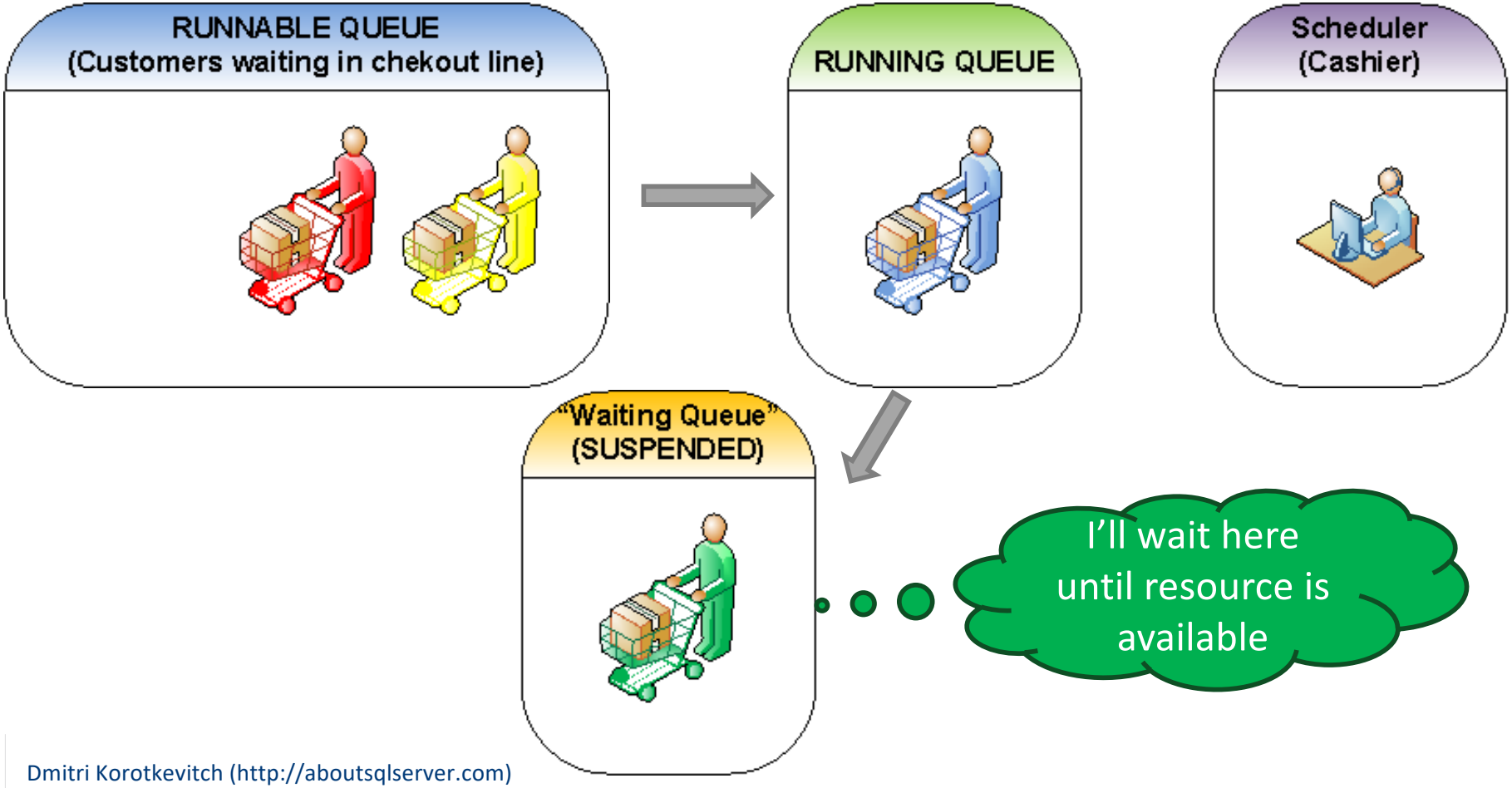
Worker states:

- Running – currently executing on CPU
- Suspended – waiting for the resource
- Runnable – waiting for CPU to execute

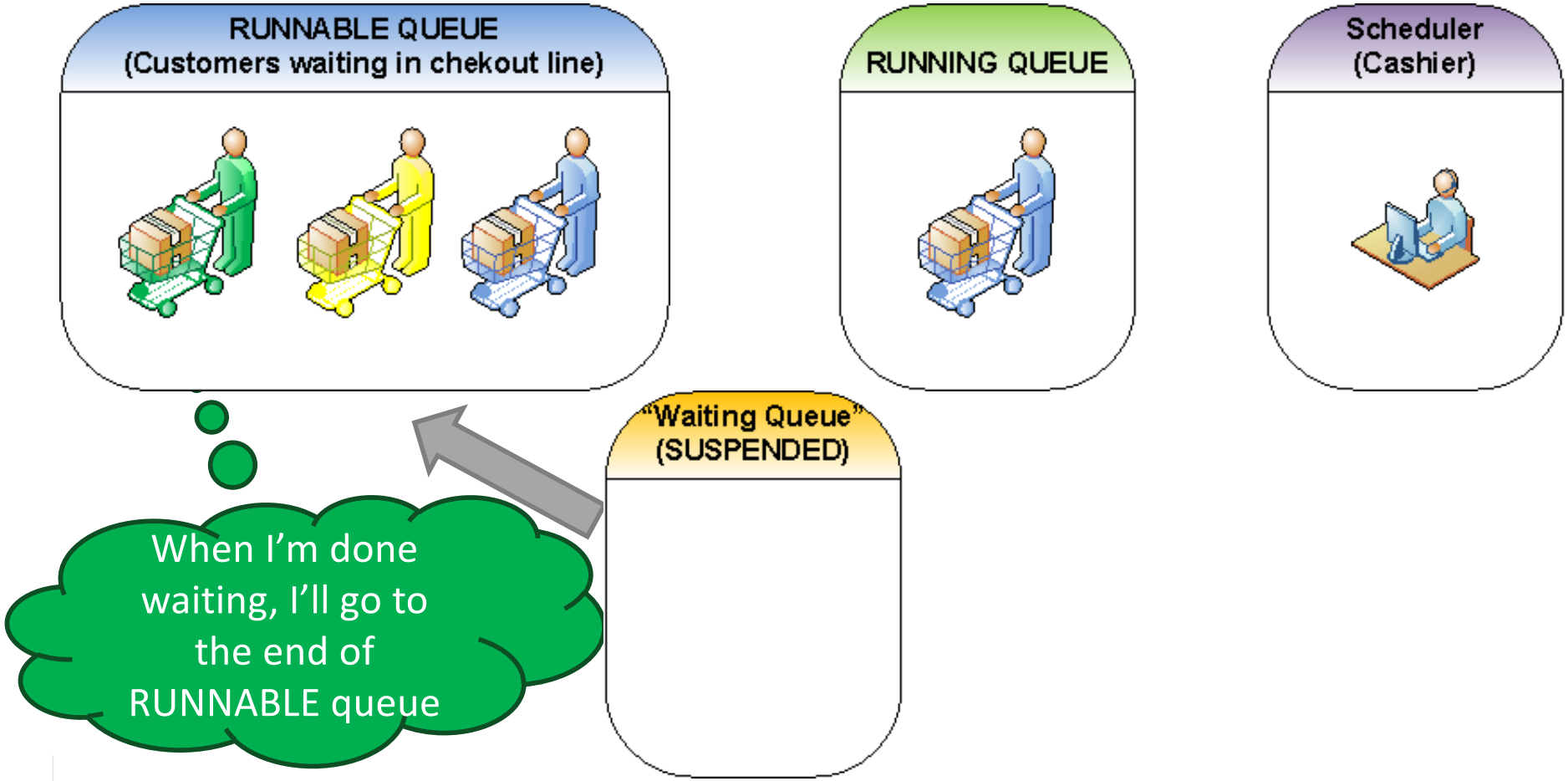
Execution Model – 1 Scheduler



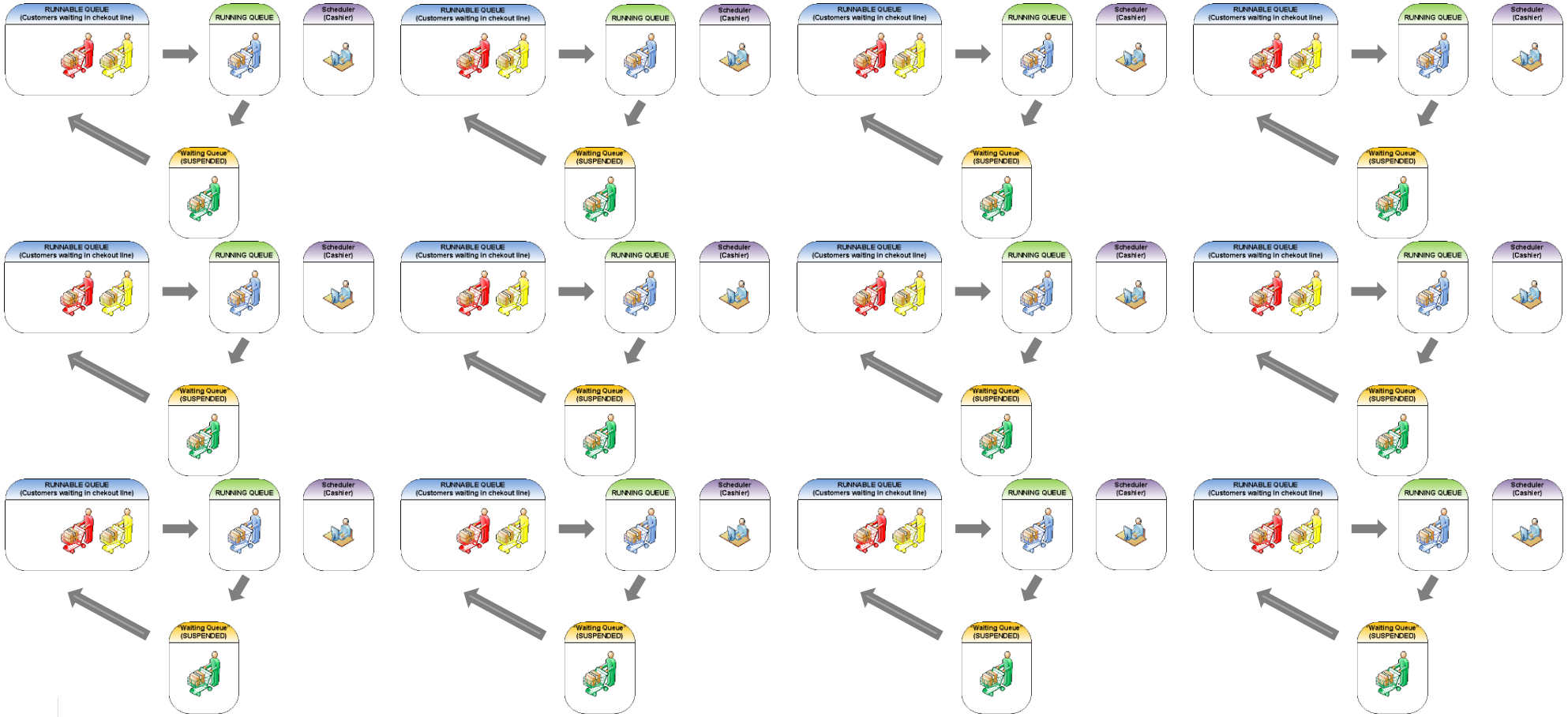
Execution Model – 1 Scheduler



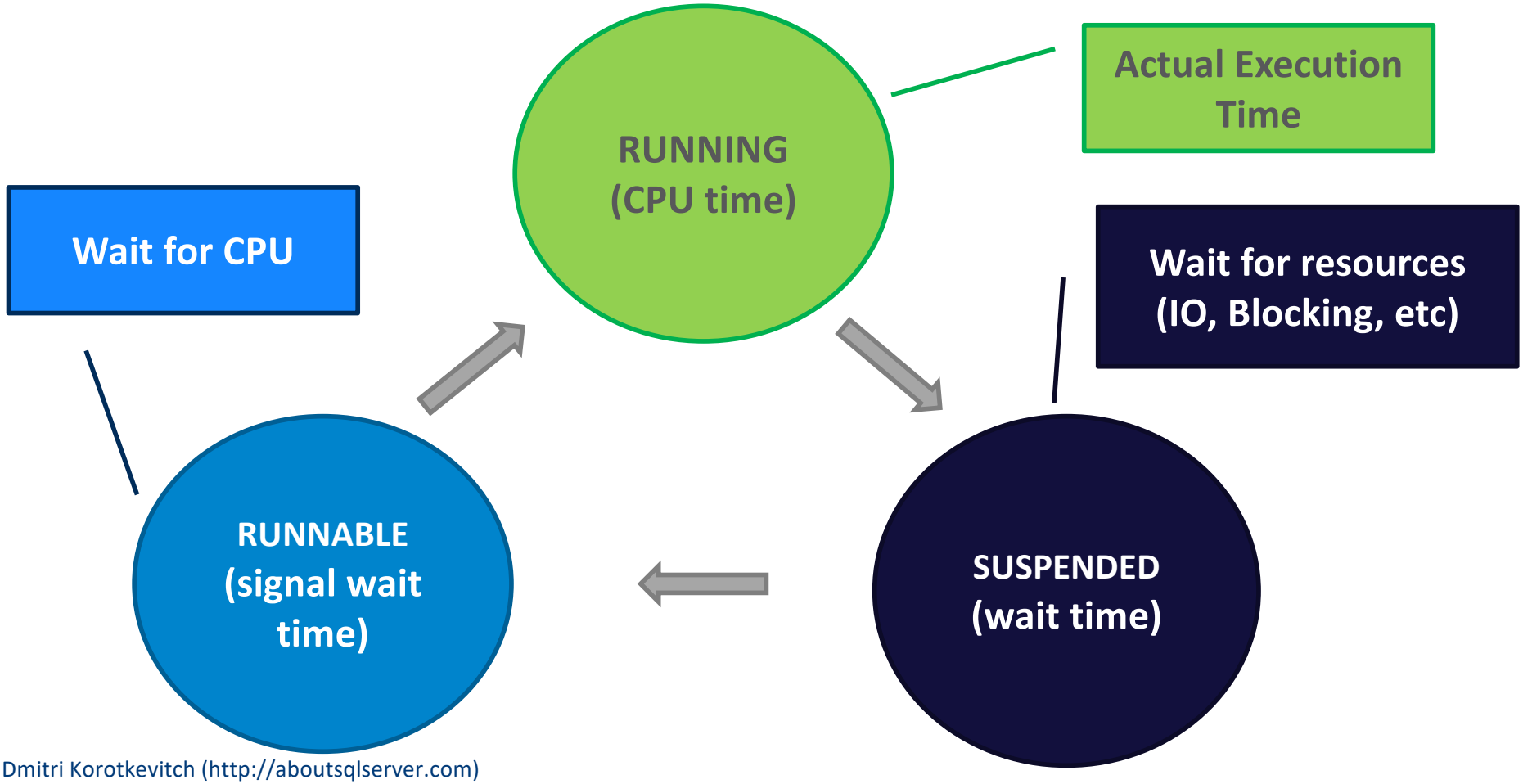
Execution Model – 1 Scheduler



More than 1 CPU?



Query Life Cycle (Simplified)



Wait Statistics 101

What are we waiting for?

```
select
    wait_type, wait_time_ms, waiting_ta
    ,wait_time_ms - signal_wait_time_ms
    ,100. * wait_time_ms / SUM(wait_tim
from sys.dm_os_wait_stats with (nolock)
where
    wait_type not in /* Filtering out n
    (N'BROKER_EVENTHANDLER',N'BROKER_RE
    ,N'BROKER_TO_FLUSH',N'BROKER_TRANSM
    ,N'CLR_SEMAPHORE',N'CLR_AUTO_EVENT'
    ,N'DBMIRROR_DBM_EVENT',N'DBMIRROR_E
    ,N'DBMIRRORING_CMD',N'DIRTY_PAGE_PO
    ,N'EXECSYNC',N'FSAGENT',N'FT_IFTS_S
    ,N'HADR_CLUSAPI_CALL',N'HADR_FILEST
```

	Wait Type	Wait Count	Wait Time	Avg Wait Time	Signal Wait Time	Avg Signal Wait Time	Resource Wait Time	Avg Resource Wait Time	Percent	Running Percent
1	LCK_M_U	548216435	2430865.645	4.0	148157.438	0.0	2282708.207	4.0	37.644	37.644
2	CXPACKET	696649076	1991476.659	2.0	224445.736	0.0	1767030.923	2.0	30.840	68.484
3	LCK_M_S	609624311	327349.394	0.0	326909.609	0.0	439.785	0.0	5.069	73.553
4	HADR_SYNC_COMMIT	393862506	277294.283	0.0	128633.010	0.0	148661.273	0.0	4.294	77.847
5	PAGEIOLATCH_EX	155976551	229145.101	1.0	3258.831	0.0	225886.270	1.0	3.548	81.395
6	BACKUPIO	84529112	161681.111	1.0	2084.816	0.0	159596.295	1.0	2.504	83.899
7	ASYNC_IO_COMPLETION	48377	157225.551	3250.0	86.153	1.0	157139.398	3248.0	2.435	86.334
8	LATCH_EX	171800900	141038.765	0.0	34869.237	0.0	106169.528	0.0	2.184	88.518
9	PAGEIOLATCH_SH	97400378	120563.694	1.0	1936.229	0.0	118627.465	1.0	1.867	90.385
10	BACKUPBUFFER	120290500	109741.003	0.0	7385.949	0.0	102355.054	0.0	1.699	92.084
11	PAGELATCH_EX	3103005547	89580.939	0.0	84153.002	0.0	5427.937	0.0	1.387	93.472
12	BACKUPTHREAD	527984	69381.750	131.0	180.629	0.0	69201.121	131.0	1.074	94.546
13	LCK_M_IX	21926	53148.574	2423.0	25.892	1.0	53122.682	2422.0	0.823	95.369

Wait Statistics 101

sys.dm_os_wait_stats view – historical information (server)
sys.dm_exec_session_wait_stats view (2016+) - session

- **wait_type**: type of the wait
- **waiting_task_count**: number of waits
- **wait_time_ms**: cumulative wait time
- **signal_wait_time**: wait in *runnable* state

DBCC SQLPERF('sys.dm_os_wait_stats', CLEAR)

- Make sure stats is representative

Wait Statistics 101

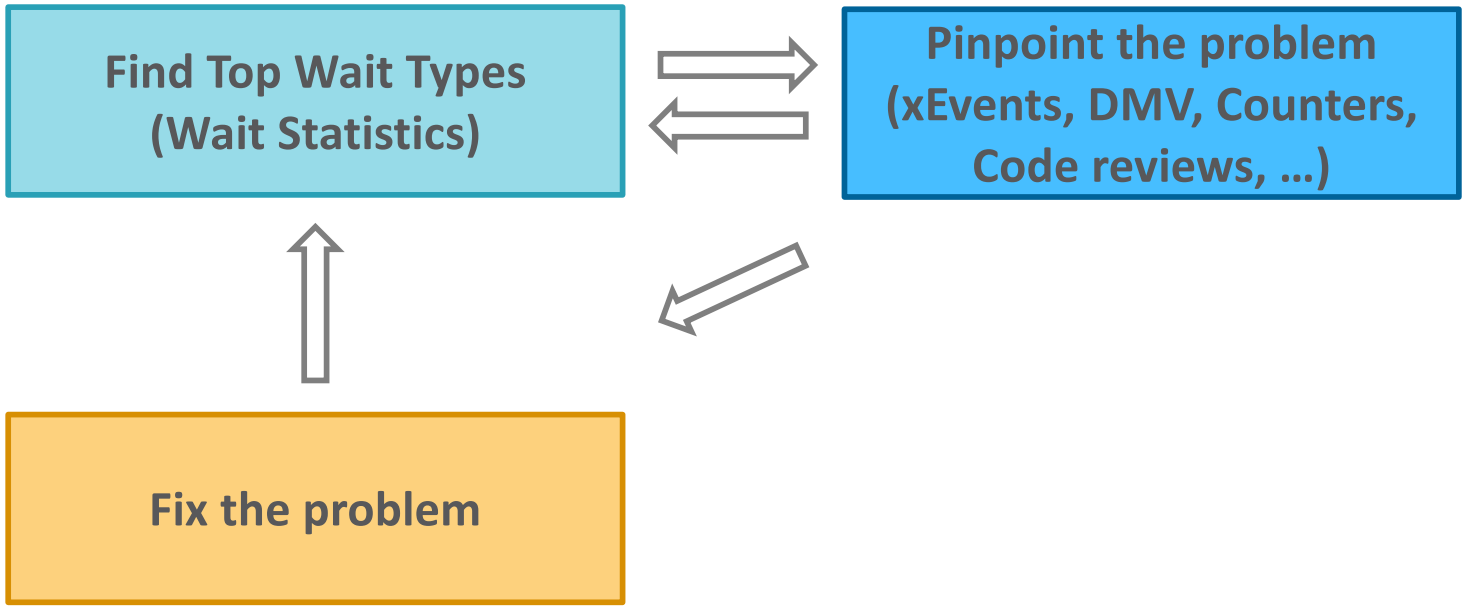
sys.dm_os_waiting_tasks view – what is happening now

- **wait_type**: type of the wait
- **wait_duration_ms**: current wait time
- **resource_description**: resource information
- **blocking_session_id**: blocker, when applicable

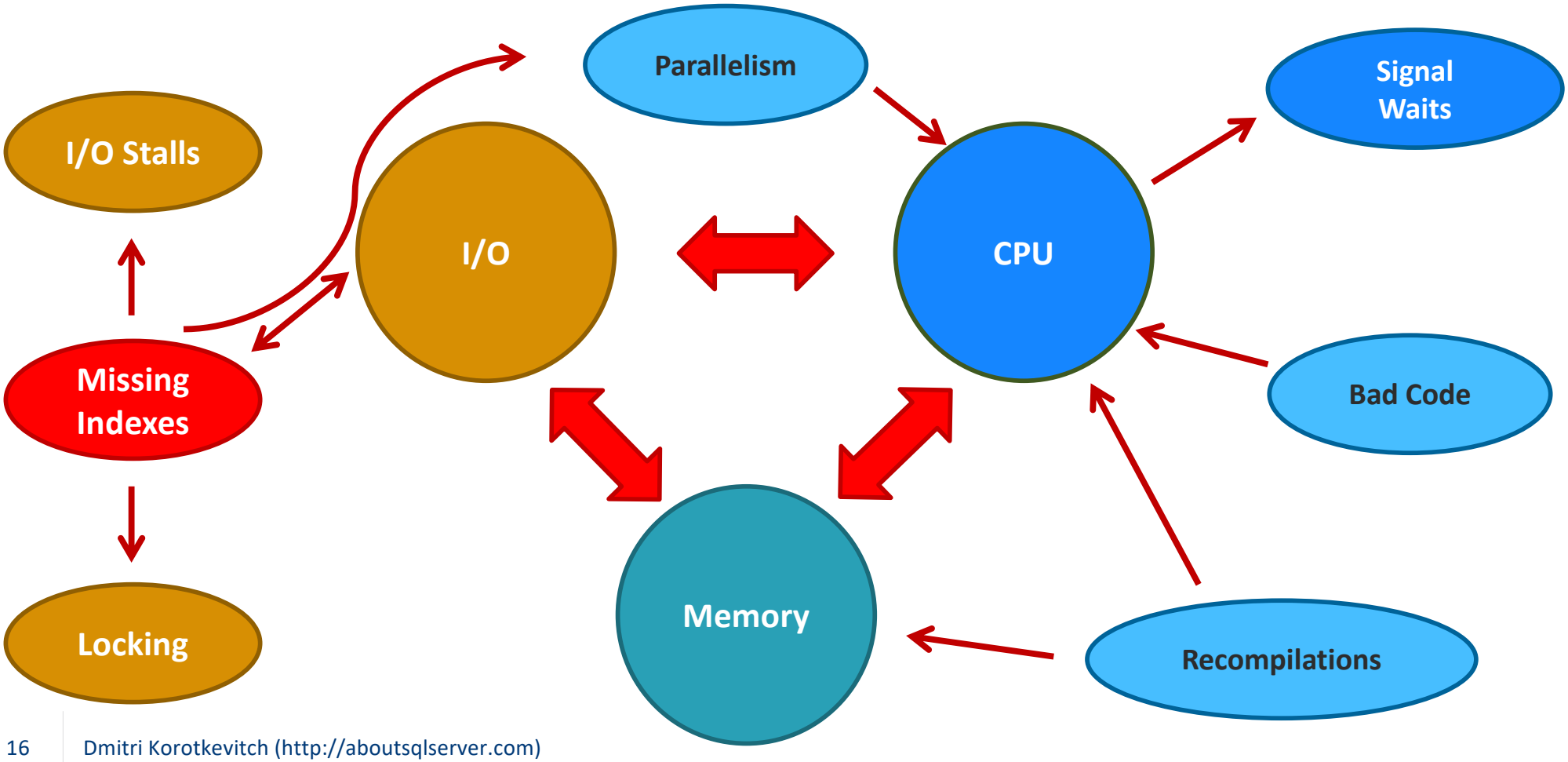
sys.dm_exec_requests view – request information

- **status**: running, runnable, suspended, etc
- **wait_type, wait_time, wait_resource, blocking_session_id:** wait information

Never-Ending Troubleshooting

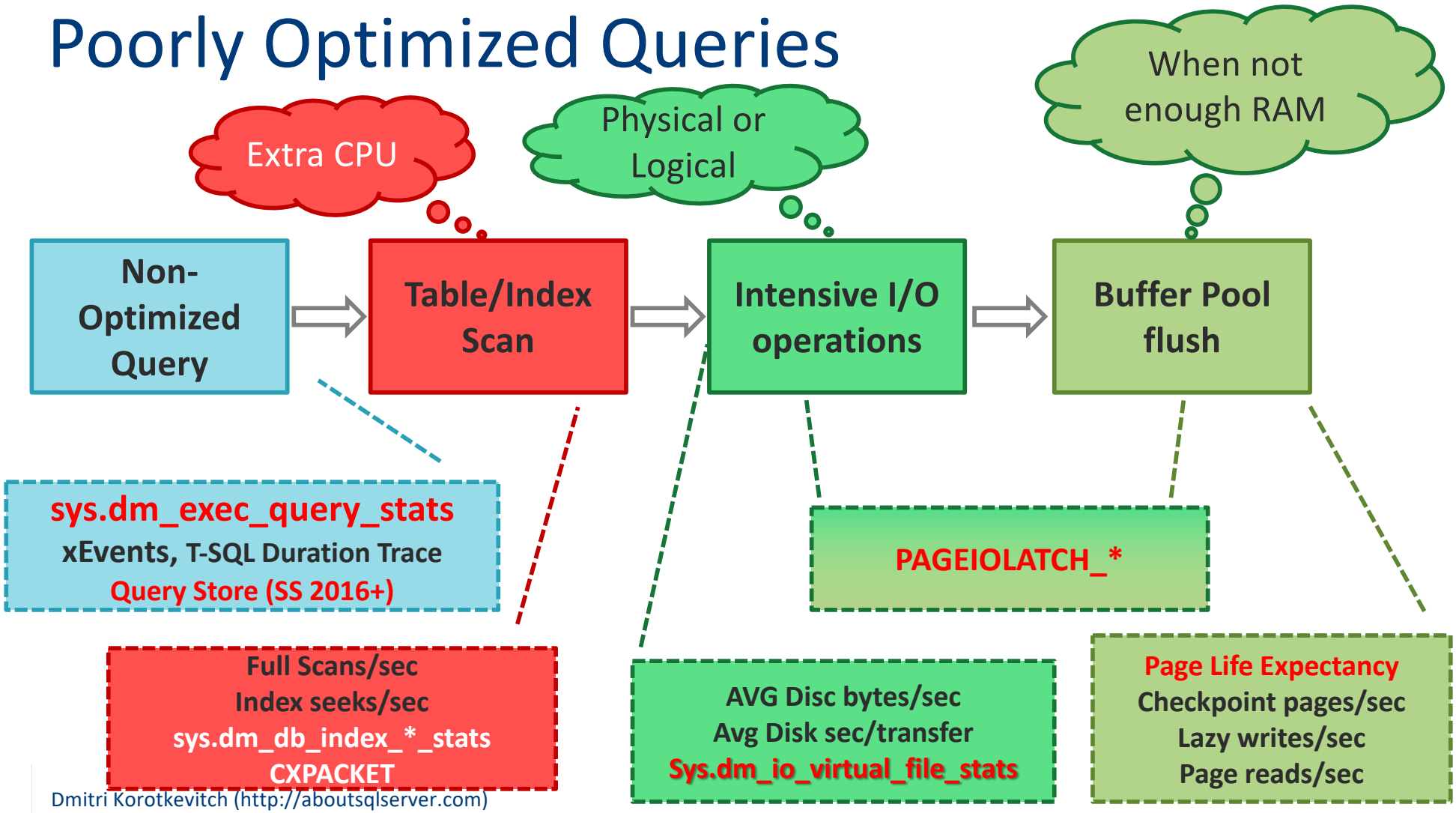


Everything is Related



Poorly Optimized Queries

Poorly Optimized Queries



Key Metrics

PAGELATCHIO* waits

- Waits for data page reads
- Physical data I/O only

Page Life Expectancy

- How long data page stays cached in buffer pool
- Rule of thumb: 300 * 4GB of SQL Server memory

CXPACKET in OLTP

- Complex plans -> Parallelism (more later)

Disk Throughput (sys.dm_io_virtual_file_stats)

	DB ID	File Id	File Name	File Path	Type	Time	Reads	Read Bytes	Writes	Written Bytes	IO Count	Read %	Write %	Read Stall	Write Stall	Avg Read St...	Avg Write Stall
1	5	1	CTCloud_Primary_file	F:\Databas...	ROWS	2082125133	28640	634617856	5777438	74420396032	5806078	0.85	99.15	234236	20120804	8.179	3.483
2	5	2	CTCloud_Log_file	F:\Databas...	LOG	2082125133	5786691	4856755533312	255007793	11700852344320	260794484	29.33	70.67	18821891	511247441	3.253	2.005
3	5	3	CTCloud_Entities_file1	F:\Databas...	ROWS	2082125133	3836810	213045534720	65903077	792104493056	69739887	21.20	78.80	25738654	307697614	6.708	4.669
4	5	4	CTCloud_Entities_file2	F:\Databas...	ROWS	2082125133	3606346	164764073984	45587023	547723214848	49193369	23.13	76.87	24074753	100938420	6.676	2.214
5	5	5	CTCloud_Entities_file3	F:\Databas...	ROWS	2082125133	3707361	131695001600	89880283	1003102298112	93587644	11.61	88.39	19261189	144064060	5.195	1.603

Look at overall latency

- Targets for t-log write stalls < 3ms
- Targets for read stalls < 5-10ms
- Targets for write stalls < 3-5ms

Look at throughput (especially in tempdb)

Detecting Poorly Optimized Queries

Query Store (SS 2016+)

- OK to enable in majority of the systems
  - Careful with heavy ad-hoc workload -> Do not enable “Capture All”
- Monitor QDS* waits
- Be on the recent SP

Plan Cache-based execution statistics

- sys.dm_exec_query_stats
- sys.dm_exec_procedure_stats
- sys.dm_exec_function_stats (SS2016+)

xEvents

- Introduce overhead but may be acceptable for quick profiling

sys.dm_exec_query_stats

```
SELECT TOP 250
    SUBSTRING(qt.TEXT, (qs.statement_start_offset/2)+1,
        ((
```

	SQL	Exec ...	Avg IO	query_plan	Total Reads	Total Writes	Total CPU
1	select Subj, cast(R...	1	6816382	<ShowPlanXM...	6816296	86	24297389
2	select UID, DOCTY...	26455	4143503	<ShowPlanXM...	109616393555	0	154369131409
3	DELETE TOP (@d...	1	4096631	<ShowPlanXM...	4096468	163	26538518
4	insert into #tmpRep...	62	3690210	NULL	228750206	42859	3351099613
5	update #tmpReportl...	62	3139967	NULL	194677952	7	2406888686
6	insert into #tmpRep...	58	2516483	NULL	145905711	50341	1761652781
7	select D.*, O.CATE...	16	1848720	<ShowPlanXM...	29579527	0	64629691
8	update #tmpReport...	13	1520333	<ShowPlanXM...	19764334	5	194722131
9	select D.*, I.Catego...	36	1511917	<ShowPlanXM...	54429042	0	114735561
10	update #tmpReport...	26	1459946	<ShowPlanXM...	37958482	138	447010567
11	update #tmpReport...	12	1426777	<ShowPlanXM...	17121325	4	164099386
12	insert into #tmpRep...	53	1079374	NULL	57198359	8467	865721533

```
        CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
        OUTER APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
ORDER BY
    [Avg IO] desc
option (recompile)
```

Works based on plan cache

Poorly Optimized Queries

Type	Metric	Description
Wait Types	**PAGEIOLATCH_***	Disk to memory transfer (data pages)
Performance Objects	~~Buffer Cache Hit Ratio~~	How often page is found in the buffer pool - do not use
	Page Life Expectancy	How long page stays in the cache – watch the trends. As the starting point should be > (Buffer Pool Size / 4GB) * 300
	Checkpoint Pages/Sec Lazy Writes/Sec	How often pages are saved to disk Memory Pressure: High values + low Page Life Expectancy
	Page Reads/Sec	Number of physical page reads per sec – watch the trends
	Avg Disk Bytes/Transfers	Disk performance counters

Poorly Optimized Queries

Type	Metric	Description
DMV	**sys.dm_exec_query_stats**	Query execution statistics
	sys.dm_io_virtual_file_stats	I/O statistics on per-file basis
	sys.dm_os_memory_clerks DBCC MEMORYSTATUS	Memory consumers
xEvents	rpc_completed, sql_statement_completed	Filter by I/O, CPU or Duration metrics. Be careful on the busy system
Query Store		SQL Server 2016+, Azure SQL DB

Optimization Approach

Analyze SQL Query

- Check SARGability, functions, data type conversion, etc

Check Actual vs. Estimated # of rows

- Outdated statistics?
- Check join types

Analyze efficiency of Index Seeks

- # of rows read; actual # of rows

Address large scans and Key Lookups

Other I/O Waits

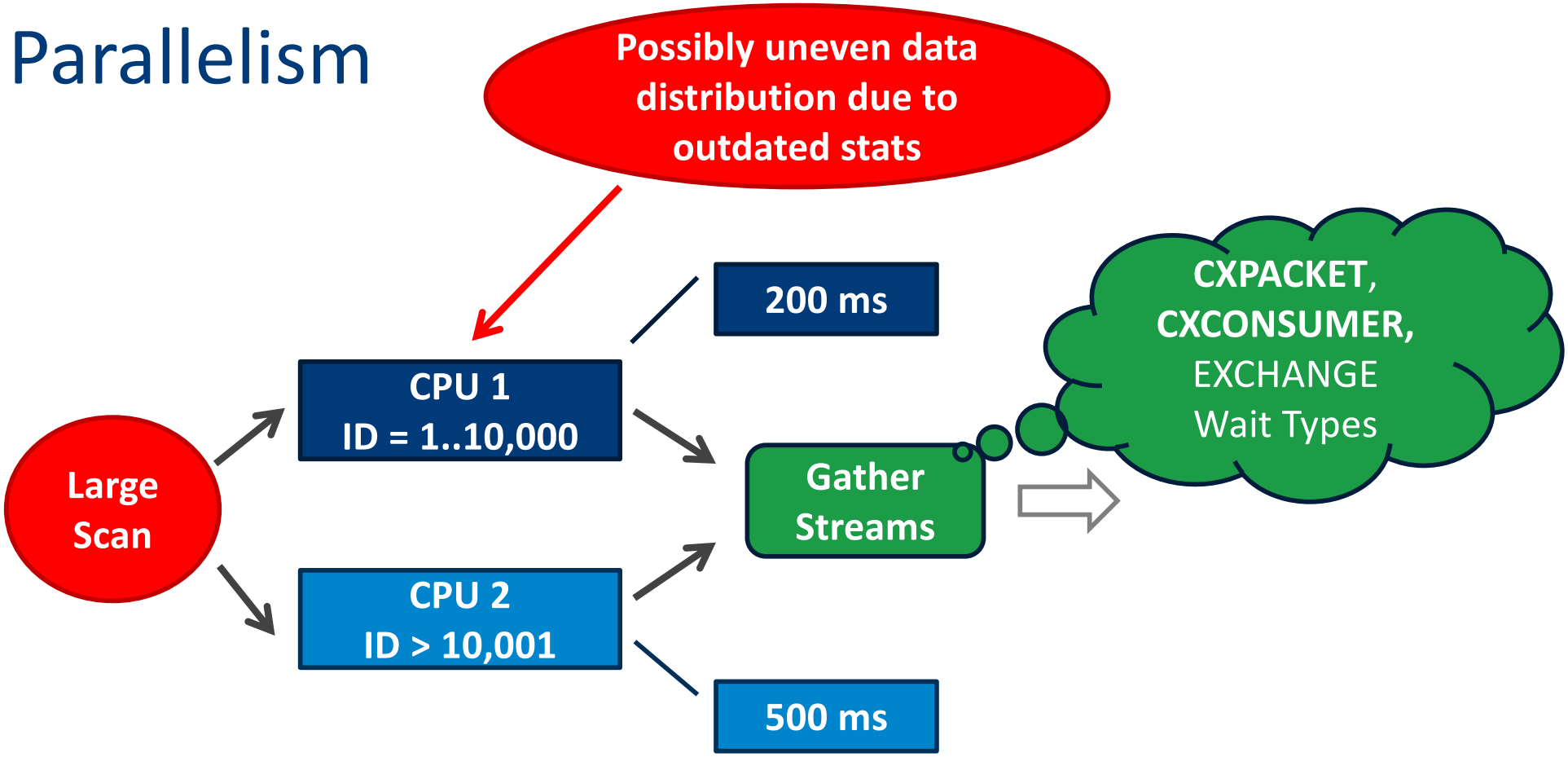
IO_COMPLETION

- Often tempdb performance
- Check tempdb spills

ASYNC_IO_COMPLETION

- Usually Data Backup related
- Check performance of backup drive and process throughput
- Often comes with BACKUPIO, BACKUPTHREAD waits

Parallelism



Parallelism

Tuned OLTP queries rarely benefit from parallelism

Parallelism is required for DW workload

Consider to increase “Cost Threshold for Parallelism” rather than changing MAXDOP

- You can see plan cost at the top element of the execution plan

Monitor statistics

When Everything is Cached..

Non-optimized systems with DB cached in buffer pool

- Low PAGEIOLATCH_* waits and high *Page Life Expectancy*
- Often IO_COMPLETION waits and heavy tempdb usage/latency
- High CXPACKET waits unless MAXDOP = 1
- Often high CPU usage and signal time
- Sometimes blocking

Look at queries I/O metrics in plan cache / Query Store

"Do nothing" is often the best solution..

- Monitor that *hot* data fits into the memory

High CPU

Possible Causes

Current hardware / workload trends

Inefficient queries

- General performance tuning usually helps

Bad T-SQL code

- Functions, cursors, XML, data type conversions, etc

Recompilations

- Especially with heavy ad-hoc workload

Metrics

SQL Server vs App CPU

- sys.dm_os_ring_buffers (script is included)
- Process: % Priveleged Time (kernel mode); % User Time (user mode)

Wait statistics: signal_wait_time_ms

- SOS_SCHEDULER_YIELD wait type
- signal_wait_time_ms – <15% of wait_time_ms in OLTP

Check for “Observer Effect”, audit, etc

Schedulers and NUMAs

Evenly divide schedules across NUMA nodes

- Especially with AlwaysOn Availability Groups in busy OLTP systems

```
select
    parent_node_id
    ,scheduler_id
    ,status
    ,current_tasks_count
    ,runnable_tasks_count
from
    sys.dm_os_schedulers
where
    status = 'VISIBLE ONLINE'
order by
    scheduler_id
```

	parent_node_id	scheduler_id	status	current_tasks_count	runnable_tasks_count
1	0	0	VISIBLE ONLINE	7	0
2	0	1	VISIBLE ONLINE	6	0
3	0	2	VISIBLE ONLINE	4	0
4	0	3	VISIBLE ONLINE	6	0
5	0	4	VISIBLE ONLINE	4	0
6	0	5	VISIBLE ONLINE	4	1
7	0	6	VISIBLE ONLINE	6	0
8	0	7	VISIBLE ONLINE	7	0
9	1	8	VISIBLE ONLINE	19	0
10	1	9	VISIBLE ONLINE	21	0

Metrics

Queries

- Execution statistics (plan cache and/or query store)
- Currently executed: sys.dm_exec_requests.cpu_time
- Patterns:
  - One bad query
  - Death by thousand cuts
  - Parameter sniffing-related issues

Recompilations

- In OLTP systems:
  - Initial Compilations = SQL Compilations/Sec – SQL Re-Compilations/Sec
  - Plan Reuse = (Batch Requests/Sec – Initial Compilations) / Batch Requests/Sec > 90% - 95%

Addressing Recompilations

Parameterize the code!

Consider FORCED PARAMETERIZATION

- Beware of Parameter Sniffing-related issues
- Consider OPTIMIZE FOR UNKNOWN
  - SQL Server 2016 – Database Scoped Configuration
  - SQL Server prior 2016 – TF 4136

Cautionary Tale of Recompilation, Plan Caching and High CPU Usage - demos

- Available at: https://aboutsqlserver.com/presentations

Memory Grants

Optimize Queries; Check Statistics

Queries need memory to execute

- Some memory to execute + memory for SORT and HASH + parallelism
- By default query can use 25% of workspace memory (~25% of 75%)
  - Can be changed in Resource Governor

SQL Server uses 2 semaphores for memory grant allocations

- Queries will wait until memory is available

Metrics

- RESOUCE_SEMAPHORE wait
- SQL Server:Memory Manager\Memory Grants Pending > 0
- sys.dm_exec_query_stats – memory grant-related columns
- Query Store

Other Waits

RESOURCE_SEMAPHORE_COMPILE

- Wait for memory grant during compilations
- Causes:
  - Excessive compilations
  - I saw issues with <u>async</u> statistics update during online index rebuild on large tables in busy OLTP systems

CMEMTHREAD

- Usually related to plan cache management / memory allocation
- Often happens with old versions of SQL Server on modern hardware
  - T8048 may help <SS 2016
- Reduce excessive compilations

ASYNC_NETWORK_IO

Server waits for client to consume data

Possible causes

- Network performance
- Excessive amount of data to send (select *)
- Application issues

ASYNC_NETWORK_IO - Bad

```
using (SqlDataReader reader = cmd.ExecuteReader())
{
    while (reader.Read())
    {
        ProcessOrder((IDataRecord)reader);
    }
}
```

ASYNC_NETWORK_IO - Good

```
List<Orders> list = new List<Orders>();
using (SqlDataReader reader = cmd.ExecuteReader())
{
    while (reader.Read())
    {
        list.Add(ReadOrder((IDataRecord)reader));
    }
}
ProcessAllOrders(list);
```

Locking & Blocking

Locks acquired on resources (rows, pages, tables, db, etc)

There are many lock types

- Exclusive (X) – acquired when data is modified and held till end of tran
- Update (U) – update scans
- Shared (S) – acquired by readers (SELECT) in some isolation levels
- Intent (IS, IU, IX) - indicate locks on child objects
- Schema Modification (Sch-M) - exclusive access to the object during alteration
- Schema Stability (Sch-S) – protects object from alteration if there is no I* locks

Lock Escalation

- SQL Server may replace row-level with table-level locks during batch operations

Locking Compatibility

	IS	IU	IX	S	U	X	Sch-S	Sch-M
IS								
IU								
IX								
S								
U								
X								
Sch-S								
Sch-M								

Majority of blocking issues are due to non-optimized queries

Wait Types

Metric	Possible Causes
LCK_M_SCH_*	Index and/or partition maintenance, frequent schema modifications (app issues)
LCK_M_I*	Lock escalation, schema modifications (see LCK_M_SCH*)
LCK_M_RS*	SERIALIZABLE transactions, IGNORE_DUP_KEY option in NCI
LCK_M_U	Nonoptimized DML queries. Perform query tuning. Analyze transaction management
LCK_M_S	Nonoptimized SELECT queries. Perform query tuning. Consider RCSI to hide the problem.

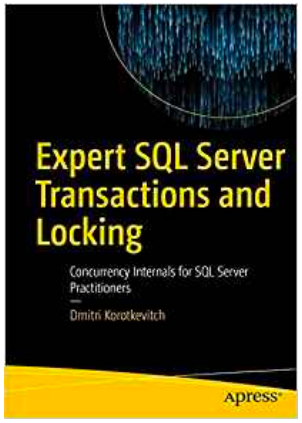
Locking & Blocking – Additional Info

Additional resources from my blog

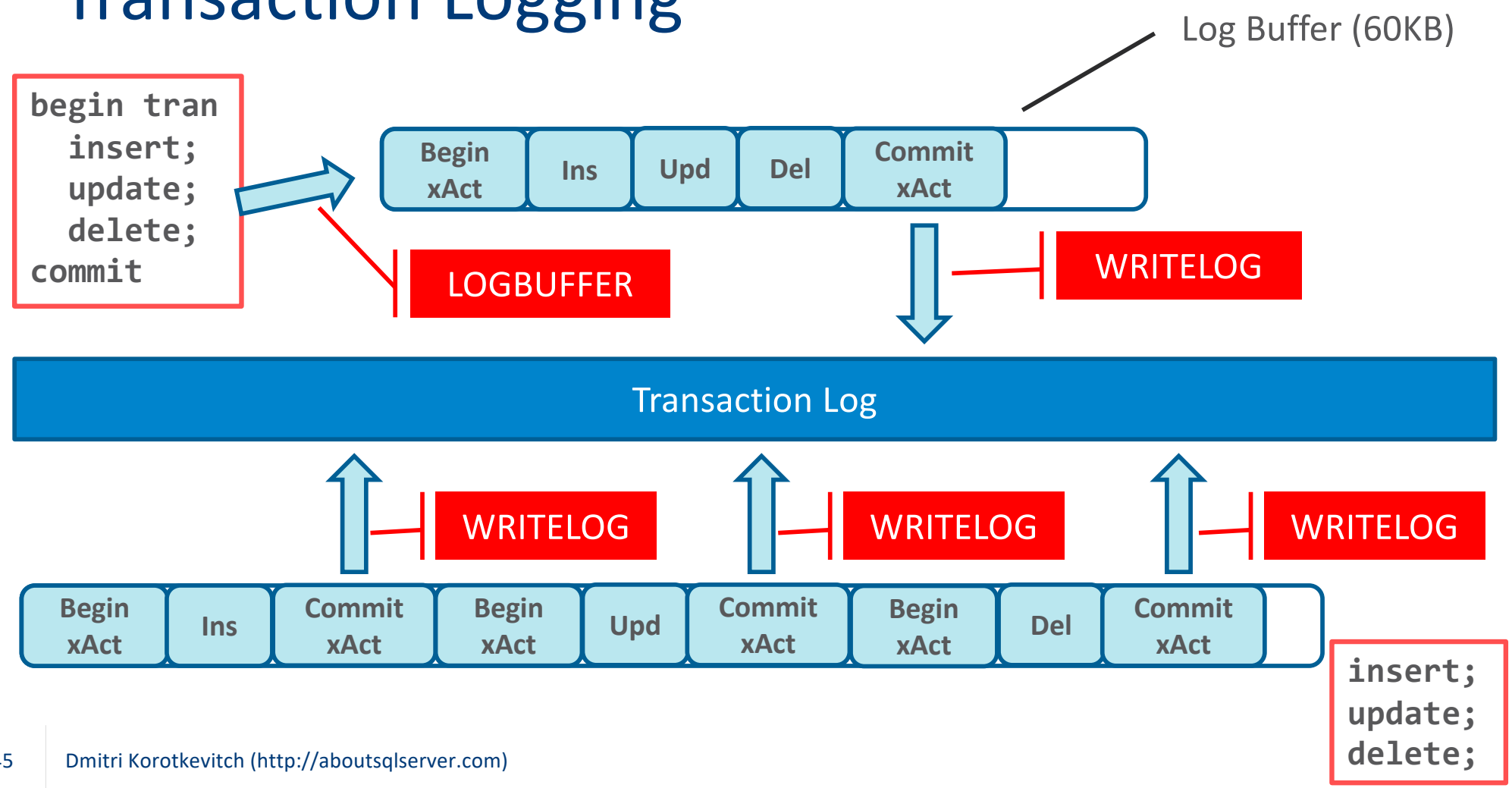
- *Deep dive into Blocking and Deadlock troubleshooting* presentation
  - Slides and demos are available at: https://aboutsqlserver.com/presentations/
- Locking and Blocking posts: https://aboutsqlserver.com/lockingblocking/
- Blocking Monitoring Framework: https://aboutsqlserver.com/bmframework/

Expert SQL Server Transactions and Locking book

- Available from Amazon.com and Torrent ☺



Transaction Logging



WRITELOG and LOGBUFFER waits

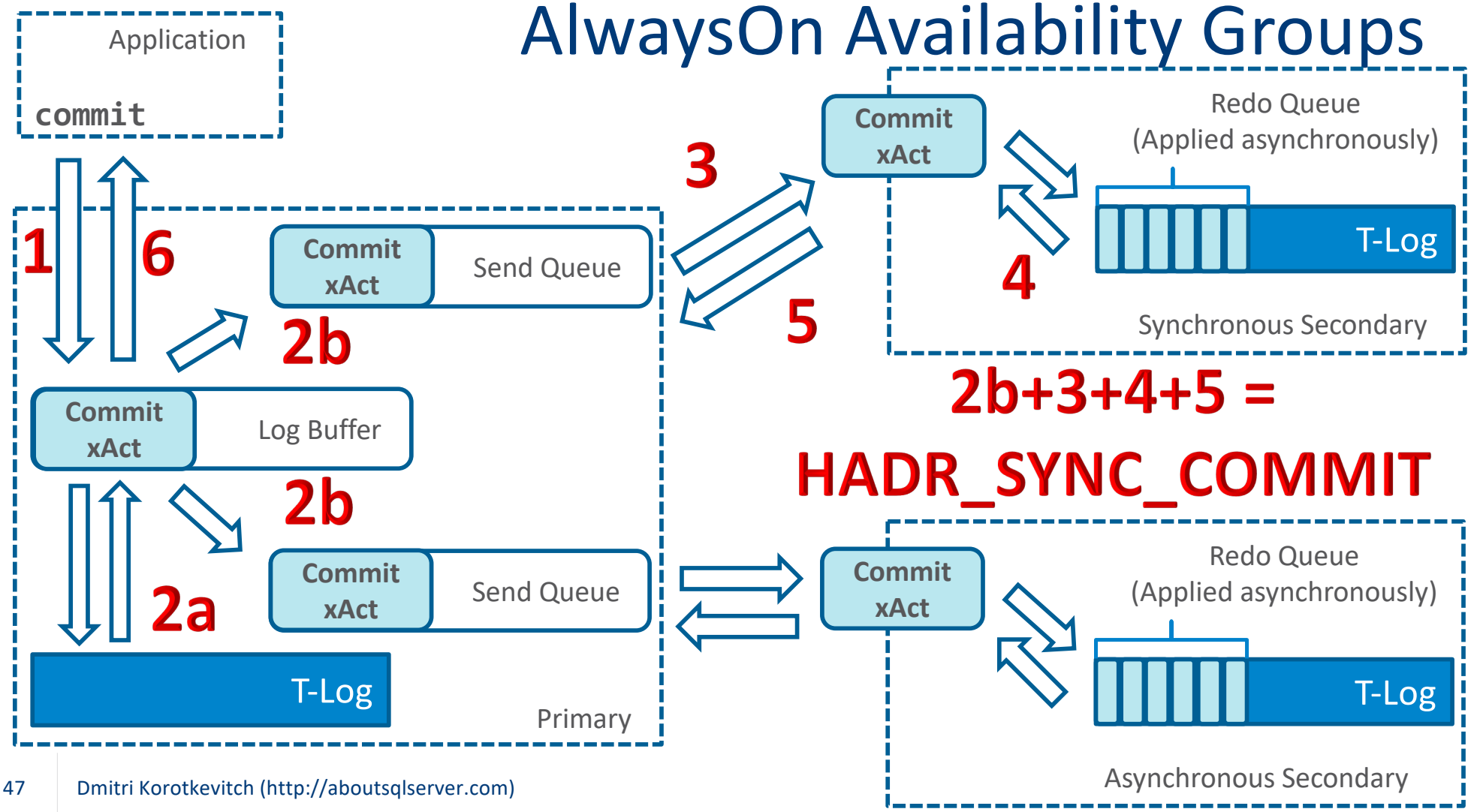
Analyze

- Log drive throughput (sys.dm_io_virtual_file_stats)
- Log generation rate
- Code (autocommitted transaction)

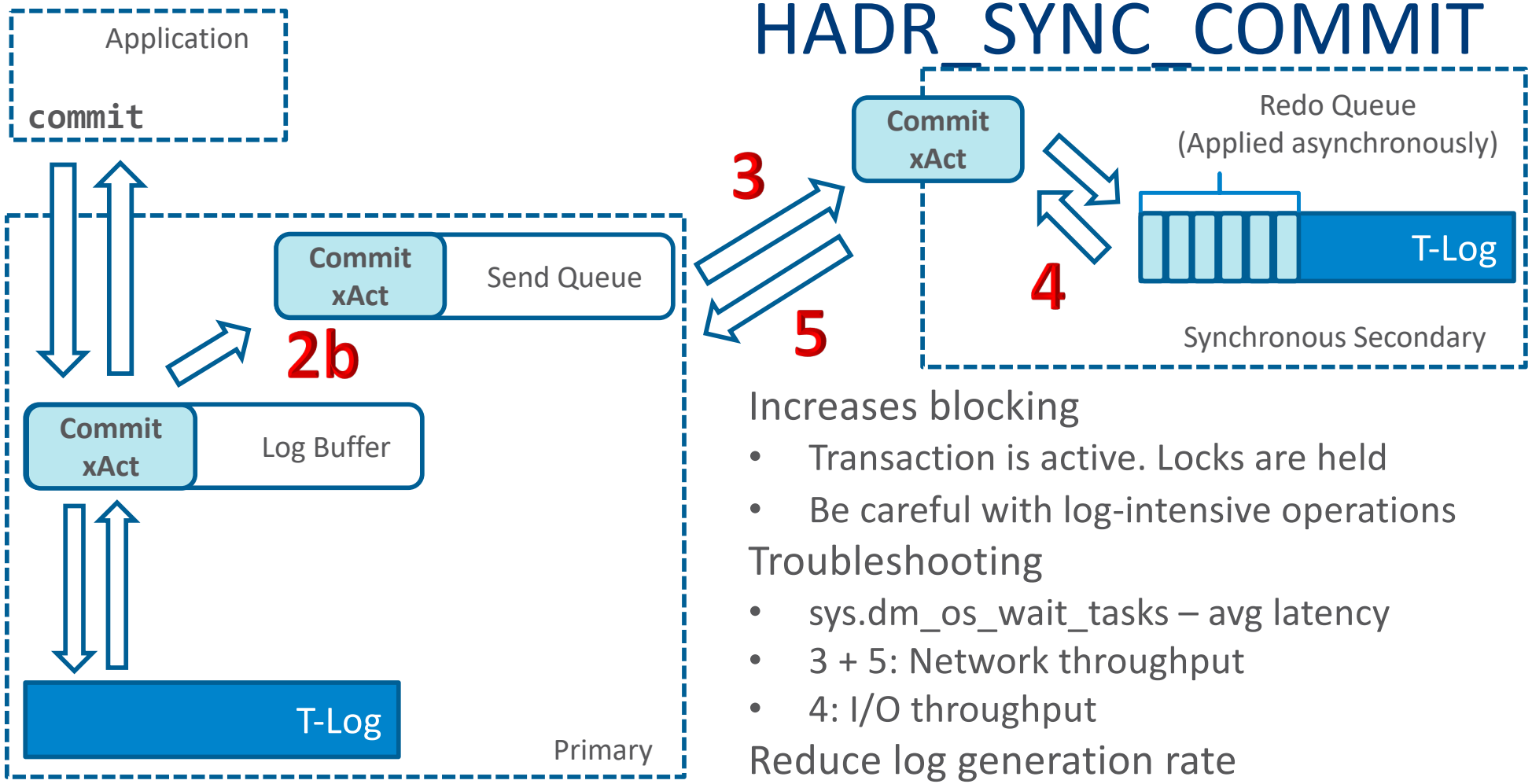
Possible solutions

- Upgrade I/O subsystem
- Reduce t-log activity
- Consider delayed durability (possible data loss)
- Reduce data I/O if everything is on the same drive

AlwaysOn Availability Groups



HADR_SYNC_COMMIT



Increases blocking

- Transaction is active. Locks are held
- Be careful with log-intensive operations

Troubleshooting

- sys.dm_os_wait_tasks – avg latency
- 3 + 5: Network throughput
- 4: I/O throughput

Reduce log generation rate

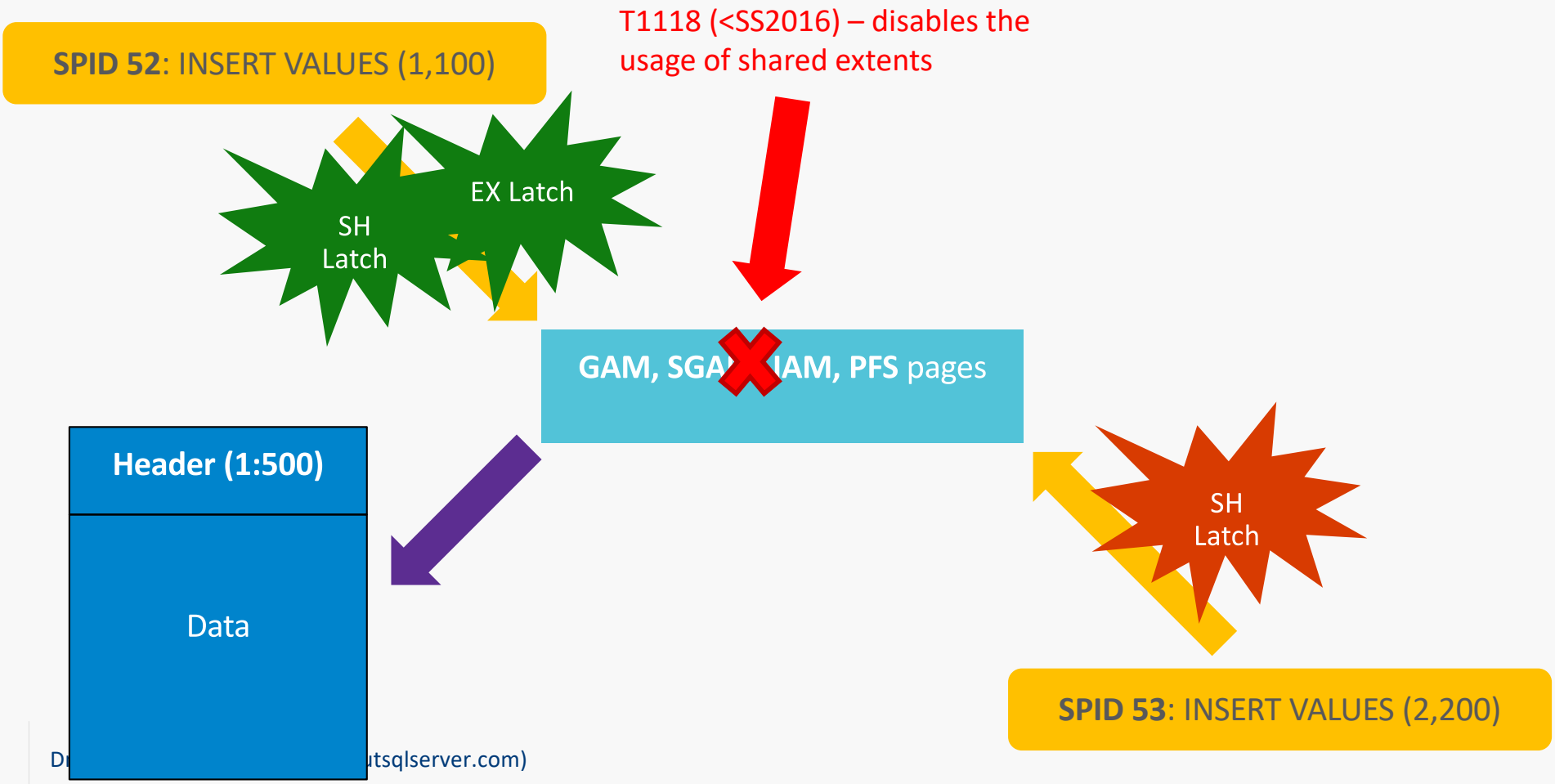
Upgrade to SQL Server 2016+

HADR_SYNC_COMMIT & AG Issues

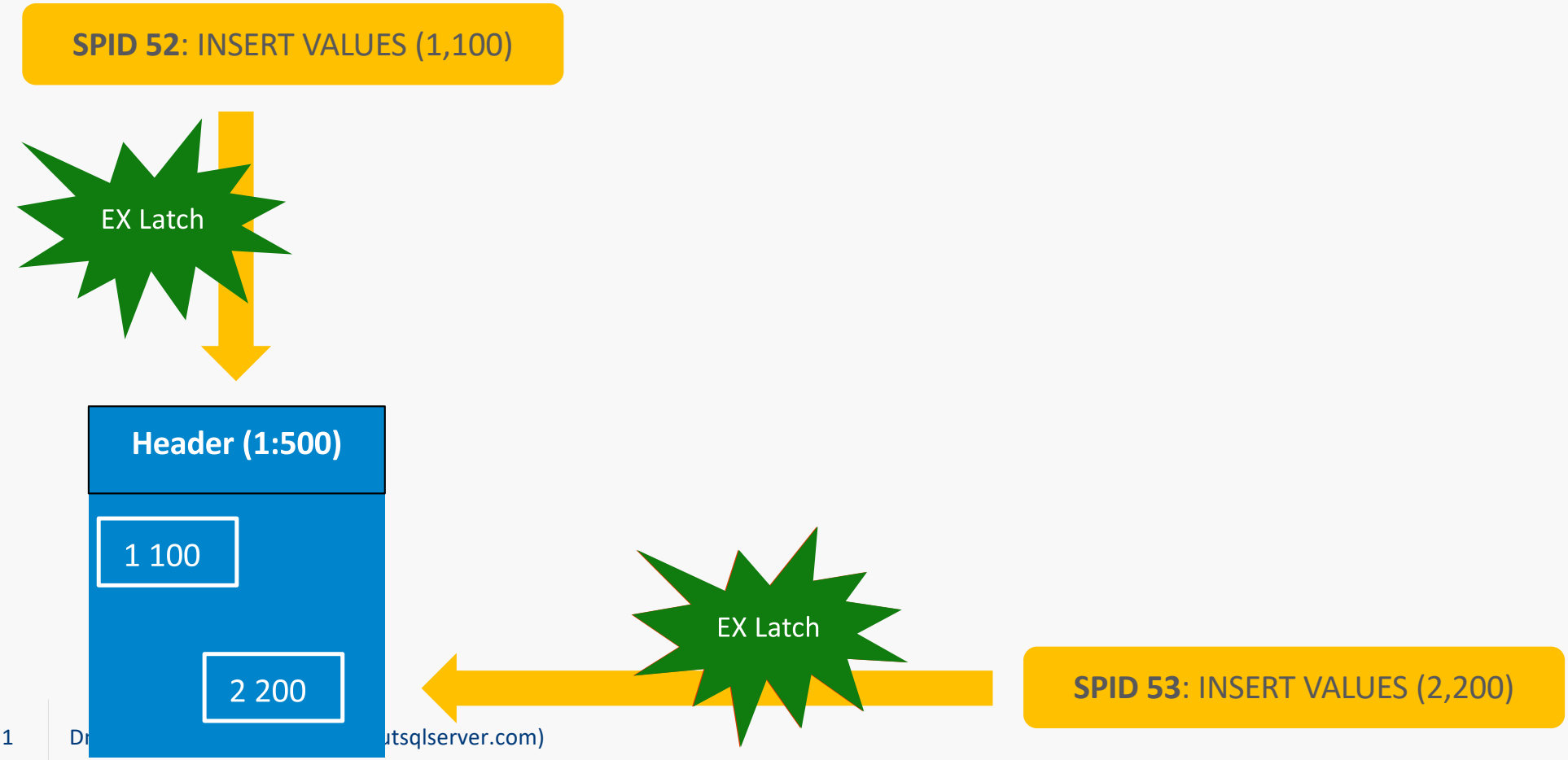
Five Availability Group Issues That May Ruin Your Day

- Slides and demos are available at: https://aboutsqlserver.com/presentations

Latches


SPID 52: INSERT VALUES (1,100)
T1118 (<SS2016) – disables the usage of shared extents
SH Latch
EX Latch
GAM, SGA IAM, PFS pages
Header (1:500)
Data
SH Latch
SPID 53: INSERT VALUES (2,200)

Latches


SPID 52: INSERT VALUES (1,100)
EX Latch
Header (1:500)
1 100
2 200
EX Latch
SPID 53: INSERT VALUES (2,200)

Latches

Protects internal data structures

Wait statistics exposes 3 major latch types

- PAGEIOLATCH_* - data pages-related I/O
- PAGELATCH_* - access to data pages in memory
- LATCH_* - general latch

PAGELATCH troubleshooting

- Capture resource with *sys.dm_os_waiting_tasks* and/or xEvents
- DBID=2: (2,1,1), (2,1,3) – tempdb allocations (T1118 + add files)
- User databases: Most likely hot spots. Find *object_id* (DBCC PAGE) and add data files and/or change the schema.

sys.dm_db_index_operational_stats

Access methods, I/O, locking, latching activity

- Insert, Update, Delete counts (# of rows)
- singleton_lookup_count: Single-row Index Seek operations
- range_scan: Index Seeks on the range of rows + Index Scans
- LOB and ROW_OVERFLOW statistics
- Lock counts and waits on row- and page-levels
- **Page latch count and waits**
- Page IO latch count and waits
- And more..

	index_id	Table	Index	range_scan_count	singleton_lookup_count	row_lock_wait_in_ms	page_latch_wait_in_ms	page_io_latch_wait_in_ms
1	1			1411162	3638897399	0	71286	13634302
2	2			774	0	0	338	283589
3	3			760095	0	0	121	329284
4	4			32726	0	0	828	7878183
5	5			358	0	0	66	3138358
6	6			21	0	0	124	138602
7	8			0	0	0	519	68238
8	33			1	0	0	60	574234
9	34			11012	0	0	490	626016
10	35			0	0	0	126	23540

Latches (sys.dm_os_latch_stats)

	Latch Type	Wait Count	Wait Time	Avg Wait Time	Percent	Running Percent
1	ACCESS_METHODS_DATASET_PARENT	25546519	51728.454	2.0	92.530	92.530
2	LOG_MANAGER	938	2805.912	2991.0	5.019	97.550

Parallelism: ACCESS_METHOD_DATASET_PARENT, ACCESS_METHOD_SCAN_*, NESTING_TRANSACTION_FULL

LOG_MANAGER: Usually transaction log is growing. Check why it is not truncating.

FGCB_ADD_REMOVE: Growing, shrinking files in the filegroup. Check *Instant File Initialization* and *Auto Shrink* option.

ACCESS_METHOD_HOBT_VIRTUAL_ROOT: Can be large amount of page splits.

ACCESS_METHOD_HOBT_COUNT: Heavy concurrent data modifications

TRACE_CONTROLLER: Excessive amount of traces

THREADPOOL Waits

Workers starvation – SQL Server cannot assign workers to the new requests

Sign of serious issues – even in small %

Possible causes

- Memory Pressure / Low amount of memory in OS (especially in VMs)
- Long blocking chains
- Very large # of connections (bad connection management)

THREADPOOL Waits (sys.dm_os_waiting_tasks)

	session_id	wait_type	wait_duration_...	blocking_session...	resource_description
466	NULL	THREADPOOL	52907	NULL	threadpool id=scheduler2f92a0040
467	NULL	THREADPOOL	52906	NULL	threadpool id=scheduler2f92a0040
468	NULL	THREADPOOL	27866	NULL	threadpool id=scheduler2f92a0040
471	52	LCK_M_IS	58301	57	objectlock lockPartition=0 objid=2...
472	54	LCK_M_IS	58293	57	objectlock lockPartition=0 objid=2...
473	55	LCK_M_IS	58293	57	objectlock lockPartition=0 objid=2...
474	56	LCK_M_IS	58294	57	objectlock lockPartition=0 objid=2...
475	57	LCK_M_IS	58302	51	objectlock lockPartition=0 objid=2...
476	58	LCK_M_IS	58293	57	objectlock lockPartition=0 objid=2...

Use Dedicated Admin Connection to Troubleshoot

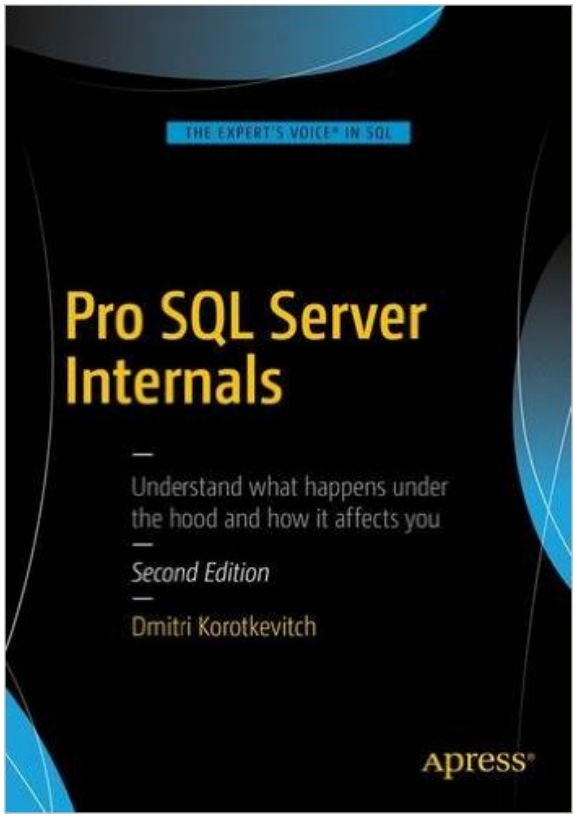
Additional Resources

Whitepapers:

- Original: https://technet.microsoft.com/en-us/library/cc966413.aspx
- SQLSkills: https://www.sqlskills.com/help/sql-server-performance-tuning-using-wait-statistics/

SQLSkills Wait Types Library: https://www.sqlskills.com/help/waits/

Additional Resources



Email me anytime:

dk@aboutsqlserver.com

Slides and Demos:

http://aboutsqlserver.com/presentations

Video:

Youtube Channel: aboutsqlserver

Thank You