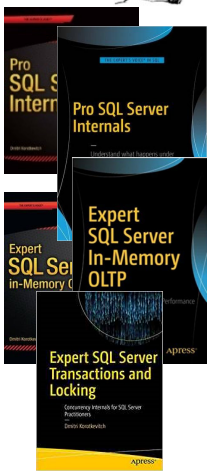
5 Availability Groups Issues That May Ruin Your Day

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
Director of Database Services @ chewy.com
SQL Server Consultant & Trainer
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**
Blog: http://aboutsqlserver.com
Email: dk@aboutsqlserver.com

Agenda

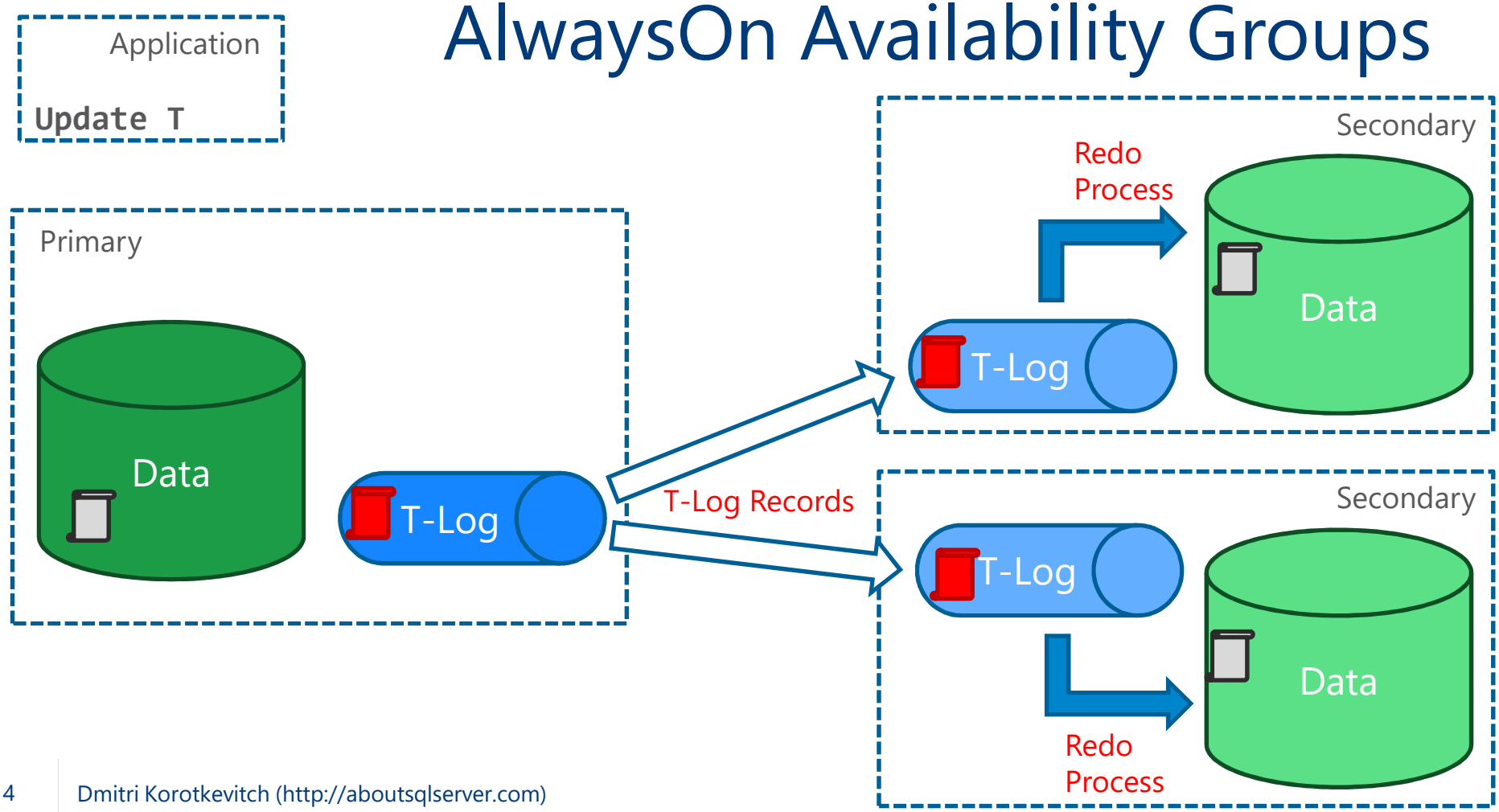
Availability Group Internals 101

Issues that occur when “everything works as designed”

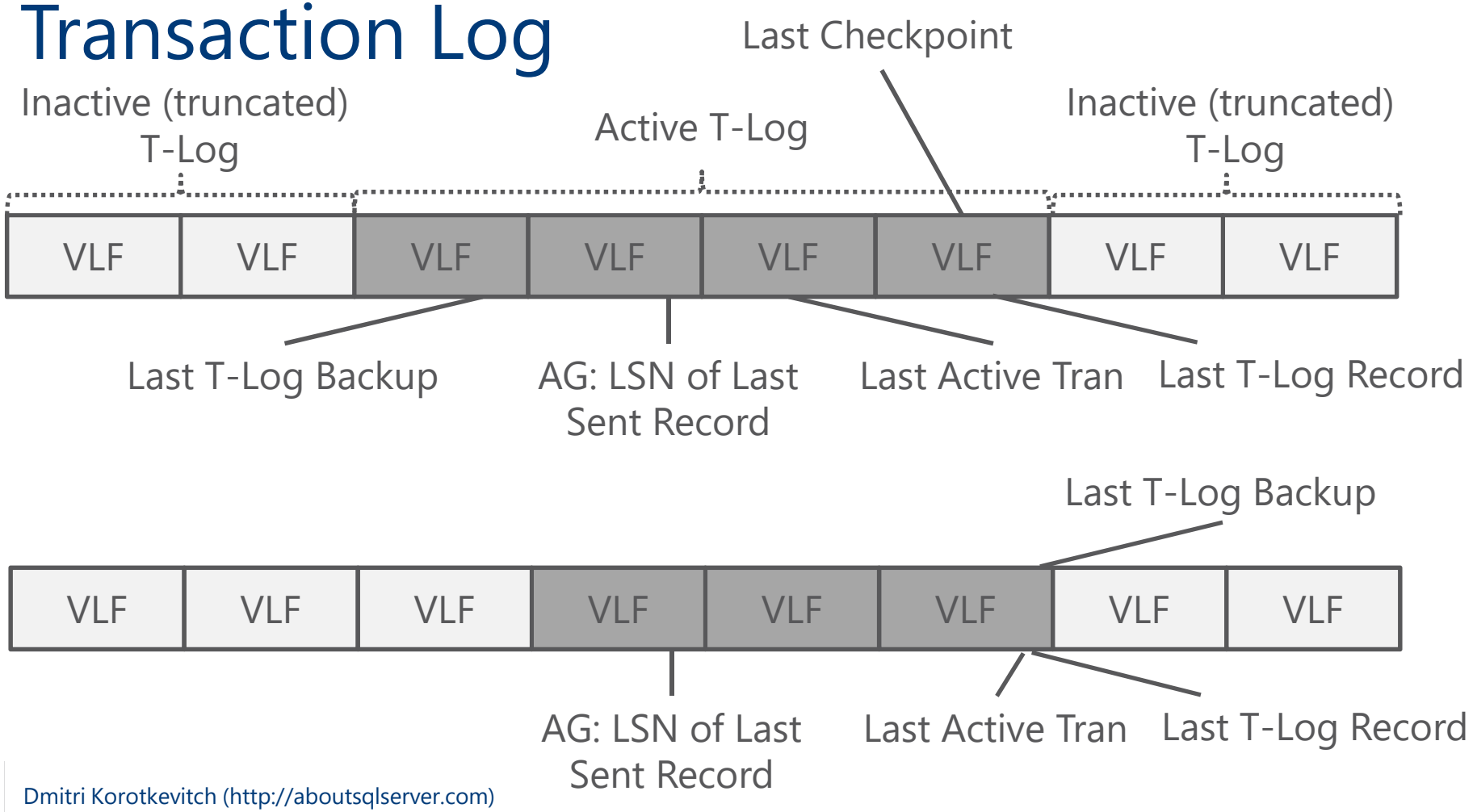
Out-of-scope:

- Availability Groups setup
- Failover troubleshooting

AlwaysOn Availability Groups



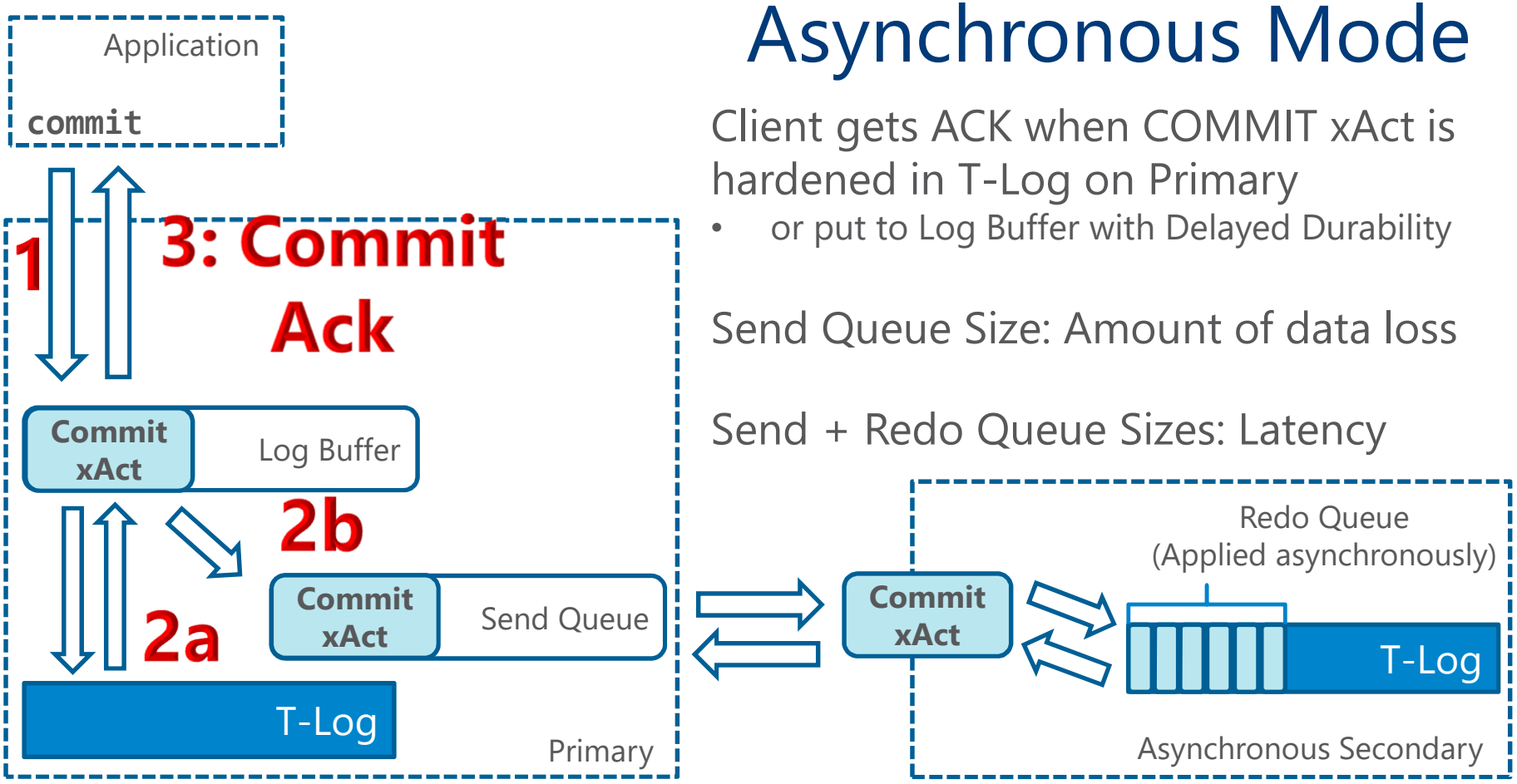
Transaction Log



Availability Groups and T-Log Truncation

Demo

Asynchronous Mode



Client gets ACK when COMMIT xAct is hardened in T-Log on Primary

- or put to Log Buffer with Delayed Durability

Send Queue Size: Amount of data loss

Send + Redo Queue Sizes: Latency

Availability Groups: Asynchronous Mode

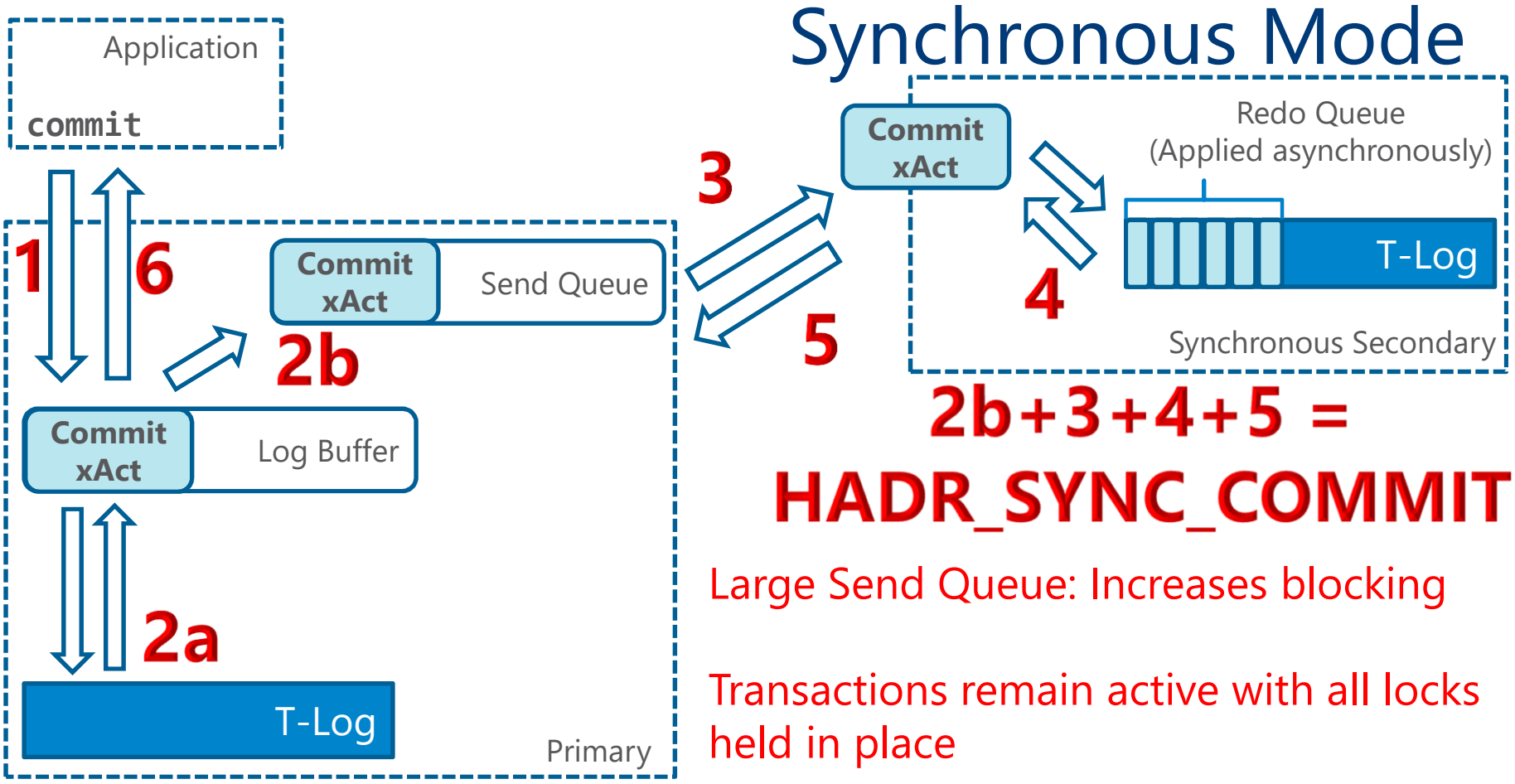
Demo

Readable Secondaries: Best Practices

Do not use Readable Secondaries for business-critical processes

Do not spread read/write transactions across servers

Synchronous Mode



2b+3+4+5 =

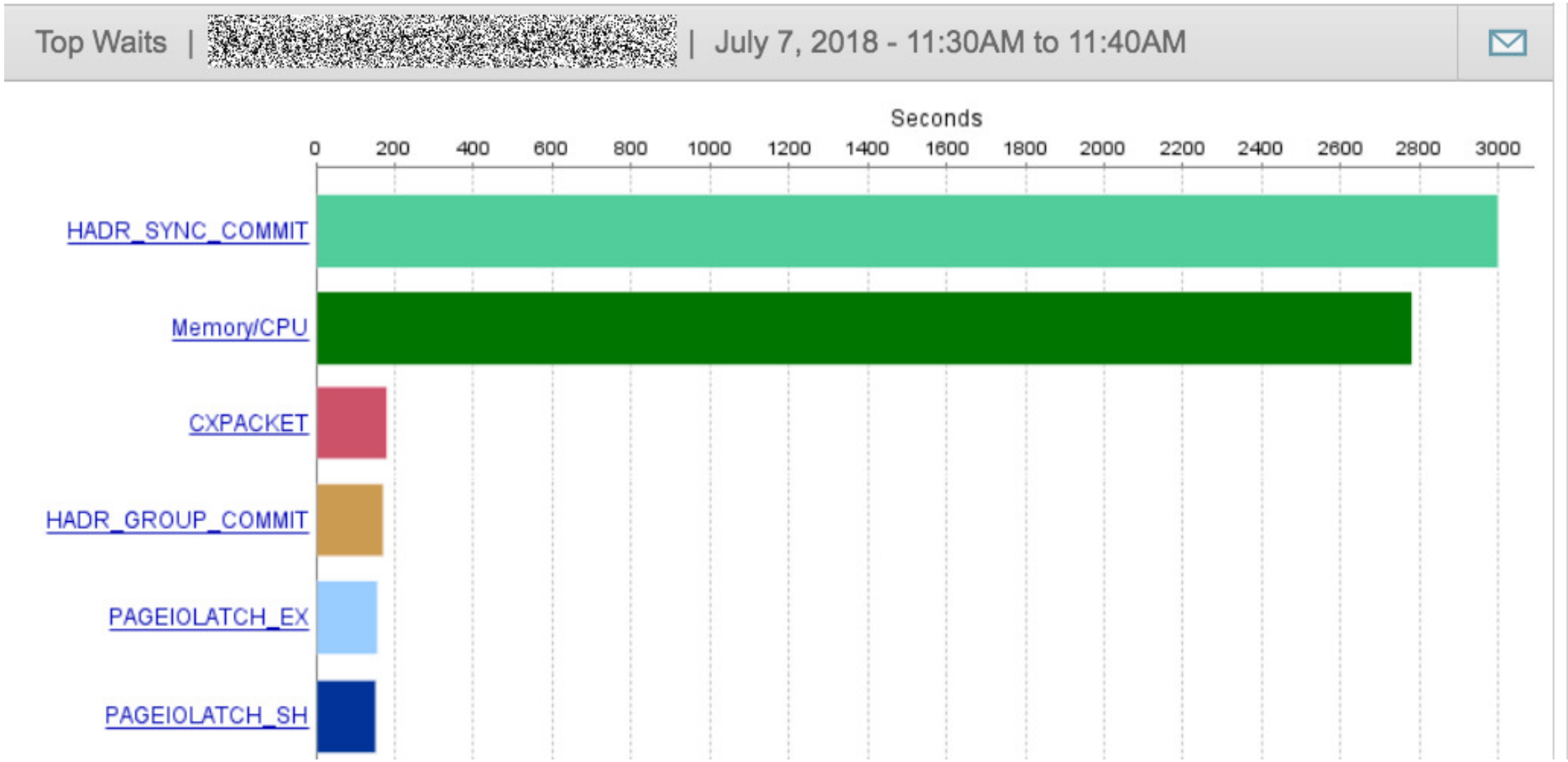
HADR_SYNC_COMMIT

Large Send Queue: Increases blocking

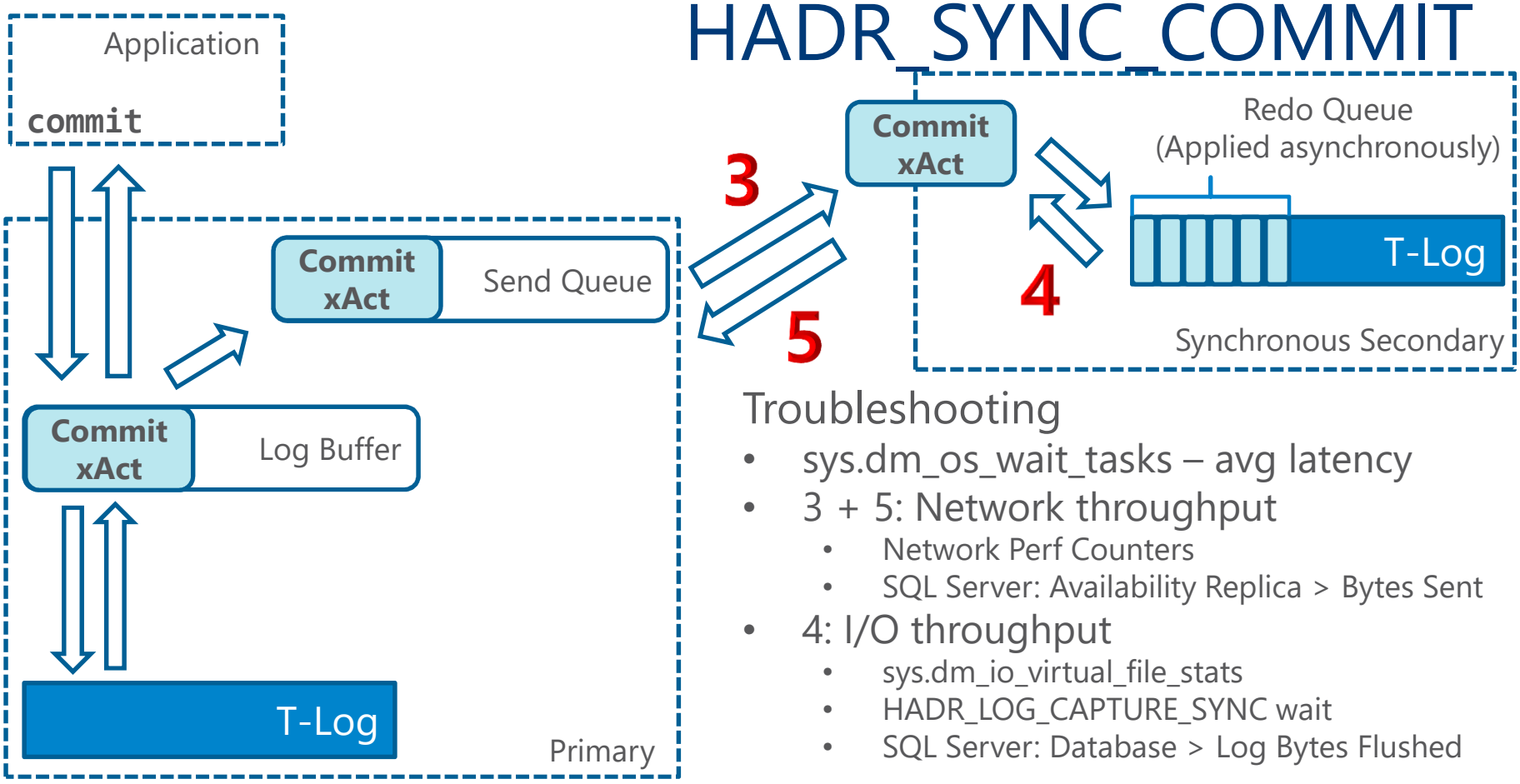
Transactions remain active with all locks held in place

Availability Groups: Synchronous Mode

Demo


Top Waits |
| July 7, 2018 - 11:30AM to 11:40AM
Seconds
0
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
2400
2600
2800
3000
HADR_SYNC_COMMIT
Memory/CPU
CXPACKET
HADR_GROUP_COMMIT
PAGEIOLATCH_EX
PAGEIOLATCH_SH

HADR_SYNC_COMMIT



Troubleshooting

- sys.dm_os_wait_tasks – avg latency
- 3 + 5: Network throughput
  - Network Perf Counters
  - SQL Server: Availability Replica > Bytes Sent
- 4: I/O throughput
  - sys.dm_io_virtual_file_stats
  - HADR_LOG_CAPTURE_SYNC wait
  - SQL Server: Database > Log Bytes Flushed

Best practices

Build dedicated subnets for:

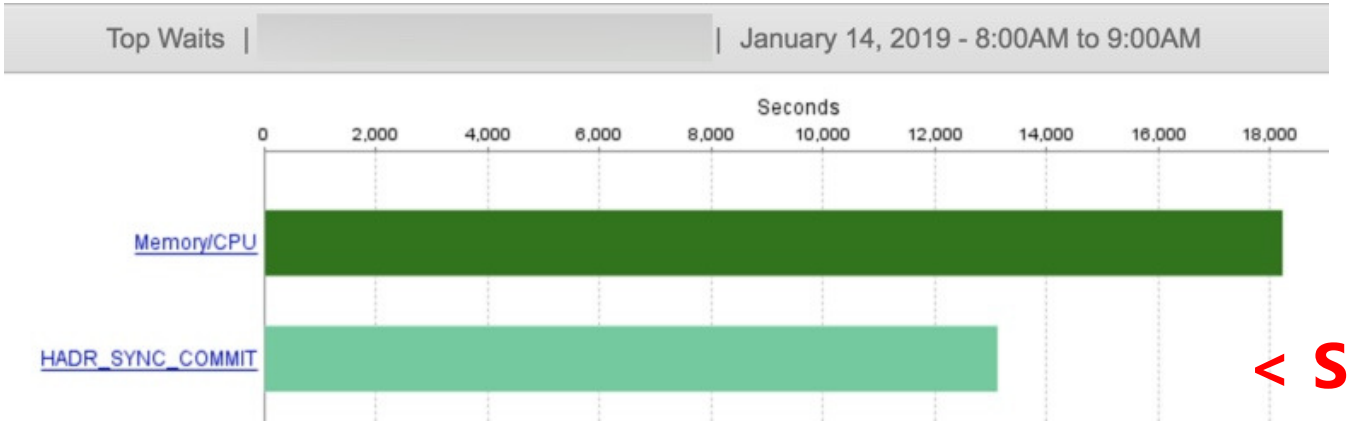
- Cluster Heartbeat
- Availability Groups transport

Evenly balance schedulers across NUMA nodes

```
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
```

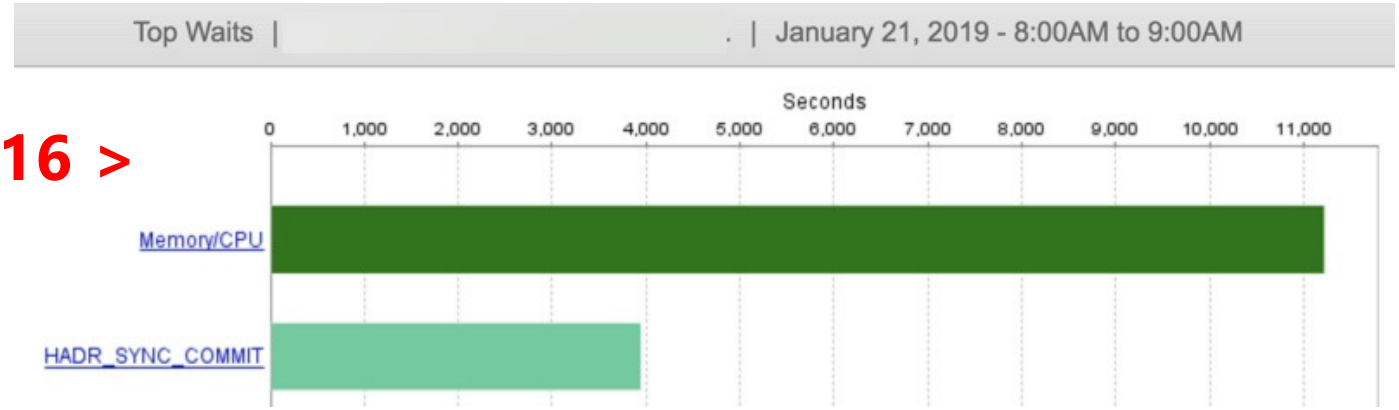
	parent_node_id	scheduler_id	status	current_tasks_count	runnable_tasks_count
1	0	0	VISIBLE ONLINE	7	0
2	0	1	VISIBLE ONLINE	6	0
3	0	2	VISIBLE ONLINE	4	0
4	0	3	VISIBLE ONLINE	6	0
5	0	4	VISIBLE ONLINE	4	0
6	0	5	VISIBLE ONLINE	4	1
7	0	6	VISIBLE ONLINE	6	0
8	0	7	VISIBLE ONLINE	7	0
9	1	8	VISIBLE ONLINE	19	0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

Upgrade to SQL Server 2016+



< SQL Server 2012

SQL Server 2016 >



Index Maintenance

Most impactful:

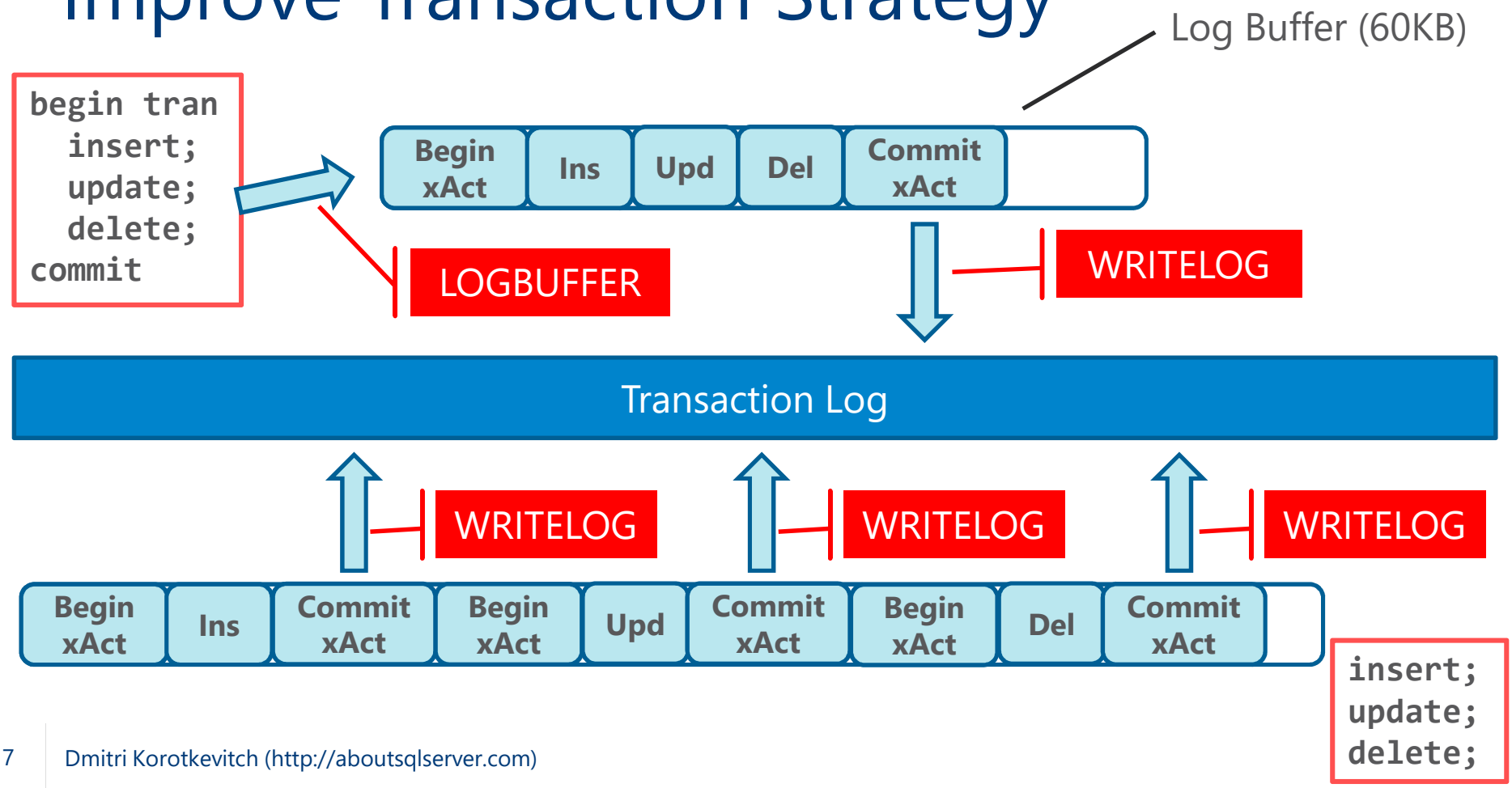
- ALTER INDEX REBUILD: When index has LOB columns
- Fast CPU and Disk subsystem: AG Transport may not be able to keep up

Mitigation

- Throttle with MAXDOP and/or Resource Governor
- Resumable index rebuild with frequent pauses/resumes (SQL Server 2017+)
- Switch to ASYNC mode when possible

Beware of any operations that generate excessive t-log activity

Improve Transaction Strategy



Autocommitted transactions overhead

Demo

Queues Monitoring

```
select
    ag.name as [
    ,ar.replica_
    ,db_name(drs
    ,case when a
    ,ars.role_de
    ,drs.synchro
    ,ars.synchro
    ,drs.log_ser
    ,drs.log_ser
    ,drs.redo_qu
    ,drs.redo_ra
from
    sys.availabi
        join sys
            ag.g
        join sys
            ar.r
        join sys
            ag.g
order by
    ag.name, drs
```

	Availability Group	Server	Database	DB Location	Replica Role	Sync State
1	PROD	N1	DB1	Local	PRIMARY	SYNCHRONIZED
2	PROD	N2	DB1	Remote	SECONDARY	SYNCHRONIZED
3	PROD	N3	DB1	Remote	SECONDARY	SYNCHRONIZING
4	PROD	N4	DB1	Remote	SECONDARY	SYNCHRONIZING
5	PROD	N1	DB2	Local	PRIMARY	SYNCHRONIZED
6	PROD	N2	DB2	Remote	SECONDARY	SYNCHRONIZED
7	PROD	N3	DB2	Remote	SECONDARY	SYNCHRONIZING
8	PROD	N4	DB2	Remote	SECONDARY	SYNCHRONIZING

Health State	Send Queue Size (KB)	Send Rate KB/Sec	Redo Queue Size (KB)	Redo Rate KB/Sec
HEALTHY	NULL	NULL	NULL	NULL
HEALTHY	0	40594	71	78495
HEALTHY	0	35420	78	44129
HEALTHY	60	13590	116	35905
HEALTHY	NULL	NULL	NULL	NULL
HEALTHY	0	45000	0	107036
HEALTHY	0	15000	0	79523
HEALTHY	0	3214	0	93234

Parallel REDO (SQL Server 2016+)

Improve performance of REDO process on Secondaries

- Does not always work as expected (especially with readable secondaries)

Monitor

- REDO queue size
- PARALLEL_REDO_* waits
- DIRTY_PAGE_TABLE_LOCK wait
- DPT_ENTRY_LOCK wait

Mitigation

- Latest SP/CU
- TF 3459 – disables parallel redo
  - Can be enabled (turned on) online
  - Requires restart to disable prior (SS 2017 CU9, SS 2016 SP1 CU10, SS2016, SP2 CU2)

Readable Secondaries

Rely on row-versioning

- Transactions on Secondaries run in SNAPSHOT isolation level
- Turns on row-versioning on Primary

Potential issues

- Increase index fragmentation
- Long running transactions on Secondaries defer *Ghost Cleanup* and *Version Store Cleanup* on Primary

Readable Secondaries Overhead

Demo

Key Points

Properly design the infrastructure

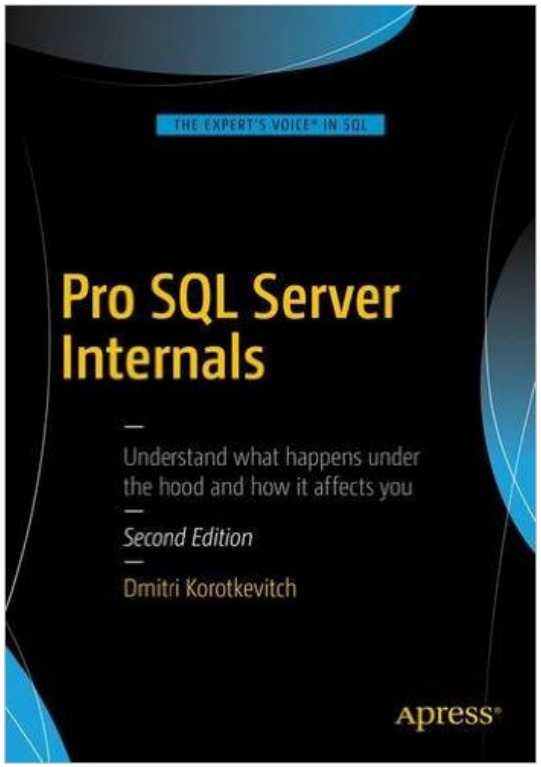
Upgrade to SQL Server 2016+

Monitor Availability Group Queues

Beware of HADR_SYNC_COMMIT wait and blocking

Monitor long-running transactions on Readable Secondaries

Q&A



Email me anytime:

dk@aboutsqlserver.com

Slides and Demos:

https://aboutsqlserver.com

Thank You