nitrosphere [WhitePaper]

DBA TACTICS FOR OPTIMIZING SQL SERVER NETWORK PERFORMANCE

KENNETH FISHER & ROBERT L. DAVIS

Networking is a very broad topic, most of which is completely transparent to SQL Server and Database Administrators (DBAs). One of the places this fact is most noticeable is in the SQL Server wait statistics (sys.dm_os_wait_stats and sys.dm_os_waiting_tasks). There are a few wait types specifically associated with networking scenarios, but the wait type you most commonly see that may indicate a network issue is ASYNC_NETWORK_IO.

The Books Online description of ASYNC_NETWORK_IO is:

Occurs on network writes when the task is blocked behind the network. Verify that the client is processing data from the server.

So when you see high ASYNC_NETWORK_IO waits it could be an indicator of a bottleneck somewhere in the network between SQL Server and the reader of the data. In most cases, the root cause can be traced to the reading client performing paged reads or simply reading the data slower than SQL Server can send it. If there is a network issue, it may show in this wait type as well.

The trouble with network bottlenecks is that DBAs typically have very little insight into the network. There are a handful of performance counters that we can track with Performance Monitor (PerfMon) or the Dynamic Management View (DMV) sys.dm_os_performance_counters that can help us determine where the problem lies by using baselining and looking for patterns where the counters skew from the norm. But they don't really help us determine where the issue lies when it is a network bottleneck.

When the network turns out to be the bottleneck, the DBA is equally limited in what they can do to try to alleviate the problem. There may be configuration changes that the DBA or network administrators can make to improve the network throughput. Sometimes the bottleneck may simply be a limitation of the infrastructure. With the increase of mobile apps, remote users, and self-service tools, DBAs are dealing with slow network connections more and more these days. Add to this the aging, slower networks that some companies are still using and expecting to get top speeds for their applications, and it becomes obvious that the network has many ways to slow things down.

When people talk about speed or performance of the network, they are referring to the amount of data that can move across the network. This is often expressed as being the size of the pipe (amount of data that can be transmitted across the network). This is the domain of the network administrator, but there are tactics that the DBA can use to improve SQL Server network performance in those cases where making changes to the network is not possible or plausible.

TDS, MTU, AND NETWORK PACKET SIZE

You hear a lot of talk about network packet size in the DBA world, and this is definitely one of the places a DBA can look at for improving SQL network performance. There is a network packet size setting in the server configurations that you can customize, and you will often see recommendations to increase this setting above the default of 4096 bytes or 4 KB (to a maximum of 32 KB or 32767 bytes). This is a common recommendation for replication performance due to the large amounts of data that get moved from server to server.

You can check the current configuration for network packet size or set it to a new value using the Server Properties dialog or the system stored procedure sp_configure. Network packet size is an advanced option and you must enable advanced options to be able to view or set it with sp_configure. You can also check the current setting, but not change it, by querying the system catalog view sys.configurations.

To check the current setting via sys.configurations:

```
-- Check network packet size
Select *
From sys.configurations
Where name = ‘network packet size (B)’;
```

```
To check the current setting via sp_configure:
```

```
-- Enable advanced options
Exec sp_configure ‘show advanced options’, 1;
Reconfigure;

-- Check network packet size
Exec sp_configure ‘network packet size’;
```

```
To change the current setting via sp_configure:
```

```
-- Enable advanced options
Exec sp_configure 'show advanced options', 1;
Reconfigure;

-- Change network packet size
Exec sp_configure 'network packet size', 8192;
Reconfigure;
```

And of course, be sure to disable advanced options when you are done by setting it to zero again followed by the RECONFIGURE command.

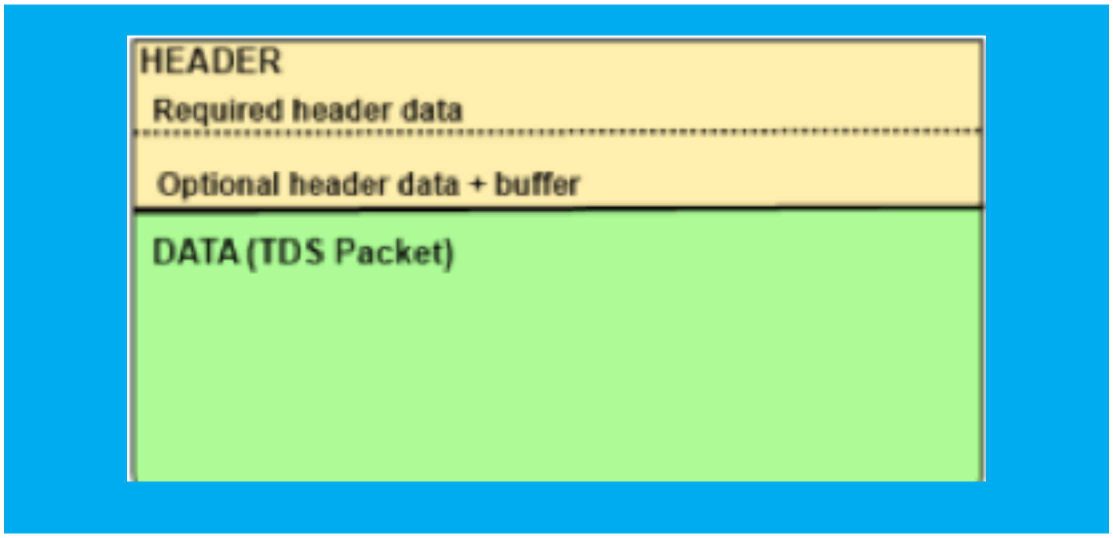
How many DBAs actually realize what they are changing when they change the network packet size? If you change the setting in SQL Server, it does not mean that anything at the network layer has changed. This setting actually changes the size of the packet for the Tabular Data Stream (TDS).

SQL Server and many other database systems utilize a protocol known as TDS that streams data out as columns and rows, or what we like to think of as a table. There is special handling for large XML data that gets streamed out as a single Unicode string broken up as needed into multiple TDS packets.

The protocol that is being used at the network layer then encapsulates the TDS packet into its own network packet. For the sake of simplicity, we are going to limit the information in this document to the most common network protocol used with SQL Server, TCP/IP. A network's packet size is completely independent of the size of the TDS packet. A common mistake that a lot of DBAs make is to increase the network packet size in SQL Server without regard to what the TCP/IP network packet size is. If the TDS packet is larger than the network packet size, the TDS packet will be broken up into multiple network packets before sending across the network and then reassembled on the other end before delivering to the client.

The TCP/IP packet consists of a header that is at least 20 bytes in size, an optional area for customized options including buffer space, and the rest is the data portion of the packet. The TDS packet makes up the data portion of the TCP/IP network packet.

Here is a simplified diagram of a TCP/IP packet:



If the SQL Server packet size is configured higher than the network packet size, the overhead of breaking down the TDS packets into multiple packets and reassembling them can be greater than any benefit you get by reducing the number of TDS packets being streamed. The TCP/IP packet size is configurable, but it generally means engaging the network administrator to first determine the capabilities of the network and how big of a packet it can support.

You can use the Windows built-in ping utility to determine the Maximum Transfer Unit (MTU) size of the network. This will tell you how big of a packet size the network is able to support between two points. This is especially useful if you are mainly concerned with maximizing network performance between two servers, like between two SQL Servers or between SQL Server and a web server or between SQL Server and a critical user.

The ping command-line utility is a tool that DBAs use frequently to test if servers are online and responding. But the utility has a couple of options that make it very useful for finding the MTU of a specific network pathway. The –l option is the size of the buffer to send in the ping packet and the –f option tells it to not fragment the packet. If the do-not-fragment option is set, and the resulting packet is larger than the maximum size packet that can be transferred (MTU), the send will fail rather than breaking into multiple packets. The header of the ping packet is 28 bytes plus the size of the buffer you specify. Since there is no data section to be sent, the header plus buffer is the entire packet. For example, if I specify a buffer size of 1000 bytes, the whole packet will be 1028 bytes.

With a little trial and error, I can find out the MTU by finding the largest packet size that does not fail. Below is a ping test to an internal server from my laptop. Through trial of several different sizes, I was able to find the point where the packet fails.

```
C:\Users\John.Q.Developer>ping 10.10.10.10 -l 1272 -f

Pinging 10.10.10.10 with 1272 bytes of data:
Reply from 10.10.10.10: bytes=1272 time=4ms TTL=120
Reply from 10.10.10.10: bytes=1272 time=4ms TTL=120
Reply from 10.10.10.10: bytes=1272 time=6ms TTL=120
Reply from 10.10.10.10: bytes=1272 time=4ms TTL=120

Ping statistics for 10.10.10.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 6ms, Average = 4ms

C:\Users\John.Q.Developer>ping 10.10.10.10 -l 1273 -f

Pinging 10.10.10.10 with 1273 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.

Ping statistics for 10.10.10.10:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

The test failed with a buffer size of 1273 but was successful for a size of 1272. Add the 28 bytes for the header and the MTU is 1300 bytes. The MTU is already smaller than the default packet size for the TDS packet (4096 bytes) so increasing the TDS packet size would not yield any benefit unless network changes were also made to increase the MTU. I do not recommend setting the network packet size lower than 4096 bytes if the MTU is lower than 4096. The MTU should only be considered for evaluating if the network packet size can be increased.

SPEEDING UP SQL PERFORMANCE IN SLOW NETWORKS

There are many reasons why the network may be a bottleneck. The demand for access to data keeps growing, budgets for network infrastructures are not expanding comparatively, there are more remote users of data than ever before, and enterprises are becoming more geographically dispersed.

A network path is only as fast as its slowest point. In many of these cases, speeding up the network itself is not possible or plausible. There are options for DBAs to boost SQL Server network performance without having to make changes to the network itself. We can compress the network traffic to reduce the number of packets that must cross the network.

One way to do this is with a WAN optimization appliance to compress the TCP/IP packets. This approach requires installing hardware at both ends of the network to compress the packets at one end and decompress at the other end. This is a very expensive solution that works great for data center to data center traffic between servers. This is not a viable solution for restricted budgets, cases where you need to improve the connection between SQL Server and individual users or when you only have physical access to one side of the connection.

There is a software solution that does not require installing new hardware in the data center and is more budget friendly than appliances. Nitrosphere offers NitroAccelerator to compress the SQL Server TDS stream to reduce the number of packets required to transmit the data.

Like a hardware solution, the tool works by compressing packets on one end of the network connection and decompressing on the other end. NitroAccelerator works by intercepting the TDS stream and compressing it before it is sent to the network layer so that more data can be sent within a single packet. In fact NitroAccelerator dynamically adjusts to the size of the TDS packet (reducing the number of configurations the DBA has to worry about by one) and transfers the maximum amount of data per MTU. Then on the receiving end, the TDS packets are decompressed after they are received.

Unlike the hardware solution, NitroAccelerator is a feasible solution for almost every endpoint, not just servers, and not just servers in data centers you control. First of all the cost point for client installations is low enough to make it affordable to compress the data from the servers to large numbers of clients. Also, simply said, NitroAccelerator is a piece of software. This means it can be used even when physical access to the data center hosting a server isn't possible. For example if a server is rented from a hosting data center then a hardware solution may not be possible. It certainly won't be when using a cloud solution and isn't really feasible with an individual's workstation either. You certainly wouldn't want to try carrying around a hardware solution with your laptop.

Another key difference between the hardware solution and the software solution is that the DBA can take full ownership of the software solution. This is a tactic specifically for DBAs without need to engage the network or system administration teams. In a modern technical environment everyone is busy and highly interdependent. Sometimes being able to provide a solution that can be handled by just one team is highly desirable. NitroAccelerator's intelligent protocol detection provides further independence by providing a solution that can dynamically adjust to changing network/port settings (at both the instance and network level) as needed.

To increase the performance gains beyond those gained by compression Nitrosphere has added HyperCache, a caching option that provides a kernel level cache that can profoundly impact latency for frequently used queries with small result sets. This can provide significant benefit to applications like web forms that make repetitive use of small lookup tables to populate the option fields.

In addition to performance benefits, NitroAccelerator can also encrypt the TDS data stream thus ensuring that data being sent to remote users is secure without having to implement something like SSL certificates within SQL Server. Normally, enforcing encryption in SQL Server is complex and requires that the client be able to support the encryption protocol as well. This makes implementing encryption simpler and agnostic to the client application being used. This is highly desirable in today's climate of almost constant data theft.

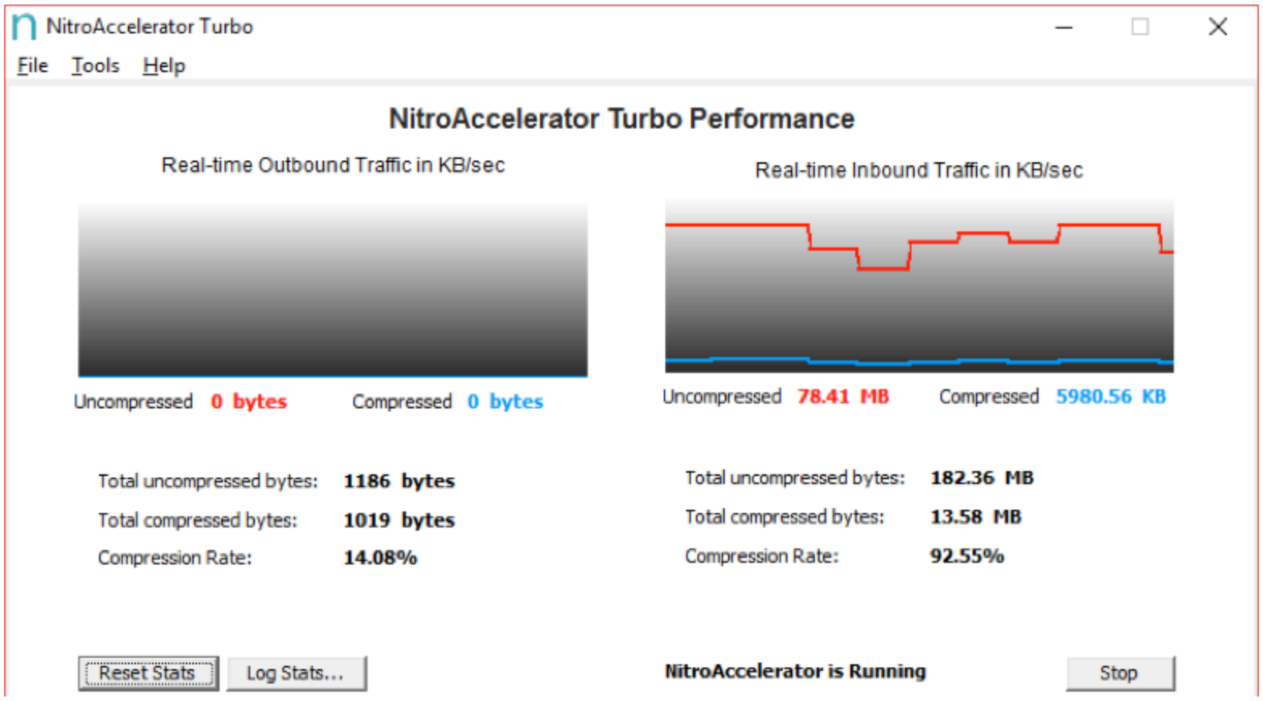
SQL NETWORK PERFORMANCE COMPARISON

NitroAccelerator includes a graphical monitor that displays the real-time compression statistics for the SQL Server network traffic and a logging option to collect data for later review. Using the graphical monitor you can immediately see the benefits of compression. In testing on real-world systems, I was able to get greater than 80% compression spiking up to mid-90s. Queries that used to take up to a minute to return the dataset were being returned in 10 to 20 seconds. Queries that had previously taken half a minute to return the data were now returning it in less than 10 seconds. The larger the dataset, the greater the benefit will be from compression. Reports that consume large amounts of data, and batch processing can see tremendous increases in performance. In particular, as BI continues to become a greater and greater presence in the world of data, the ability to transfer and consume large amounts of data quickly is vitally important.

I wanted to manufacture a test that would require a moderate to large data transfer. I started with a table with a lot of rows in the AdventureWorks2014 database. I wrote a simple view to generate a data set of 856,282 rows or just over 2 GB of data then downloaded from a database on an Azure server using BCP.

```
USE AdventureWorks2014;
GO
CREATE VIEW PersonSquared AS
SELECT p2.*, p1.FirstName+’ ‘+p1.LastName AS FullName,
      p1.Demographics demo1
FROM Person.Person AS p1
INNER JOIN Person.Person AS p2
      ON p1.FirstName = p2.FirstName;
```

bcp “PersonSquared” out c:\temp\PersonSquared.txt -S “99.999.99.99” -d Adventure-Works2014 -U UserName -P Password –n



NitroAccelerator Turbo Performance

Real-time Outbound Traffic in KB/sec

Uncompressed 0 bytes   Compressed 0 bytes

Total uncompressed bytes: 1186 bytes
Total compressed bytes: 1019 bytes
Compression Rate: 14.08%

Real-time Inbound Traffic in KB/sec

Uncompressed 78.41 MB   Compressed 5980.56 KB

Total uncompressed bytes: 182.36 MB
Total compressed bytes: 13.58 MB
Compression Rate: 92.55%

NitroAccelerator is Running

The results of this test was a compression rate a little higher than 90% which equals a reduction to approximately 145 KB of compressed data going across the network.

CONCLUSION

Meeting ever-increasing demands for access to data across networks with limited capacity is bringing network performance to the forefront, especially with users who are connecting across slower networks like wireless, mobile, or VPN and that doesn’t even include the difficulties of working with the Cloud. Delivering data as quickly as possible is becoming more critical, but a DBA’s ability to improve the actual network is severely limited. DBAs can use the tactics covered here to improve SQL Server network performance even when the network itself cannot be changed.

One tactic that DBAs now have is the option to implement NitroAccelerator from Nitrosphere to compress the SQL Server TDS stream and boost the SQL Server network throughput for the data that their users are accessing. Unlike similar hardware solutions, NitroAccelerator has a pricing model that makes it affordable to scale out to a large number of client systems. It is a solution that is easy to implement, easy to fit into today's restricted budgets, and can be completely under the control of the DBA.

ABOUT THE AUTHORS

Kenneth Fisher *is a senior database administrator at large multi-national insurance company. He is a prolific blogger and speaks at a number of local events.*
Blog: www.sqlstudies.com
Twitter: @sqlstudent144

Robert L Davis *is a database engineer at BlueMountain Capital Management. He is a speaker, trainer, writer, Microsoft Certified Master and Data Platform MVP.*
Blog: www.sqlsoldier.com
Twitter: @SQLSoldier

Take a test drive of NitroAccelerator Turbo today! Click Here