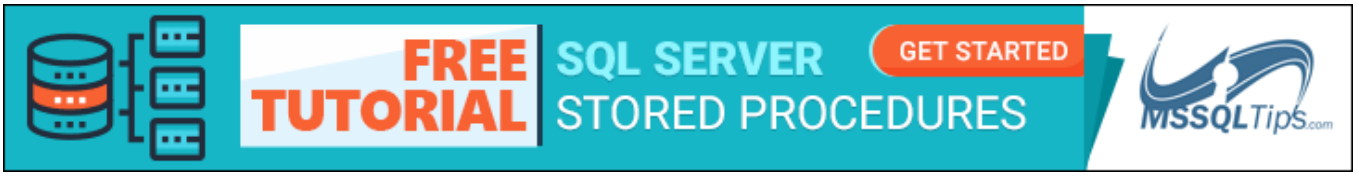
MSSQLTips Free SQL Server Conference Registration + Travel Expenses click to learn more

SQL Server Transparent Data Encryption (TDE) Performance Comparison



Free Webcast - Simplify SQL Server Administration

Problem

Every new release of SQL Server gives us new features that will hopefully improve our database system in some way. Data security has been a hot topic over the last few years and one of the new features introduced in SQL Server 2008, Transparent Data Encryption (TDE), addresses this topic. This feature gives us a way to secure our data on the backend without any front end code changes. The only concern I had with this new feature was exactly how much overhead would it produce on my server. This tip will take a look at the performance impact of implementing this feature using a few basic database operations.

Solution

For those that aren't yet familiar with TDE, you can read the following links which give an overview of TDE and also some information on how to get started implementing it on one of your own databases. In this tip will we will simply focus on the performance impact of having a TDE enabled database versus a database without TDE enabled and will not get into any details on the configuration of this feature.

- Understanding Transparent Data Encryption
- Database Encryption in SQL Server 2008 Enterprise Edition
- SQL Server 2008: Transparent data encryption feature - a quick overview
- SQL Server 2008 Transparent Data Encryption getting started
- Implementing Transparent Data Encryption in SQL Server 2008

One thing to note is all the tests below were run on a server running SQL Server 2008 with a 3GB database. Prior to executing each statement I also ran DBCC DROPCLEANBUFFERS to empty the buffer cache. This ensured that for every operation data would have to be decrypted if required and not just read directly from the buffer pool as data in the buffer pool is not encrypted.

Backup Performance

As it's an operation that we perform daily, let's first take a look at how this feature will affect our backup performance. For this test I ran scripts that performed both compressed and uncompressed full backups with TDE enabled and disabled. Each backup script was run 3 times and the averages are displayed below for each type of script execution.

	TDE Enabled				TDE not Enabled			
	CPU (ms)	READS	WRITES	DURATION (ms)	CPU (ms)	READS	WRITES	DURATION (ms)
With compression	234	319	14	57264	195	298	13	51632
No compression	203	319	14	70384	171	300	13	55714

We see from the results above that having TDE enabled results in about a 16% increase in the cpu used during the backup operation. As well, we see the total duration of the backup took anywhere from 10-20% longer with TDE enabled (depending on whether or not you were using backup compression). All in all this is a fairly small price to pay to have your database encrypted. An important thing to note with having TDE enabled on your database is that you get the extra benefit of having your backups encrypted as well thus providing extra security for all of your offline data.

SELECT, INSERT, UPDATE and DELETE Performance

Now that we know the impact TDE is going to have on our backups let's take a look at what if any impact we can anticipate for the applications that access our database. For this test I created a script with 10 SELECT, 5 INSERT, 8 UPDATE and 5 DELETE statements. Each script was run 3 times and the averages are displayed below broken up by statement type.

	TDE Enabled				TDE not Enabled			

	CPU (ms)	READS	WRITES	DURATION (ms)	CPU (ms)	READS	WRITES	DURATION (ms)
SELECT	7390	121576	18	26037103	3062	121397	1	23409175
INSERT	125	784	16	1509177	32	798	16	1254881
UPDATE	500	2181	31	1376947	108	2215	24	1080074
DELETE	502	2380	30	1271479	63	2417	29	708979

Looking at these results we can see that we get approximately a 10% increase in the duration of our SQL statements with TDE enabled which isn't too bad. The real impact of using TDE is evident here with the performance hit we take in CPU. With TDE enabled we use approximately 60% more CPU for the same workload. Keep in mind these results are measuring the performance of the entire workload. Looking at each statement type individually we can see that we take an even bigger performance hit in CPU when doing inserts, updates and deletes as compared to select statements. Depending on the behaviour of your application this may be an important thing to consider when deciding on whether or not you want to use TDE.

Index Rebuild Performance

Another operation that we as DBAs perform routinely, whether it be weekly, monthly or some other schedule, is index maintenance. For this test I created a script that rebuilt every index in my database. The script was run 3 times and the averages are displayed below.

	CPU (ms)	READS	WRITES	DURATION (ms)
TDE Enabled	14936	347345	46028	21898
TDE not Enabled	7420	347116	46190	17133

The results here are pretty consistent with what we had seen above with our other DML statements. For an index rebuild we see approximate a 50% increase in CPU usage and a 20% increase in duration with TDE enabled as compared to having it not enabled.

Summary

While this does not represent every type of operation executed on your database it gives a good indication of what you could expect the impact to be if you were to implement TDE on your database. One thing to keep in mind is that these numbers are a worst case scenario. In a real world application a lot of the data would already be in the buffer pool when a statement is run so it would not have to be decrypted and therefore the performance impact would be much less. I've read from many difference sources that in real world applications you can see anywhere from a 3-10% degradation in performance with TDE enabled. As always you should test in your own environment as your results may vary based on the nature of the data in your database and the applications accessing it.

Next Steps

- Test performance of other forms of SQL Server encryption:
  - SQL Server Column Encryption
  - SQL Server Column Level Encryption Example using Symmetric Keys
  - SQL Server Encryption Symmetric vs. Asymmetric Keys
- Test performance using third party tools to encrypt your database
- Key management:
  - SQL Server Encryption Key Management between Dev, Test and Prod
  - Managing SQL Server 2005 Master Keys for Encryption

Last Update: 2012-03-06

You're invited
to our next free
SQL Server webcast!

click to learn more

Next Tip

Click Here

About the author

Ben Snaidero has been a SQL Server and Oracle DBA for over 10 years and focuses on performance tuning.

View all my tips

Related Resources

- How to simulate transparent data encryption TDE wi...
- Implementing Transparent Data Encryption in SQL Se...
- SQL Server 2008 Transparent Data Encryption gettin...

- Configuring Transparent Data Encryption with SQL S...
- SQL Server Transparent Data Encryption (TDE) Perfo...
- Recovering a SQL Server TDE Encrypted Database Suc...
- More SQL Server DBA Tips...

Post a comment or let the author know this tip helped.

All comments are reviewed, so stay on subject or we may delete your comment. Note: your email address is not published. Required fields are marked with an asterisk (*).

*Name *Email Notify for updates

*** NOTE *** - If you want to include code from SQL Server Management Studio (SSMS) in your post, please copy the code from SSMS and paste the code into a text editor like NotePad before copying the code below to remove the SSMS formatting.

p

Send me SQL tips:

I'm not a robot

reCAPTCHA
Privacy - Terms

Save Comment

Saturday, March 09, 2013 - 3:42:45 AM - Jorge Reyes Back To Top

Hi Ben, I was wondering if you have the script that you used for the backup performance test, i really would appreciate it , I do not know much about database scripts, and I would like to perform the same test to my database, this because I am using transparent data encryption, and I would like to check the repercussion.

Thanks

Tuesday, March 06, 2012 - 8:12:37 PM - Ben Snaidero Back To Top

Hi Ruth,

All the statistics in the article were collected using SQL Profiler.

Thanks for reading

Ben

Tuesday, March 06, 2012 - 8:11:42 PM - Ben Snaidero Back To Top

Hi Desh,

I am assuming you mean solid state drives and I think although it would probably perform a little faster I think we would still see the same amount of overhead if we ran a similar test on that type of hardware given from the stats I provided most of the performance impact came from higher cpu usage. Unfortunately I don't have this type of hardware to test on.

Thanks for reading

Ben.

Tuesday, March 06, 2012 - 4:21:41 PM - Ruth Back To Top

Hi Ben,

Thanks for the article, it's very helpful information since we haven't used this feature yet.

I assum a t-sql like SP_WhoIsActive would help, just wonder what you used to collect those data in the article.

Thank you very much!

Ruth

Tuesday, March 06, 2012 - 10:05:53 AM - Desh Maharaj Back To Top

Hi. Thanks for that analysis. On the average it looks like a 20% overhead for encryption.

Also can you provide the parallel stats for similar records being processed with inline memory computing (Solid state memory). i am sure that the differetial will be minimised.

enter business email Join