**Assessing the SQL Server In-Memory OLTP Performance Benefits**

By: [Artemakis Artemiou](https://www.mssqltips.com/sqlserverauthor/183/artemakis-artemiou-/)

fROM: <https://www.mssqltips.com/sqlservertip/4201/assessing-the-sql-server-inmemory-oltp-performance-benefits/>

**Problem**

[In-Memory OLTP](https://www.mssqltips.com/sqlservertip/3121/getting-started-with-sql-server-2014-inmemory-oltp/), originally shipped with SQL Server 2014, is a powerful engine seamlessly integrated into SQL Server's Database Engine. In-Memory OLTP relies on memory-optimized data structures and performs database operations very fast. Even though In-Memory OLTP is a very promising technology, I am sure that many of us have one basic question: "Will I benefit from In-Memory OLTP and if yes, how?" The answer to this question can be very subjective because it depends on your workload.

**Solution**

[SQL Server In-Memory OLTP](https://www.mssqltips.com/sqlservertip/3121/getting-started-with-sql-server-2014-inmemory-oltp/), among other, features memory-optimized tables and natively-compiled stored procedures. Memory-optimized tables store their data in memory. They have two durability options: (i) 'SCHEMA\_ONLY' based on which the data is stored only in memory, and (ii) 'SCHEMA\_AND\_DATA' based on which the data is stored in memory but also on disk, in order to remain available after a failover or server crash. Natively-compiled stored procedures process only memory-optimized objects and are compiled into a DLL upon their creation, thus eliminating the overhead of stored procedures' interpretation during first run.

As its name implies, In-Memory OLTP is mainly intended for Online Transaction Processing (OLTP) workloads meaning that this type of workloads can benefit the most when being processed using In-Memory OLTP. Nevertheless, this does not mean that other workload types cannot also have a performance gain when they are processed using In-Memory OLTP. As stated in the beginning of this tip, the performance gain depends on your workload's type as well as on its relevant parameters.

A set of [best-suited workloads](https://msdn.microsoft.com/en-us/library/dn673538.aspx) for In-Memory OLTP has been [proposed](https://msdn.microsoft.com/en-us/library/dn673538.aspx) which summarizes workloads that have been tested against In-Memory OLTP and the performance gain was significant. These workload types are:

* High Data Insert Rate
  + Examples: Smart Metering, System Telemetry
* Read Performance and Scale
  + Example: Social Network Browsing
* Compute Heavy Data Processing
  + Examples: Manufacturing supply chains or retailers
* Low latency Execution
  + Examples: Online Gaming Platforms, Capital Markets
* Session State Management
  + Example: Managing sessions (i.e. user requests,etc.) for a heavily-visited websites

In spite of that, the majority of organizations and even individuals, prior to start using a new technology, need to be convinced that performance gain will be guaranteed when investing in that technology. So how can you get a solid indication that by using In-Memory OLTP a significant performance gain will be achieved for the target workload? There are a few approaches that can be followed towards this goal.

Approach 1: Study Published Case Studies

A first and quite simplistic approach is to read published case studies that feature similar workloads to yours. By the time the workloads and the various parameters are the same or similar, there is a good chance that you will achieve a similar performance gain when using In-Memory OLTP.

Approach 2: Analyze your Workload

A more hands-on approach is to analyze your workload and run some tests. There is a series of things to check towards this goal. These include:

1. Track the database tables that host and process thousands/millions of records (i.e. part of an ETL process) and that have a high read or write rate.
2. Analyze the tables gathered from the previous step and check what indexes and constraints they have (i.e. foreign keys, etc.).
3. Decide what are the candidate tables to be migrated to memory-optimized tables.
4. Backup the database and restore it to a non-Production environment in order to work there.
5. Run [SQL Server Memory Optimization Advisor](https://msdn.microsoft.com/en-us/library/dn284308.aspx) against the candidate tables.
6. Memory Optimization Advisor will guide you through the process of migrating the tables to memory-optimized tables and will report any issues.
7. Run queries relevant to your workload against both table types (disk-based and memory-optimized) and compare the execution times.

Approach 3 (suggested): Simulate your Workload Against In-Memory OLTP by Using Various Parameters

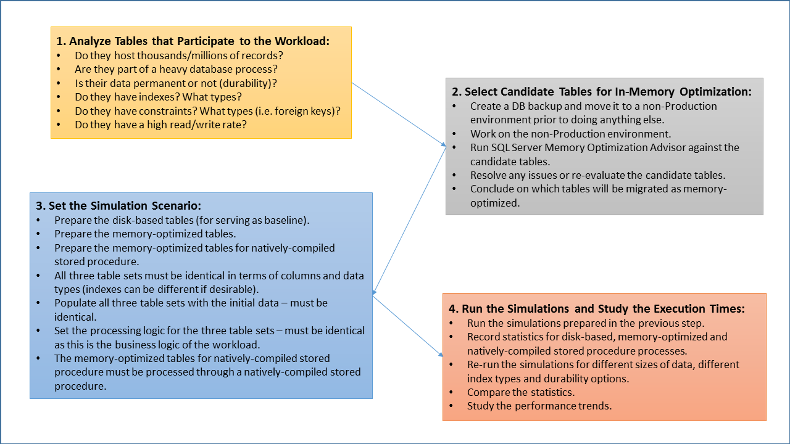
The third and suggested approach, which is actually an extension of the second, but with significant additional steps, is to simulate your workload against SQL Server In-Memory OLTP by using different table and process setups, compare the execution times and reach to conclusions about possible performance gain by observing the trend.

To this end, the steps to be performed are:

1. Track the database tables that host and process thousands/millions of records (i.e. part of an ETL process) and that have a high read or write rate.
2. Analyze the tables gathered from the previous step and check what indexes and constraints they have (i.e. foreign keys, etc.).
3. Decide what are the candidate tables to be migrated to memory-optimized tables.
4. Backup the database and restore it to a non-production environment in order to work there.
5. Run [SQL Server Memory Optimization Advisor](https://msdn.microsoft.com/en-us/library/dn284308.aspx) against the candidate tables.
6. Resolve any issues and finalize the candidate tables to be migrated as memory-optimized.
7. Prepare three table sets (tables must be identical in terms of columns and data types among each set):
   * Disk-based tables
   * Memory-optimized tables
   * Memory-optimized tables for the natively-compiled stored procedure
8. Design the three processes where each one will be executed against the corresponding table set (the last one will be implemented as a natively-compiled stored procedure).
9. Run the three processes/simulations for different 'versions' of your workload type (i.e. different numbers of records, different durability options, different indexes, etc.) and record the execution times.
10. Compare the execution times and see the trends on performance.

By using the above approach you will be able to see and examine the performance trends when using In-Memory OLTP (memory optimized tables as well as natively-compiled stored procedures) against different 'versions' of you workload in comparison to the baseline, that is the disk-based tables and their respective process. This will definitely give you a strong indication whether you are going to benefit if you use In-Memory OLTP for your workload type or not, and if yes how much (approximately), without affecting your current Production environment.

The below diagram illustrates the suggested process in four main steps:



**Next Steps**

Review the following tips and other resources:

* [Books Online: In-Memory OLTP](https://msdn.microsoft.com/en-us/library/dn133186.aspx)
* [Article: Introducing SQL Server In-Memory OLTP](https://www.simple-talk.com/sql/learn-sql-server/introducing-sql-server-in-memory-oltp/)
* [Article: Getting started with SQL Server 2014 In-Memory OLTP](https://www.mssqltips.com/sqlservertip/3121/getting-started-with-sql-server-2014-inmemory-oltp/)
* [Microsoft Whitepaper: In-Memory OLTP – Common Workload Patterns and Migration Considerations](https://msdn.microsoft.com/en-us/library/dn673538.aspx)

Last Update: 3/11/2016