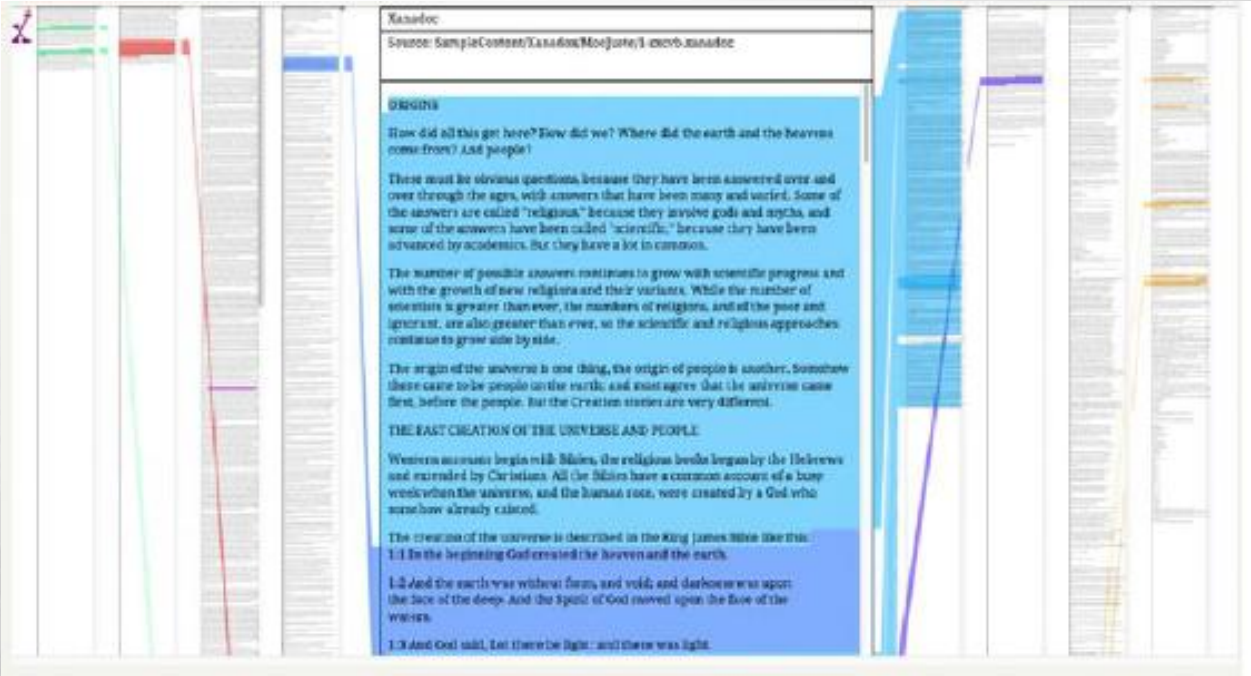
Novick Software

SQL Server 2014
In-Memory Tables *(Hekaton)*

Microsoft Most Valuable Professional

DID YOU EVER WORK ON A SOFTWARE PROJECT THAT WAS LATE?

Xanadu, The World's Most Delayed Software, Is Finally Released After 54 Years In The Making!



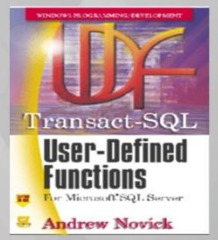
Novick Software

Andy Novick

Novick Software

- SQL Server Developer/Consultant
- SQL Server MVP since 2010
- Author of 2 books on SQL Server
- anovick@novickSoftware.com
- www.NovickSoftware.com

Microsoft® Most Valuable Professional



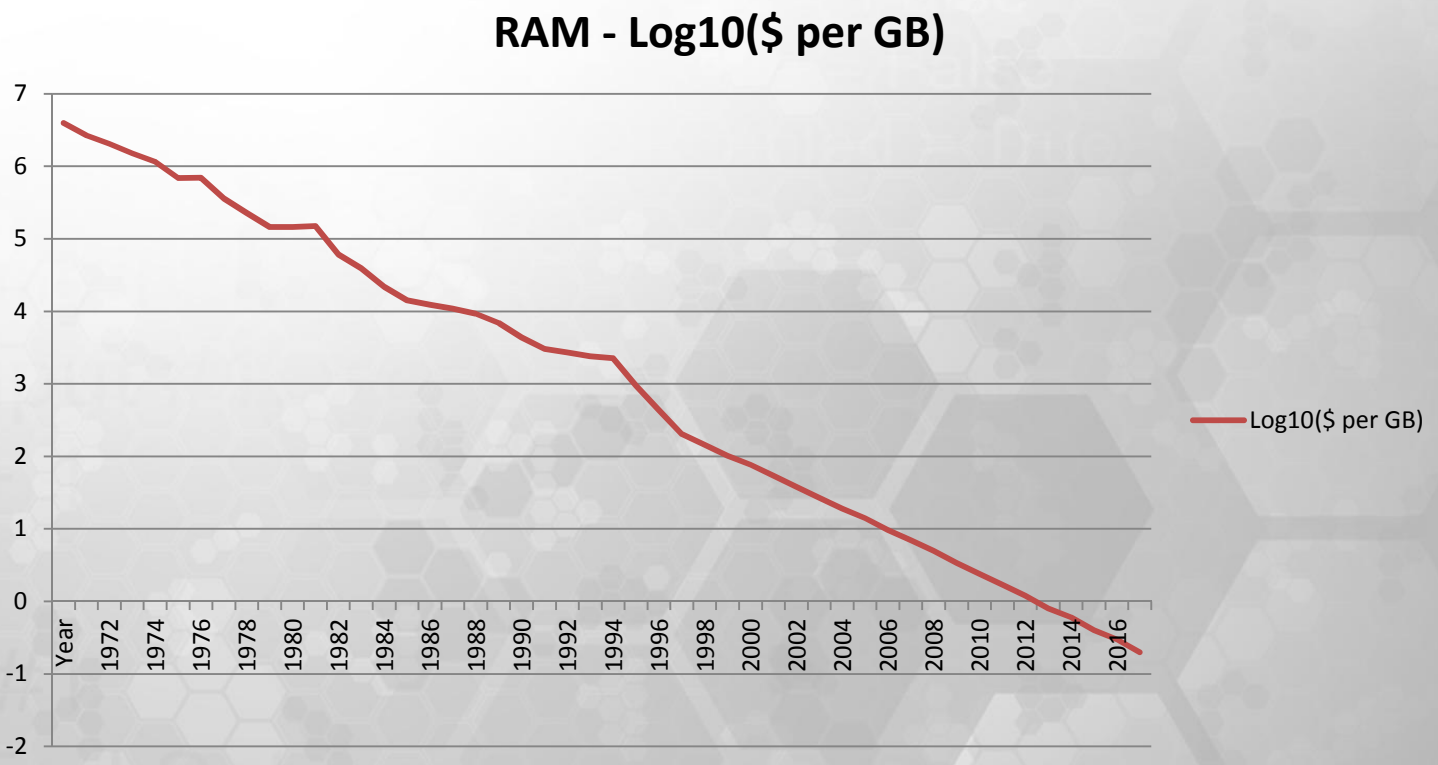
Objectives of this presentation

- Overview of In-Memory Technology
- Tools to get started trying them out

Databases with In-Memory Capability

Source: InformationWeek

RAM Prices Over the Last 40 Years



Novick Software

Why In-Memory?

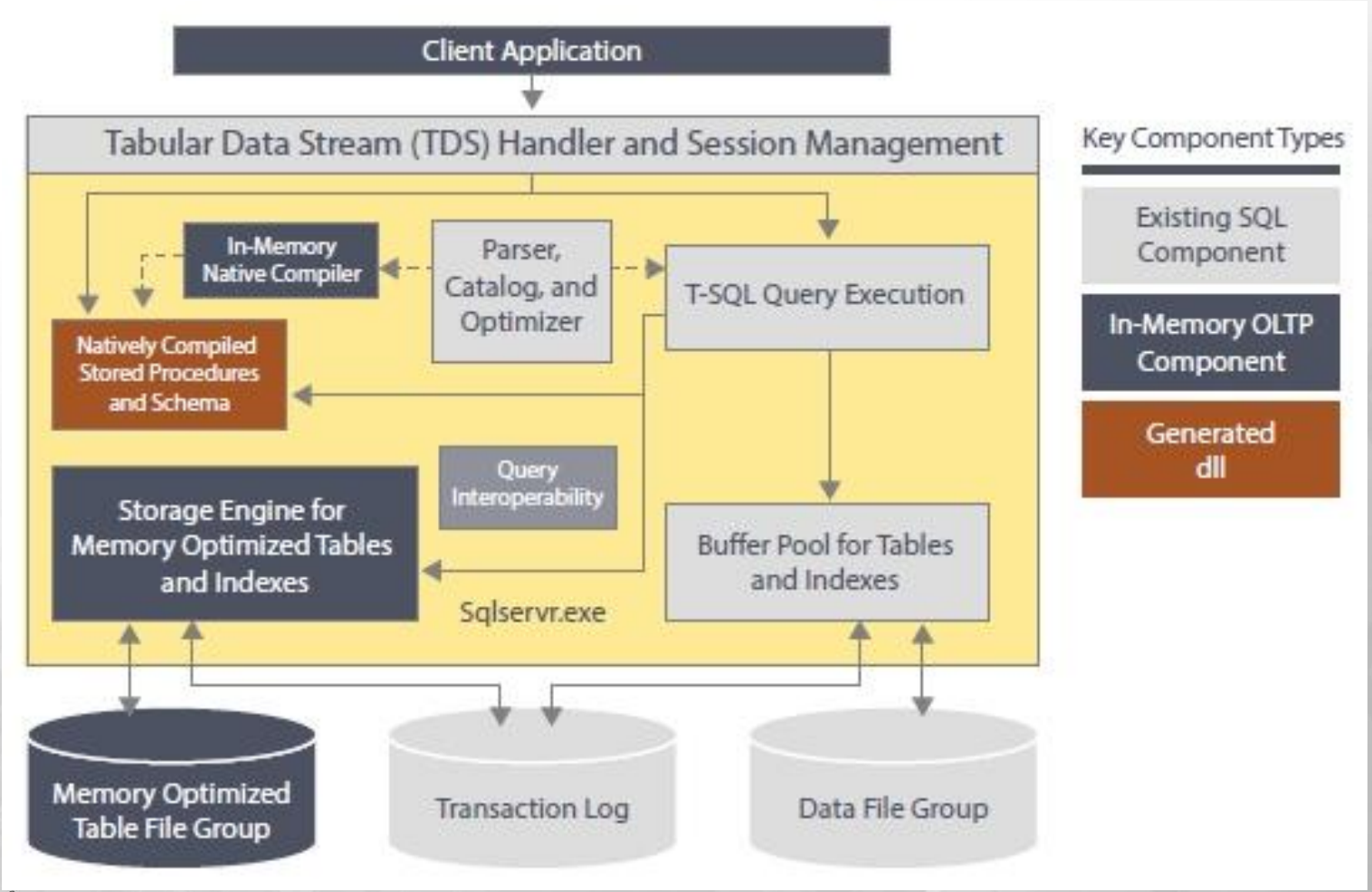
Fast OLTP

What's a Heketon?

- SQL Server In-Memory Technology
- 100 Times Faster!
  - They didn't get there ☹
    - But impressively close ☺

What goes into Hekaton

- In-Memory Tables
  - Tables with the ACID Properties
- Natively Compiled T-SQL Stored Procedures
  - Compiled to machine code including query plans


Client Application
Tabular Data Stream (TDS) Handler and Session Management
In-Memory Native Compiler
Parser, Catalog, and Optimizer
T-SQL Query Execution
Natively Compiled Stored Procedures and Schema
Query Interoperability
Storage Engine for Memory Optimized Tables and Indexes
Buffer Pool for Tables and Indexes
Sqlservr.exe
Memory Optimized Table File Group
Transaction Log
Data File Group
Key Component Types
Existing SQL Component
In-Memory OLTP Component
Generated dll

In-Memory Tables: Requirements

- 64 Bit System
- Enterprise Edition
- Processor with cmpxchg16b instruction
- Windows Server 2008 R2 SP2
  or Windows Server 2012 R2

IN-MEMORY TABLES

In-Memory Tables: 3 things you must know

- They're Tables
- They're Tables
- They're Tables

In-Memory Tables - What Makes the Them Fast?

- No locks, latches, spinlocks, etc.
- Data is written sequentially, only!
- No indexes are stored or logged. They're in memory only.
- Non-durable tables. No I/O
- Checkpoints are background sequential I/O
- Compatible with Delayed Durability

In-Memory Tables – CREATE TABLE

```
CREATE TABLE Product_inmem(
    ProdID int IDENTITY(1,1) NOT NULL,
    Name nvarchar(50) COLLATE Latin1_General_100_BIN2 NOT NULL,
    Color nvarchar(15) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    ModifiedDate datetime2(7) NOT NULL
            CONSTRAINT DF_Product_ModifiedDate DEFAULT (sysdatetime()),

    CONSTRAINT IMPK_Product_ProdID PRIMARY KEY NONCLUSTERED HASH
                        (ProdID) WITH ( BUCKET_COUNT = 1048576),

    INDEX IX_Name NONCLUSTERED (Name ASC),

 )WITH ( MEMORY_OPTIMIZED = ON , DURABILITY = SCHEMA_AND_DATA )
```

In-Mem Tables - Indexes

- Must have one index
- Up to 8 indexes per table
- Inherently covering
  - NO INCLUDE needed
- Types
  - HASH – good for equality predicates
  - RANGE – good for index seeks, ordering, etc.

In-Mem Tables - Logging

- Uses the traditional log
  - The same one for disk based tables
- Only logical changes are logged.
- All 3 recovery models are supported

In-Mem Tables - Durability

- It's optional
  - Data and Schema
  - Schema Only
- Can be delayed
- Implemented with 2 types of 2 files
  - Data files
  - Delta files

In-Mem Tables Adding the FileGroup

```
ALTER DATABASE AdventureWorks2014
  ADD FILEGROUP AdventureWorks2014_mod
        CONTAINS MEMORY_OPTIMIZED_DATA

ALTER DATABASE AdventurWorks2014
  ADD FILE (name=‘AdventureWorks2014_mod',
            filename=‘c:\db\inmem\AdventureWorks2014_mod')
  TO FILEGROUP AdventureWorks2014_mod
```

In-Mem Tables - Recovery

- Recovery at the speed of I/O
- This is a function of database size!

In-Mem Tables - Merge Process

- Processes the changes and produces a new starting data file
- sys.sp_xtp_merge_checkpoint_files
  @database_name
  , @transaction_lower_bound
  , @transaction_upper_bound
- DBA job

In-Mem Tables – Limitations (1)

- No LOB – Varchar(MAX), NVARCHAR(MAX)
  - Text, ntext, image, varbinary(max)
- No computed columns
- No CHECK, Unique or Foreign Key constraints
- No Triggers
- No Replication
- No Collations other than Bin2! ('A' != 'a')

In-Mem Tables – Limitations (2)

- Only Code Page 1252
- Update of Primary Key Columns
- No Alter Table
- Create or Drop Index

In-Mem Tables - Memory Consumption

- Memory Size = Table Size + SUM(Index Size)

- TABLE Size = Row Size * Row Count
- Row Size = SUM (Column Size) + Header
- Header = 24 + 8 * Index Count

- Hash Index size = Bucket_Count * 8 bytes

- The multiply it all by 2

In-Mem Tables - Can They Crash SQL Server?

- YES
- Integrated with Resource Governor

```
CREATE RESOURCE POOL Pool_AdventureWorksMemory
            WITH ( MAX_MEMORY_PERCENT = 80);

ALTER RESOURCE GOVERNOR RECONFIGURE;
```

In-mem Tables: Table Variables

- Created using a table type (CREATE TYPE)

```
WITH ( MEMORY_OPTIMIZED = ON )
```

- Requires at least one index
- Restrictions on data types, etc. remain
- The table is in the user database not tempdb
- No parallel plans

In-Mem Tables: Table Variable Example

```
CREATE TYPE Sales.SalesOrderDetailType_inmem AS TABLE(
                    OrderQty smallint NOT NULL,
                    ProductID int NOT NULL,
                    SpecialOfferID int NOT NULL,
INDEX IX_ProdID NONCLUSTERED HASH (ProductID) WITH (BUCKET_COUNT = 8),
INDEX IX_SpclOfferID NONCLUSTERED HASH (SpecialOfferID)
          WITH ( BUCKET_COUNT = 8)
)   WITH ( MEMORY_OPTIMIZED = ON )
GO
declare @tab Sales.SalesOrderDetailType_inmem
insert into @tab values (1, 1, 1), (2,2,2), (3,3,3)
select * from @tab
```

In-Mem Tables: Table Variable Advantages

- Not transactional
- Can be used in Cross-Database queries
- Can be used as Table-Valued Parameters to stored procedures

In-Memory Tables: Interop

- Interpreted T-SQL can use both In-Mem and On-Disk tables
- Some Restrictions
  - Not target of MERGE
- No Parallel Plans

Not Supported – Interpreted TSQL

- TRUNCATE TABLE
- MERGE (memory-optimized table as target)
- Dynamic and keyset cursors (these automatically degrade to static).
- Access from CLR modules, using the context connection.
- Referencing a memory-optimized table from an indexed view.
- Cross-database queries
- Cross-database transactions
- Linked servers

Interop vs Compiled Stored Procedure

Interop

- Access both memory and disk tables
- Most of T-SQL
- Best for
  - Ad hoc queries
  - Reporting Queries
  - Migration

Memory with Compiled Sproc

- Only Memory tables
- Limited T-SQL
- Best for
  - OLTP
  - Optimized performance

Interop vs Compiled Stored Procedure

Interop

- Use memory and disk tables
- Most of T-SQL

- Best for
  - Ad hoc queries
  - Reporting Queries
  - Migration

In-Mem /Compiled Sproc

- Only Memory tables
- Limited T-SQL

- Best for
  - OLTP
  - Optimized performance

DEMO – IN-MEMORY TABLES

Concurrency: Optimistic Locking

- There are no traditional:
  - Locks
  - Latches
  - Spin-locks
- Multi-version Optimistic Locking

- You MUST code for failure.

Isolation Levels

- Repeatable Read
  - Reads same row version a commit time as earlier
- Serializable
  - As if there were no concurrent transactions
  - All actions happen at a single commit time
- Snapshot
  - Reads are consistent as of the start of the TXN
  - Writes are always consistent

Concurrency: How

- Global Transaction Timestamp
- Multiple Row versions can live in memory
- Every row has a
  - BeginT(imestamp)
  - EndT(imestamp)

Concurrency: Issues it solves

- Dirty Reads
  - Read data from uncommitted transactions
- Non-Repeatable Reads
  - Multiple reads in a TRAN get different rows
- Phantom Reads
  - New rows read in second read in a TRAN

Concurrency: Isolation Level

Isolation Level	Dirty Reads	Non-Repeatable Reads	Phantom Reads	Rows As-of	Read-Write Write-Write
Read Uncommitted	Yes	Yes	Yes	Now	N/A
Read Committed	No	Yes	Yes	Committed/ Now	Blocks N/A
Repeatable Read	No	No	Yes	First Read	Blocks/No-Block Fail at COMMIT
Serializable	No	No	No	First Read	Blocks/No-Block Fails at COMMIT
Snapshot	No	No	No	Begin TRAN	as-of BEGIN TRAN Fails at COMMIT

COMPILED T-SQL PROCEDURES

Compiled Stored Procedures

- T-SQL stored procedure
- Compiled to C then Machine Code
  - Not Dot Net
- Include the Query Plans
- Limited Surface Area

Compiled Stored Procedures

- Native Compilation
- SCHEMABINDING
- EXECUTE AS OWNER

- BEGIN Atomic – Begin a Tran or Savepoint
  - Isolation Level Must be declared
  - LANGUAGE – Fixed at compile time

Compiled Proc - Surface Area Limitations

- Memory optimized tables only
- INNER JOIN supported
- OUTER JOIN not supported

- Also no: MERGE, OUTPUT, OR, ……

DEMO – COMPILED SPROC

What's not faster

- Chatty applications may not see any gain.
- Logging is still done (except for indexes)
  - So Low Log Latency is still important
- DML on many rows

Hardware Required

- Memory
  - All the memory you need for disk based tables, plus
  - 2 x Amount of In-Memory tables
  - Resource Governor can prevent excess memory use
- Disk
  - Log – Must be fast (SSD, PCIe based Flash)
  - Data
    - Mostly Sequential
    - I/O path speed is key

No Supported in Compiled Sprocs

- Merge
- OUTER JOINS
- Lots of other stuff
  - OUTPUT
  - CURSOR

DEMO

OLTP Demo

Stored Procedure Calls	OnDisk Seconds	InMem Seconds
1,000,000	133	14

Novick Software

SQL Server 2014: Other Great Features

- Delayed Durability
- New Cardinality Estimates
- Incremental Statistics
- Buffer Pool extended to SSD
- 640 Processors and 4 TB of RAM
- Updatable Column Store Indexes

References

- SQL Server 2014 In-Mem Sample
  http://msftdbprodsamples.codeplex.com/releases/view/125550
- Why Hekaton is Truly Revolutionary
  http://sqlblog.com/blogs/merrill_aldrich/archive/2013/10/21/why-hekaton-in-memory-oltp-truly-is-revolutionary.aspx

Downloads

- **Adventure Works 2014 Sample Databases**
  - **http://msftdbprodsamples.codeplex.com/releases/view/125550**
  - **exec sp_ChangeDBOwner sa**
- **SQL Server 2014 RTM In-Memory OLTP Sample**
  - http://msftdbprodsamples.codeplex.com/releases/view/114491
  - Change file location and be sure to use SQLCMD mode
- **RML Utilities for SQL Server (x64) CU4**
  - http://www.microsoft.com/en-us/download/details.aspx?id=4511

Coming Attractions

- Writing Faster Stored Procedures
  - Sea Coast SQL Server User Group
  - April 27th
- New England Microsoft Developers
  - Meets 1st Thursday of the Month
  - At Foliage: 20 North Ave. Burlington, MA
  - Meetup http://www.meetup.com/NE-MSFT-Devs/

Andy Novick

Thank you for coming

anovick@novicksoftware.com