Query Performance Tuning: Start to Finish

Grant Fritchey
Who?

- Product Evangelist for Red Gate Software
- Microsoft SQL Server MVP
- PASS Chapter President
- Author:
  - SQL Server Execution Plans
  - SQL Server 2008 Query Performance Tuning Distilled
Why Tune Queries?

- Most volatile aspect of a database system
- Subject to changes in data
- Affected by changes in structure
- Impacted by poor coding choices
- Victim of resource contention
Finish Line

• The ability to collect performance metrics on their servers as part of an overall query tuning methodology
• An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
• The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
• A working knowledge of DMVs that will help them identify and fix performance issues on their servers
• The ability to address common query performance problems
Query Performance Tuning – Start to Finish

GATHERING METRICS
The ability to collect performance metrics on their servers as part of an overall query tuning methodology

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

A working knowledge of DMVs that will help them identify and fix performance issues on their servers

The ability to address common query performance problems
Where to Start Tuning?

• Random
  – Pick a query?
  – Ask a user?
  – Alphabetically?

• Knowledge based
  – Baseline
  – Metrics
  – Records
Server Metrics

• Start Query Tuning at the Server
  – Hardware
  – Operating System
  – SQL Server

• Establish a Baseline
  – Now is a good time
  – Save the data
Tools for the Baseline

- Performance Monitor
- Dynamic Management Objects (DMO)
- SQL Data Collection
- Third Party Software
Where Do Problems Occur?

• Memory
• Disk I/O
• Processor
• Network

• DO NOT SIMPLY TRUST THESE NUMBERS
## Memory

<table>
<thead>
<tr>
<th>Object</th>
<th>Counter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>Pages/sec</td>
<td>Peaks &lt; 50</td>
</tr>
<tr>
<td></td>
<td>Page Faults/sec</td>
<td>Compare with baseline value for trend analysis</td>
</tr>
<tr>
<td>SQL Server: Buffer Manager</td>
<td>Buffer cache hit ratio</td>
<td>Average value &gt;= 90</td>
</tr>
<tr>
<td></td>
<td>Page Life Expectancy</td>
<td>Average value &gt; 300</td>
</tr>
<tr>
<td></td>
<td>Checkpoint Pages/Sec</td>
<td>Peak &lt; 30</td>
</tr>
<tr>
<td>SQL Server: Memory Manager</td>
<td>Memory Grants Pending</td>
<td>Peaks = 0</td>
</tr>
</tbody>
</table>
## Disk I/O

<table>
<thead>
<tr>
<th>Object</th>
<th>Counter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhysicalDisk</td>
<td>% Disk Time</td>
<td>Peaks &lt; 85%</td>
</tr>
<tr>
<td></td>
<td>Avg. Disk Queue Length*</td>
<td>Peaks &lt; 3 per disk</td>
</tr>
<tr>
<td></td>
<td>Disk Transfers/sec</td>
<td>Maximum value &lt; 100 per disk</td>
</tr>
<tr>
<td></td>
<td>Avg. Disk sec/Read</td>
<td>Compare to baseline</td>
</tr>
<tr>
<td></td>
<td>Avg. Disk sec/Write</td>
<td>Compare to baseline</td>
</tr>
</tbody>
</table>

* Meaningless on a SAN
## Processor

<table>
<thead>
<tr>
<th>Object</th>
<th>Counter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>% Processor Time</td>
<td>Peaks &lt; 80%</td>
</tr>
<tr>
<td>System</td>
<td>Context Switches/sec</td>
<td>Peaks &lt; 1,000</td>
</tr>
<tr>
<td></td>
<td>Processor Queue Length</td>
<td>Peaks &lt; 3</td>
</tr>
<tr>
<td>SQL Statistics</td>
<td>Batch Requests/sec</td>
<td>Compare to Baseline</td>
</tr>
<tr>
<td></td>
<td>SQL Compilations/sec</td>
<td>Peaks &gt; 100</td>
</tr>
<tr>
<td></td>
<td>SQL Recompilations/sec</td>
<td></td>
</tr>
</tbody>
</table>
## Network

<table>
<thead>
<tr>
<th>Object</th>
<th>Counter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Interface</td>
<td>Bytes Total/sec</td>
<td>Peaks &lt; 50% of NIC capacity</td>
</tr>
<tr>
<td>Network Segment</td>
<td>% Net Utilization</td>
<td>Peaks &lt; 80% of network bandwidth</td>
</tr>
</tbody>
</table>
Performance Monitor

• Positives:
  – Absolute Best “Do It Yourself” Option
  – Powerful
  – Accurate
  – Easy to Use
  – Ubiquitous

• Negatives
  – A lot of work to set up, maintain & clean data
  – Very little direction
Dynamic Management Objects

• Positives
  – Easy to use
  – In a language you know
  – Data is ready to query
  – Ubiquitous

• Negatives
  – Incomplete
  – No direction
SQL Data Collection

• Positives
  – Immediate results
  – Pre-generated reports
  – Easy to implement

• Negatives
  – Enterprise Only
  – 2008 Only
  – Limited Data Set
Third Party Tools

• Positives
  – Lots of Direction
  – Ready to consume reports
  – Immediate returns

• Negatives
  – Costly
  – May not collect what you need or want
Wait Stats & Queues

- sys.dm_os_wait_stats
- Sys.dm_exec_requests
- Sys.dm_waiting_tasks
Query Metrics

• This is where you live
• Too much information
• Save the data, just not in its original form

• DO NOT USE PROFILER ON PRODUCTION SYSTEMS
The Server Side Trace

• Profiler to Generate the Script
• Files work best
• Clean and Store the Data
• Profiler GUI can be used to Browse Data
  – Works with Perfmon Data
• Schedule the Start and Stop
RML Utilities

- Free
- Huge Time Savings
- Excellent Resource
- Still Need Long-Term Storage & Reporting
Query DMOs

- Sys.dm_exec_requests
- Sys.dm_exec_query_stats
- Sys.dm_exec_procedure_stats
Metrics Resources

- “SQL Server 2008 Query Performance Tuning Distilled”
- Microsoft White Paper: Performance Tuning Waits and Queues.doc [link]
- Microsoft White Paper: Troubleshooting Performance Problems in SQL Server 2008 [link]
- Brad McGehee
- Louis Davidson &
Finish Line

• The ability to collect performance metrics on their servers as part of an overall query tuning methodology
• An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
• The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
• A working knowledge of DMVs that will help them identify and fix performance issues on their servers
• The ability to address common query performance problems
Questions?

How would you...

What happens when...

Why does...

When do I...?
OPTIMIZER, STATISTICS, INDEXES & CONSTRAINTS
• The ability to collect performance metrics on their servers as part of an overall query tuning methodology
• An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
• The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
• A working knowledge of DMVs that will help them identify and fix performance issues on their servers
• The ability to address common query performance problems
Optimizer

• Simply an Amazing piece of Software
• Cost-based
• Not Perfect
• Plan on Helping the Optimizer
Relational Engine

QUERY
Relational Engine

QUERY

Relational Engine
Relational Engine

QUERY

Relational Engine

Query Parser

Syntax Check

Parse Tree
Relational Engine

QUERY

Relational Engine

Query Parser

Syntax Check

Parse Tree

Algebrizer

Resolves Objects

Query Processor Tree
Relational Engine

QUERY

Relational Engine

Query Parser
Syntax Check
Parse Tree

Algebrizer
Resolves Objects
Query Processor Tree

Optimizer
Execution Plan
Relational Engine

QUERY

Relational Engine

Query Parser

Syntax Check

Parse Tree

Algebrizer

Resolves Objects

Query Processor Tree

Optimizer

Execution Plan

Storage Engine

DATA
Observing the Optimizer

- `Sys.dm_exec_query_optimizer_info`
- Execution Plans
Statistics

- Information about the Distribution of the Data
  - Created on Index Keys
  - Created on columns
  - Created manually

- Cardinality

- By Default, Created Automatically

- By Default, Maintained Automatically

- Automatic Maintenance Is Not Enough
Investigating Statistics

DBCC SHOW_STATISTICS(\textit{table, target})

- Header

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|c|c|c|c|}
\hline
Name & Updated & Rows & Rows Sampled & Steps & Density & Average key len... & String Index & Filter Expression... & Unfiltered Rows \\
\hline
IX\textunderscore Transaction\textunderscore History\textunderscore Archive\textunderscore ProductID & Jan 19 2011 9:57PM & 89253 & 89253 & 200 & 0.04100511 & 8 & NO & NULL & 89253 \\
\hline
\end{tabular}
\end{table}

- Density

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
All density & Average Len... & Columns \\
\hline
1 & 0.002012072 & 4 ProductID \\
2 & 1.120411E-05 & 8 ProductID, TransactionID \\
\hline
\end{tabular}
\end{table}

- Histogram

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|l|l|}
\hline
RANGE_HI | KEY & RANGE_ROWS & EQ_ROWS & DISTINCT_RANGE_ROWS & AVG_RANGE_ROWS \\
\hline
1 & 1 & 0 & 6 & 0 & 1 \\
2 & 3 & 5 & 786 & 1 & 5 \\
3 & 316 & 6 & 786 & 1 & 6 \\
4 & 324 & 82 & 786 & 7 & 11.71429 \\
5 & 327 & 10 & 786 & 2 & 5 \\
6 & 328 & 0 & 619 & 0 & 1 \\
7 & 329 & 0 & 781 & 0 & 1 \\
8 & 331 & 56 & 786 & 1 & 58 \\
9 & 350 & 56 & 786 & 10 & 5.6 \\
\hline
\end{tabular}
\end{table}
Histogram

• 200 steps across the data
• An equal distribution of rows
• Leads to best possible sampling of data
  – But it’s not perfect
Updating Statistics

• sp_updatestats
  – Can resample
  – Won’t run everywhere

• UPDATE STATISTICS X
  – WITH FULLSCAN
Indexes

• Clustered Index
  – Primary Key Default (but not necessary)
  – Data is stored at the leaf level
  – Data is ordered by the key

• Non-clustered Index
  – Uses cluster key or RID of a heap
  – INCLUDE stored at leaf

• And the rest – outside the scope of this session
Constraints

- **Primary Key**
  - Cluster by default, but doesn’t have to be
  - Always an index
- **Foreign Key**
  - No indexes are created with this constraint
- **Unique Constraint**
  - This constraint is an index
What’s All This Mean?

SELECT ID FROM TableA WHERE ID = 42
What’s All This Mean?

SELECT a.ID, b.Name, c.Value
FROM TableA as a
JOIN TableB as b
  On a.ID = b.ID
JOIN TableC as c
  ON b.OtherID = c.OtherID
WHERE a.ID = 42

324 Possible Plans
Finish Line

• The ability to collect performance metrics on their servers as part of an overall query tuning methodology
• An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
• The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
• A working knowledge of DMVs that will help them identify and fix performance issues on their servers
• The ability to address common query performance problems
Questions?

How would you...?

What happens when...?

Why does...?

When do I...?
Optimizer Resources

• Dr. Dewitt’s Key Note, PASS Summit 2010

• “Inside SQL Server 2008 T-SQL Querying” Itzik Ben-Gan

• “SQL Server 2008 Internals” Kalen Delaney
Query Performance Tuning: Start to Finish

READING EXECUTION PLANS
Finish Line

- The ability to collect performance metrics on their servers as part of an overall query tuning methodology
- An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
- The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
- A working knowledge of DMVs that will help them identify and fix performance issues on their servers
- The ability to address common query performance problems
Why Execution Plans

• What will be accessed
• What indexes were used
• What kind of joins were used
• How much did all these operations cost
• Tuning
• Troubleshooting
Concepts and Architecture

• Relational Engine
  – Estimated Execution Plan
• Storage Engine
  – Actual Execution Plan
• Optimizer
  – Cost-Based
    • Just an estimate
    • Not based on your computer
• Cache
  – Most queries go to cache
Graphical Plans

- Basic Execution
- Join
- Update
- Delete
- Insert
- Sub-select
- Views
XML Plans

- Every Graphical Plan is XML
- All cached plans are XML
- Text plans show less information
Indicators

- Scans (but not always)
- Hash Joins (but not always)
- Work Tables (but not always)
- Extra Operators
- Fat Pipes
- Estimated/Actual Disparity
Execution Plan Resources

• SQL Server Execution Plans
• Microsoft Whitepapers and Web Sites
  – Statistics used by the Query Optimizer
  – Compilation and Caching
  – Showplan Security
  – Understanding Joins
  – Analyzing a Query
  – Database Engine Developer Info Center
  – Database Engine Architect Info Center
  – Forcing Query Plans
    http://download.microsoft.com/download/4/7/a/47a548b9-249e-484c-abd7-29f31282b04d/Forcing_Query_Plans.doc

• PASS Top 10 Execution Plan Web Sites
Finish Line

• The ability to collect performance metrics on their servers as part of an overall query tuning methodology
• An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
• The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
• A working knowledge of DMVs that will help them identify and fix performance issues on their servers
• The ability to address common query performance problems
Questions?

How would you...?

What happens when...?

Why does...

When do I...?
Query Performance Tuning: Start to Finish

DYNAMIC MANAGEMENT OBJECTS
Finish Line

- The ability to collect performance metrics on their servers as part of an overall query tuning methodology
- An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
- The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
- A working knowledge of DMVs that will help them identify and fix performance issues on their servers
- The ability to address common query performance problems
What is a DMOs

• Internal State Data
• Functions and Views
• Security
  – VIEW SERVER STATE
  – VIEW DATABASE STATE
Currently Running

• Answer Immediate Questions
• Not Cache Dependent
  – But is dependent on execution
• Options:
  – Sys.dm_exec_requests
  – Sys.dm_tran_active_transactions
  – Sys.dm_os_waiting_tasks
  – Sys.dm_db_index_operational_stats
Recently Running

• Answer Historical Questions
• Aggregate data, not individual calls
• Completely dependent on the cache
• Options:
  – Sys.dm_exec_query_stats
  – Sys.dm_exec_procedure_stats
Execution Plans

• Estimated Plans Only
• Parameters are unavailable
• Cache Dependent
• Options:
  – Sys.dm_exec_cached_plans
  – Sys.dm_exec_query_plan
  – Sys.dm_exec_text_query_plan
Query Text

- Batch Text
- Statement Text
  - Using Offset
- Cache Dependent
- Options:
  - Sys.dm_exec_sql_text
Finish Line

• The ability to collect performance metrics on their servers as part of an overall query tuning methodology
• An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
• The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
• A working knowledge of DMVs that will help them identify and fix performance issues on their servers
• The ability to address common query performance problems
Questions?

How would you...?

What happens when...?

Why does...

When do I...?
Query Performance Tuning: Start to Finish

COMMON PROBLEMS
Finish Line

- The ability to collect performance metrics on their servers as part of an overall query tuning methodology
- An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
- The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
- A working knowledge of DMVs that will help them identify and fix performance issues on their servers
- The ability to address common query performance problems
Common Problems

- Slow Running Query
- Key Lookup
- Parameter Sniffing
- Index Use
- Table Valued User Defined Functions
- Triggers
- Other Ways to Get Them
Slow Running Query

• **Description**
  – Most commonly used purpose for execution plans
  – Great tool for observing behavior
  – Lousy tool for comparing performance
  – Drilling down is the key

• **Indications**
  – Fat pipes
  – Scans
  – Estimated/Actual
  – Extra operations

• **Solutions**
  – Fix it
Key Lookup

• Description
  – AKA Bookmark Lookup
  – Not necessarily a problem

• Indications
  – Key Lookup Operator and a Join

• Solutions
  – Change Query
  – Change the index
  – INCLUDE
Parameter Sniffing

• **Description**
  – In general, this is a good thing
  – Depends on the data distribution and parameters used

• **Indications**
  – Intermittent poor performance
  – Disparity on estimated & actual rows
  – Different execution plans at different times

• **Solutions**
  – OPTIMIZE FOR query hint
  – Use local parameters
  – Last resort – RECOMPILE query hint
  – Last last resort – Plan Guides
Index Use

• Descriptions
  – Just because you see the index name, doesn’t mean it’s getting used properly
  – Scans are not necessarily bad
  – Seeks are not necessarily good

• Indications
  – Table Scan
  – Index Scan
  – Extra operators like table spool or sort

• Solutions
  – Create an index
  – Modify an index
  – Modify the query
Table Valued User Defined Functions

• Description
  – Yes, I see it. It says 0%. It’s a lie.
  – “One row is a tragedy; one million is a statistic. ”
    Joseph Stalin (sort of)

• Indications
  – Table Scan with a cost of 0%
  – Or Table Valued Function with a cost of 0%

• Solutions
  – When working with more than a few rows… don’t use them
Triggers

• Description
  – Triggers are not immediately visible
  – Estimated plan won’t display
  – Slow performance from query that shouldn’t be
  – Querying from optimizer…TEST TEST TEST this

• Indications
  – Second plan with the actual plan
  – No hint of it in the estimated plan

• Solutions
  – Be sure the trigger is optimized
  – Avoid where possible
Individual Statement is Slow

• Large queries or lots of queries
• The exact execution plan you want may be hard to find
• SHOWPLAN.XML - Estimated
• STATISTICS.XML - Actual
Query is Sometimes Slow

• Intermittent behavior is hard to catch

• Profiler
  – Not the gui
  – Server-side trace

• Even with a server-side trace, capturing execution plans is more expensive (primarily disk space), exercise restraint
  – Data size increase from 2k to 64k for an XML Plan per statement
  – Added overhead for storage and processing
Query Was Slow Earlier Today

- Knowing that the query is in cache is the key
- Once it’s in cache, DMV’s are your friend
  - sys.dm_exec_cached_plans
  - sys.dm_exec_query_plan
    - Really large plans won’t be stored here
  - sys.dm_exec_query_stats
  - sys.dm_exec_plan_attributes
  - sys.dm_exec_sql_text
  - sys.dm_exec_text_query_plan
    - Used for really large plans
Identifying Similar Queries

- Ad hoc systems need hugs/tuning too
  - Identifying similar queries can suggest needed indexes
  - Similar queries could be candidates for procedures
- Multiple stored procedures may have same query
- Query Hash to see similarities in query
- Query Plan Hash to see similarities in query plan
Working With Large Plans

• Really large plans are hard to read
• Large plans in text
• Large plans in XML
  – In XML, XQuery opens up the plan
• Using XML has other benefits
Finish Line

• The ability to collect performance metrics on their servers as part of an overall query tuning methodology
• An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
• The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
• A working knowledge of DMVs that will help them identify and fix performance issues on their servers
• The ability to address common query performance problems
Questions?

How would you...?

What happens when...?

Why does...?

When do I...?
Query Performance Tuning: Start to Finish

ADVANCED SOLUTIONS
Advanced Solutions

• Query Hints
• JOIN Hints
• Table Hints
• Plan Guides
• Plan Forcing
Hints

• Are you smarter than these guys?
• Have you spent more time working on SQL Server internals than these guys?
• Then why do you think you should take control of the optimizer?
Query Hints

- Unions
- Joins
- FORCE ORDER
- MAXDOP
- OPTIMIZE FOR
- ROBUST PLAN
- KEEPFIXED PLAN
Join Hints

• Loop
• Merge
• Hash
Table Hints

- NOEXPAND
- INDEX()
- FAST N
Plan Guides

- For Use When You Can’t Modify Code
- Three Kinds
  - Object
  - SQL
  - Template
- Applies Hints
Plan Forcing

• USE PLAN
• As close as you can get to direct control of the Optimizer
• Still can’t actually control it
• Absolute Last Ditch Efforts
• Limits:
  – Must be a valid plan
  – No INSERT, UPDATE, DELETE
  – No distributed or full text queries
  – Cursors can only be static or fast_forward
Finish Line

- The ability to collect performance metrics on their servers as part of an overall query tuning methodology
- The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries
- An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries
- A working knowledge of DMVs that will help them identify and fix performance issues on their servers
- The ability to address common query performance problems
Questions?

How would you...?

What happens when...?

Why does...?

When do I...?
Please Complete the Evaluation Form

Pick up your evaluation form:
• In each presentation room

Drop off your completed form
• Near the exit of each presentation room
• At the registration area

Presented by Dell
THANK YOU!

For attending this session and PASS SQLRally Orlando, Florida

Presented by Dell