Performance Tuning with Execution Plans
GOAL

• Show where, how and why performance issues appear within execution plans in order to better understand how to use execution plans to troubleshoot SQL Server query performance
Let’s Talk

Grant Fritchey

✉️ scarydba.com

✉️ grant@scarydba.com

🐦 @gfritchey
Today’s Agenda

• Introduction to Execution Plans
• Common T-SQL Code Smells
• Worked Examples
• Querying the Plan Cache
• More Worked Examples
• Parameter Sniffing
• Additional Tools
Introduction to Execution Plans
Execution Plans

- Execution plans are a representation of the processes used by the query engine to perform the query submitted to SQL Server.
Relational Engine

QUERY

Relational Engine

Query Parser
- Syntax Check
- Parse Tree

Algebrizer
- Resolves Objects
- Query Processor Tree

Optimizer

Query Processor
- Execution Plan

RESULT
Optimizer

- **Cost-Based**
  - Just an estimate
  - Not based on your computer

- **Statistics**
  - Defined in indexes and tables
  - Must be maintained to ensure a good execution plan

- **Cache**
  - Every query goes to cache (almost)
Generating a Plan

- **SQL Server Management Studio**
  - Estimated
  - Actual
- **Procedure Cache**
  - Estimated (sort of)
- **Extended Events**
  - Estimated
  - Actual
- **Trace Events (not recommended)**
  - Estimated
  - Actual
Tune the Query

Small to medium, look at the query first
Medium to large, go straight to the execution plan
Very large and insane, query the execution plan
Watch for low-hanging fruit
Fix syntax over stats
Stats over indexing
  Indexing over restructuring
  Restructuring
Read the execution plan
Understand the business needs
Where To Start?
Where To Start?
First Operator

- Plan size
- Compile time
- Memory grant
- Missing Indexes
- Optimization level
- Parameter
  - Compiled value
  - Runtime Value
- Query hash
- Reason for early termination
- ANSI settings
Right to Left or Left to Right?

• A clue: English
• Another clue: These things
Left to Right or Right to Left

• Answer: Both
• Logical processing order:
  – Represents how the optimizer sees the query
  – Reading it from Left to Right
• Physical processing order
  – Represents the flow of data
  – Follow the arrows/pipes from Right to Left
• Both are necessary to understand certain plans
What Else to Look For

- Warnings
- Most Costly Operations
- Fat Pipes
- Extra Operations
- Scans
- Estimated vs. Actual
Summary

- Execution plans are your view into the optimizer
- You can capture plans multiple ways
- You start with the first operator
- Additional things to look for include:
  - Warnings
  - Most costly operations
  - Fat pipes
  - Extra operations
  - Scans
  - Estimated vs. Actual
- Remember that these are just representations
Common T-SQL Code Smells
Code Smells

• A code smell is a piece of code that functions, but doesn’t function in the best possible way within a given set of circumstances
T-SQL Code Smells

• Functions on Predicates
• Data Conversion (Implicit & Explicit)
• Cursors
• Nested Views
• IF Logic
• Multi-Statement Table-Valued User Defined Functions
Worked Examples
Querying the Plan
SELECT TOP 10 
SUBSTRING(dest.text, (deqs.statement_start_offset / 2) + 1, 
  (CASE deqs.statement_end_offset 
    WHEN -1 THEN DATALENGTH(dest.text) 
    ELSE deqs.statement_end_offset 
    - deqs.statement_start_offset 
    END) / 2 + 1) AS querystatement, 
deqp.query_plan, 
deqs.query_hash, 
deqs.execution_count 
FROM sys.dm_exec_query_stats AS deqs 
CROSS APPLY sys.dm_exec_query_plan(deqs.plan_handle) AS deqp 
CROSS APPLY sys.dm_exec_sql_text(deqs.sql_handle) AS dest 
ORDER BY deqs.total_elapsed_time DESC;
SELECT DB_NAME(deqp.dbid),
    SUBSTRING(dest.text, (deqs.statement_start_offset / 2) + 1,
        (CASE deqs.statement_end_offset
            WHEN -1 THEN DATALENGTH(dest.text)
            ELSE deqs.statement_end_offset
            END - deqs.statement_start_offset) / 2 + 1) AS StatementText,
    deqs.statement_end_offset,
    deqs.statement_start_offset,
    deqp.query_plan,
    deqs.execution_count,
    deqs.total_elapsed_time,
    deqs.total_logical_reads,
    deqs.total_logical_writes
FROM sys.dm_exec_query_stats AS deqs
    CROSS APPLY sys.dm_exec_query_plan(deqs.plan_handle) AS deqp
    CROSS APPLY sys.dm_exec_sql_text(deqs.sql_handle) AS dest
WHERE CAST(deqp.query_plan AS NVARCHAR(MAX)) LIKE '%StatementOptmEarlyAbortReason="TimeOut"%';
Interesting Dynamic Management Objects

- Sys.dm_exec_query_plan
- sys.dm_exec_query_profiles
- Sys.dm_exec_text_query_plan
Additional Resources

- `Sp_whoisactive` – Adam Machanic
- Diagnostic Queries – Glen Berry
- Performance Tuning with SQL Server Dynamic Management Views – Louis Davidson and Tim Ford
More Worked Examples
Parameter Sniffing
Parameter Sniffing

- It’s a good thing… except when it isn’t
- Automatic
- Only works on parameters (with an exception)
- It’s all about statistics
  - Average vs. Specific
Bad Parameter Sniffing

• Differentiate from parameter sniffing
• Still about statistics
• Intermittent
• Different plans
• Focus on the compiled value
• Compare to runtime
• When it’s bad, it’s very bad
Local Variables

• Eliminate parameters
• Turn parameters into local variables
• Produces “generic” plan
Variable Sniffing

- The exception to parameters
- Same process
- Only works in a recompile situation
- Invisible killer or guardian angel
OPTIMIZE FOR <value>

- Specific and accurate
- Changes over time
- Produces “precise” plan
OPTIMIZE FOR UNKNOWN

- For when you’re not sure
- Changes over time
- Produces “generic” plan
WITH RECOMPILE

• Specific every time
• Increases overhead
• May be more costly than
Statistics

- After all, it’s all about the statistics
- Stats can age w/o updating
- You may have auto-update turned off
- Sampled updates may be inadequate
- Filtered statistics may help
Plan Guides

• Just a different way to use hints
• Produces whatever plan you define
Turn Sniffing Off

• Dangerous choice
• Last for a reason
• Very dangerous
• Turns it all off
• Everywhere
• Did I mention it’s dangerous?
Additional Tools
Supratimas

- Web based
- Free
- Easy to use
- Limited Functionality
SQL Sentry Plan Explorer

• Application
• Free and Paid Version
• Easy to Use
• Extensive Functionality
Query Store

• Azure SQL Database
• SQL Server 2016
• Guaranteed to change how you monitor and tune queries
Conclusion
Tune the Query

- Small to medium, look at the query first
- Medium to large, go straight to the execution plan
- Very large and insane, query the execution plan
- Watch for low-hanging fruit
- Fix syntax over stats
  - Stats over indexing
    - Indexing over restructuring
    - Restructuring
- Read the execution plan
- Understand the business needs
Why?
Resources

• Scarydba.com/resources
• SQL Server Execution Plans
• SQL Server Query Performance Tuning
Rate This Session Now!

Tell Us What You Thought of This Session

Rate with Mobile App:
- Select the session from the Agenda or Speakers menus
- Select the Actions tab
- Click Rate Session

Rate with Website:
Register at www.devconnections.com/logintoratesession
Go to www.devconnections.com/ratesession
Select this session from the list and rate it

Be Entered to WIN Prizes!

#ITDEVCON
Let’s Talk

Grant Fritchey

✉️ scarydba.com
✉️ grant@scarydba.com
🐦 @gfritchey