

Maximizing Existing Hardware Investment by Addressing SQL Server Performance Issues

Jeffry A. Schwartz

September 30, 2009

SQLRx® Luncheon

jeffrys@isi85.com



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Introduction

> Extending hardware life spans while satisfying customers

- Critical in today's economic environment
- Essential for analysts to be able to identify
 - Hardware areas under stress
 - Software behavior that is causing the stress
- Analysts must also understand how to resolve these issues



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Introduction

> Today's session

- Provides performance metrics and analysis techniques to expedite analyses
- Discusses high-level hardware-related and preliminary SQL Server performance analysis
- Describes information and techniques applicable to Windows 2000/2003/2008/Vista/7 and SQL Server 2000/2005/2008



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Query Side-Effects

- **Poorly designed SQL queries can cause a system to *appear* to be out of**
 - Processor
 - Memory
 - Disk
- **Because queries often perform excessive**
 - Processor work while grinding through memory-resident data buffers
 - Physical I/O when data is not memory-resident
 - Can ultimately exhaust physical memory
- **Large Sorts often create all of the above**



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Analysis Methodology

- **Use Windows Performance Monitor, a.k.a. PerfMon, to determine**
 - When problems occur, including their durations
 - Which hardware components are involved
 - Many small queries or a few large resource-intensive queries?
- **Capture SQL-specific information to isolate troublesome queries & identify their behavior**
 - SQL Trace and QueryStats DMV (2005/2008) can capture batch and stored procedure information (including specific statements)
 - Identify worst SQL statements
 - File statistics identify heavily used physical database files



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Analysis Methodology

> Use Query Plans of identified statements to determine

- Resource-intensive operators (# of rows, size of average row)
- Opportunities for new or revised indices
- **Join ordering** issues that result in bloated intermediate result sets when final result sets are small
 - Physical join order **often** ≠ actual join order

> Revise statements or database schema

- Common Table Expressions (CTE) can be used on 2005/2008 to insure proper **join ordering** (almost always)
- Build query results gradually using multiple CTEs
 - Eliminates need for explicit temporary tables
 - Sometimes must resort to temporary tables, but this is the **rare** exception
- Example: Query elapsed time reduced from **2.5 hours to 5 minutes**



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Analysis Methodology

> Generate estimated execution plan

- Test against representative, usually large, data set
- Insure that reduced resource consumption will actually occur

> Execute statement and use one of the following to capture execution information

- Mgmt Studio Statistics IO and Time options
- Trace
 - Most useful in my experience, especially when execution context is important
- QueryStats DMV (2005/2008)



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

PerfMon Metrics

> Review invaluable hardware and SQL Server metrics

- % User Time (Processor)
- % Privileged Time (Processor)
- % Interrupt Time (Processor)
- % DPC Time (Processor)
- % Idle Time (each Disk)
- Avg. Disk sec/Transfer (each Disk)
- Avg. Disk sec/Write (each Disk)
- Available Bytes (Memory)
- Page Life Expectancy (SQLServer:Buffer Manager)
- Page Reads/sec (Memory & SQLServer:Buffer Manager)



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Assessing Your System

➤ Potential problems exist if consistently...

Counter	Criterion
% Processor Time	> 70%
% Privileged Time	> 30% (Processor)
% Interrupt Time	> 20% (Processor)
% DPC Time	> 25% (Processor)
% Idle Time	< 40% for any Disk LUN and especially SQL LUNs
Avg. Disk sec/Transfer	> 0.040 seconds (40 ms)
Avg. Disk sec/Write	> 0.040 seconds (40 ms)
Available Bytes	< 500 MB (Memory)
Page Life Expectancy	< 300 seconds (SQLServer:Buffer Manager)

Windows Memory Counters

> Page Reads/sec

- Do not confuse this counter with SQLServer:Buffer Manager one
- SQL Server I/Os **not** counted **here**
- **Not** just reads **from** paging file!
- Should be low on dedicated SQL Server machine except when
 - Reading flat files into the database, e.g., bulk insert
 - Performing backups
 - Recreating full text indices

> Available Bytes (Kbytes or Mbytes)

- Should be **at least 500 MB** to allow for above activities
- Some books suggest 4 MB ok – it is **NOT** – system **stops** functioning **long** before this



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Physical I/O Measurements

- > Critical for SQL Server systems because they are often I/O constrained
- > I/O time measured **directly** by disk driver, which provides transfer times to Windows
- > I/O time = service time + **queue time** due to driver's location in I/O path
 - Disk **response** time
- > Queuing **not always** the cause of large I/O times
 - May not be possible to improve large service times because of physical or financial constraints

Database I/O Counters

- > *Page reads/sec* and *Page writes/sec* counters
- > Measures **physical** I/Os, not logical I/Os
 - SQL Trace measures logical I/Os
- > **May indicate**
 - Insufficient database memory
 - Applications improperly accessing database
 - Improper database table implementation
- > **Plot reads and writes on same graph**
 - Highlights changes in workload behavior
 - Heavy write activity may coincide with periods of poor performance, especially when RAID 5 disks involved
 - Add **Full Scans/sec** and **Forwarded Recs/sec** for better view

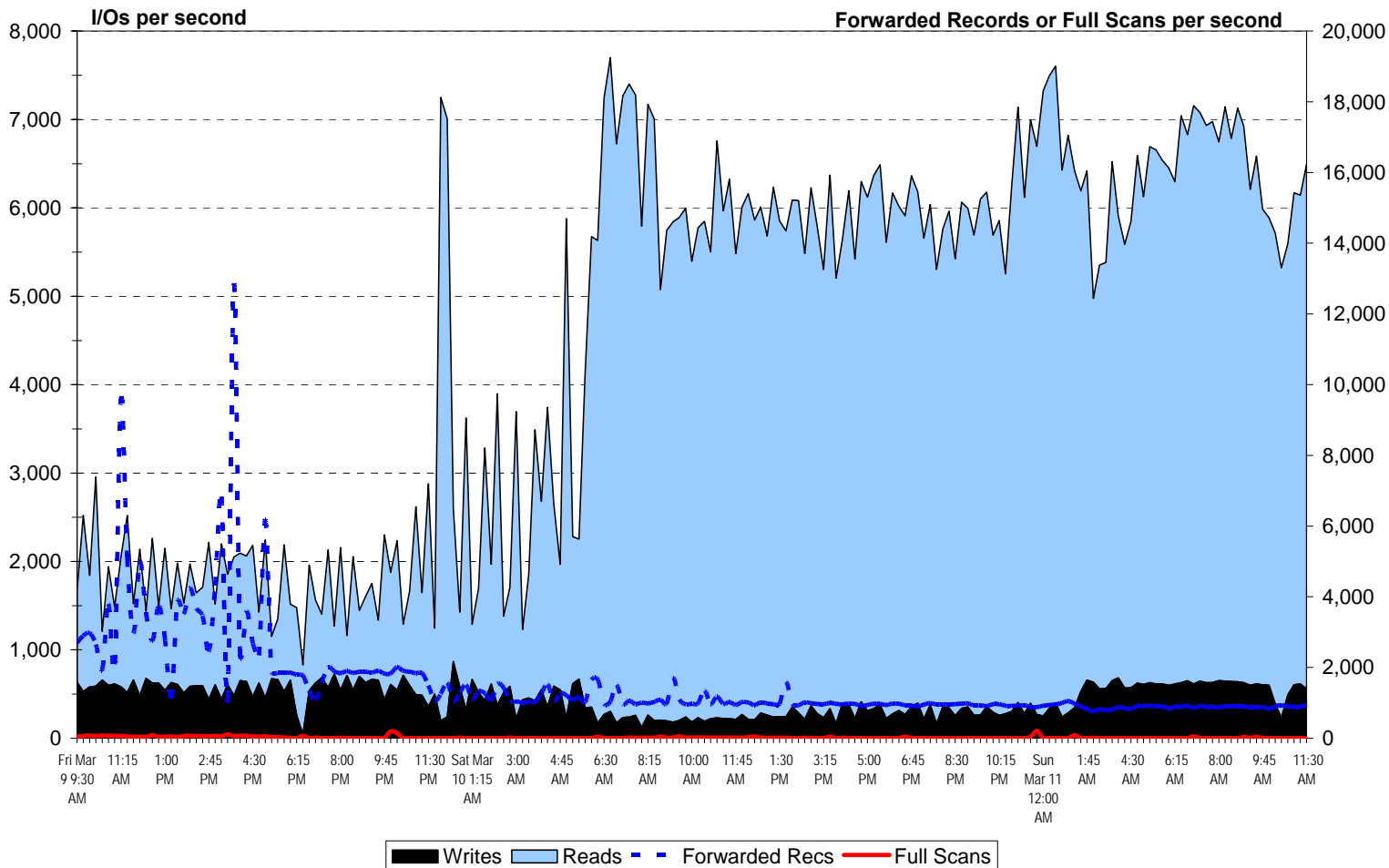


2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

I/O Activity vs. Scans and Forwarded Records Graph

SQL Server I/O, Forwarded Record, & Full Scan Activity Overview



Writes
 Reads
 Forwarded Recs
 Full Scans



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
 Database Management

SQLRx.com

Page Lookups/sec Counter

- > Measures number of times SQL Server attempted to find page in buffer pool
- > “**Logical**” read
- > Compare
 - *Page Reads/sec* with *Page Lookups/sec*
 - *Batch Requests/sec* with *Page Lookups/sec*
- > Useful for corroborating, and further quantifying, buffer cache hit ratio, especially on huge memory systems

Detecting Insufficient SQL Memory

> Use *Page Life Expectancy*

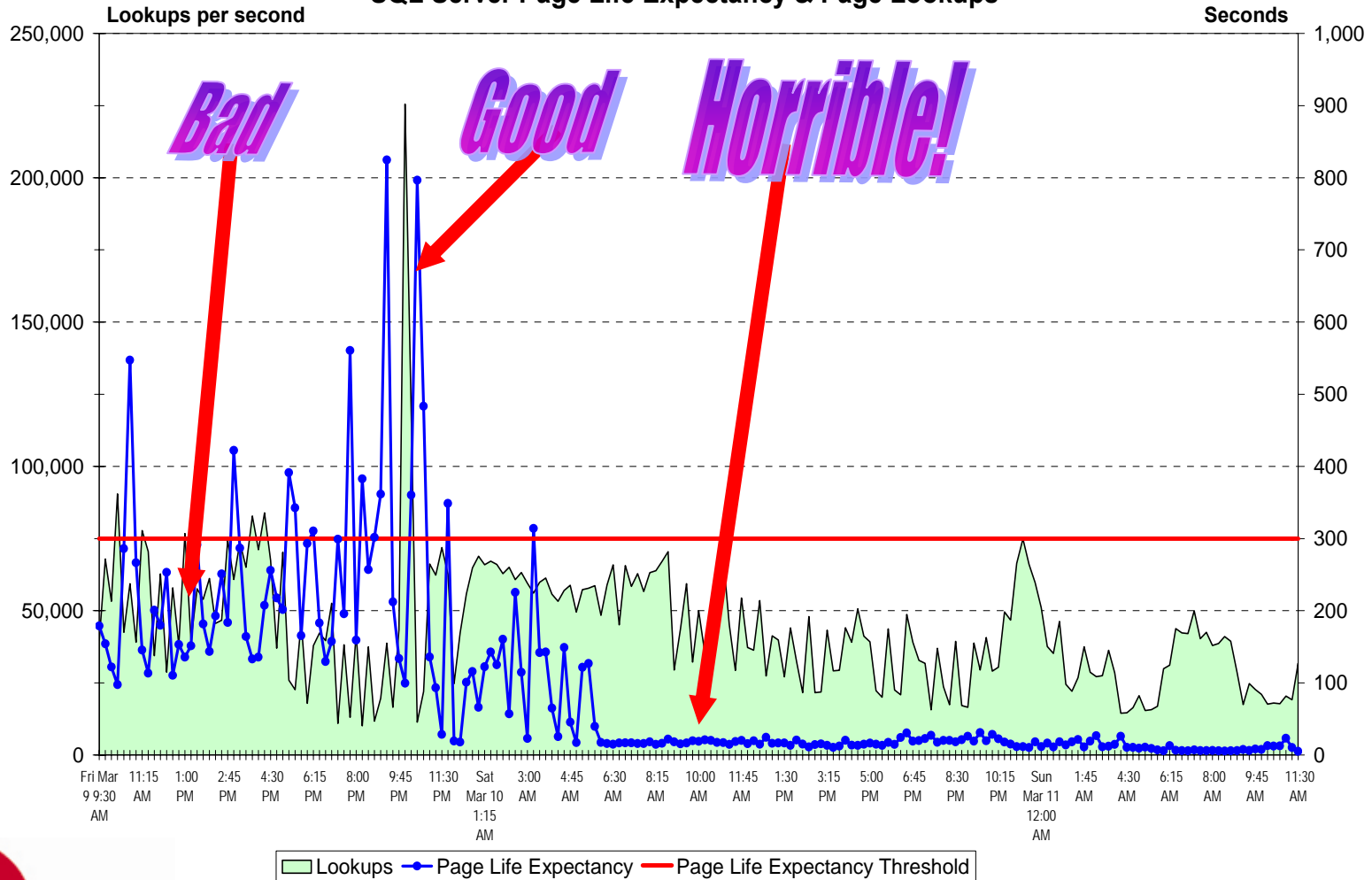
- Measures time unlocked buffers allowed to remain in buffer pool

> If *Page Life Expectancy* too low (< 300)

- Allocate more memory to SQL Server or optimize queries
- Malformed queries that read inappropriate amounts of data can cause low *Page Life Expectancy* because of data churn
 - Page reuse is **very** low
- Use SQL Trace Sort Warning records, especially hash warnings, to determine offending queries

Page Life Expectancy vs. Page Lookups Graph

SQL Server Page Life Expectancy & Page Lookups



Lookups Page Life Expectancy Page Life Expectancy Threshold



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

SQL Statement Handling

> Batch

- Group of SQL statements
- Possibly hundreds or thousands of lines
- Must be parsed and compiled into optimized execution plan



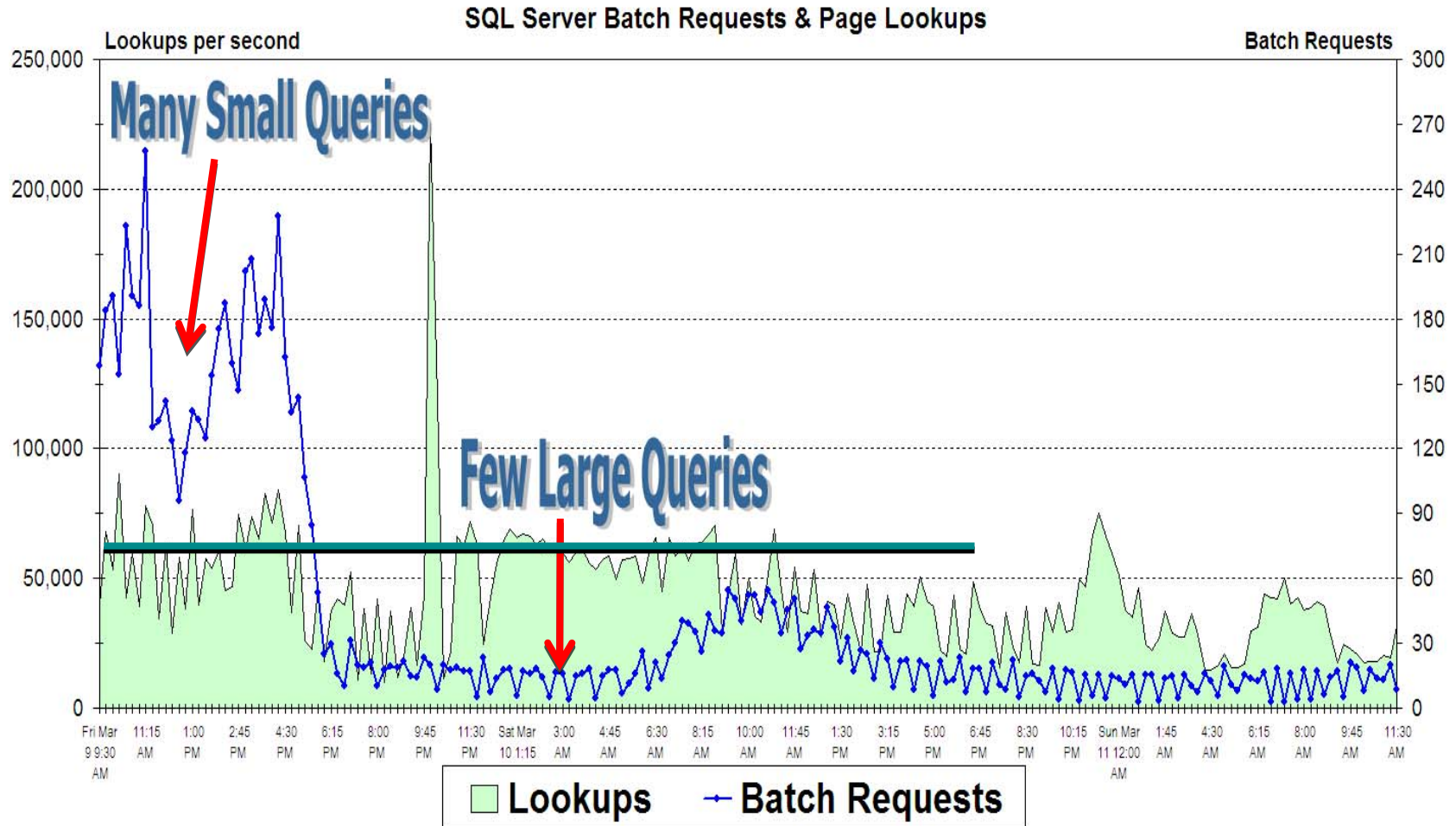
2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Batch Requests/sec

- > Number of select, insert, and delete **statements**
- > **Each** of these statements triggers a batch event, which increments the counter
- > **Note: Also includes** each of these statement types executed **within** a stored procedure

Batch Requests vs. Page Lookups Graph



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Dynamic Management Views and Functions

- > “Designed to give you a window into what's going on inside SQL Server”
- > Two types
 - **DMV** - Pure view, i.e., no parameters required
 - **DMF** - Table-valued function, i.e., parameters required
 - Parameters usually specify database, table, index, partition, etc.
- > Provide significant amount of information regarding
 - System
 - Databases
 - Internal workings of SQL Server
 - Performance



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Dynamic Management Views and Functions

> Some were possible with complex queries on 2000

- When even possible
 - **Very** clumsy to use
 - Required temp tables
 - Required **significant** understanding of the underlying tables

> Most DMVs and DMFs simple to use

- Some still require joins with other DMVs and DMFs

> Some may lessen need for using SQL Trace



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Dynamic Management Views and Functions

- > **Contain values since last SQL Server instance restart, unless manually reset**
 - Perfect for periodic or intermittent sampling
- > **Many DMVs can be reset manually using command similar to**
DBCC SQLPERF ('sys.dm_os_wait_stats', CLEAR)

Dynamic Management Views and Functions – General Guidelines

- > Most performance DMVs begin with **sys.dm_**
- > Must reconcile numeric IDs with static views that contain textual names
- > Values accumulated from last SQL Server instance restart
 - Sample rates can range from once per minute to a few times per day
 - Must calculate differences between individual sample records
 - Be sure to diff records with same database ID, object ID, and index ID



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Dynamic Management Views and Functions – General Usage Scenario

> Method 1

- Capture before monitored activities begin and store results into a SQL table or a spreadsheet
- Execute workload (whether natural production or test)
- Capture again and compare value differences
- Load into spreadsheets for further analysis, if necessary

> Method 2

- Capture every <n> minutes and store results in a flat file
- After workload and capture process complete, load data into SQL tables
- Use SQL to compare incremental value differences and store results
- Extract interval data to spreadsheets for further analysis



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Critical General Views

- > **Required** for converting numeric DMV and DMF identifiers into understandable text
- > Many were available under 2000 with slightly different names
 - 2000 sysdatabases -> 2005/2008 sys.databases
 - See “**Mapping System Tables to System Views (Transact-SQL)**” topic in BOL for further info
- > **sys.databases**
 - Lists all databases and their IDs so proper associations can be made

Critical General Views

> sys.partitions

- **Only** way to decode HOBT IDs returned by lock-specific information, e.g. blocked process records and `sys.dm_os_waiting_tasks`

> sys.configurations

- Provides information regarding OS and SQL Server configurations

> sys.dm_os_sys_info

- Provides information regarding system (can be sampled once or repeatedly)
- Useful for determining whether hyperthreading is active and accessible memory

Database-Specific Views

> Query for each database separately

> **sys.objects**

- Lists all database objects such as tables, views, stored procedures, etc.

> **sys.indexes**

- Lists all indices and their associated table IDs
- Does **not** provide row counts as sysindexes does

> **sys.filegroups**

- Lists all file groups and their IDs

> **sys.database_files**

- Lists all physical database files and their IDs



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

sys.dm_io_virtual_file_stats DMF

- > Returns I/O statistics for data and log files, one row per file
- > Ideal for determining
 - Most active files
 - Files waited for most frequently and longest
 - Read/Write ratios
 - Average I/O sizes by file
- > Use NULL parameters to obtain information for all databases and files

```
select * from sys.dm_io_virtual_file_stats (NULL, NULL)
```

- > Equivalent to SQL 2000

```
select * from ::fn_virtualfilestats(-1,-1)
```



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

sys.dm_io_virtual_file_stats Columns

- > ID of database
- > ID of file
- > Number of **milliseconds** since SQL Server instance started (useful for detecting restarts)
- > Number of reads and writes issued against this file
- > Total number of bytes read from and written to this file
- > Total time, in **milliseconds**, users waited for I/O completions overall, as well as reads and writes specifically
- > Number of **disk** bytes used by this file



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

File Statistics – Data Example

Drv	File	Db	Bytes	Total I/O Wait (Secs)	Total I/O Wait (Hrs)
F	XData	X	280,647,901,184	43,188	12
C	tempdev	tempdb	146,123,325,440	298	0
C	Y_Data	Y	52,310,769,664	7,088	2
C	Z_Data	Z	33,409,597,440	1,106	0
F	XLog	X	8,639,794,176	6	0
C	templog	tempdb	4,980,503,040	1	0
C	MSDBData	msdb	3,049,644,032	585	0
C	D_Data	D	997,187,584	301	0
C	master	master	71,811,072	106	0
F	XData	X_Training	59,179,008	35	0
F	XData	X_Test	52,903,936	50	0
C	E_Data	E	50,675,712	28	0

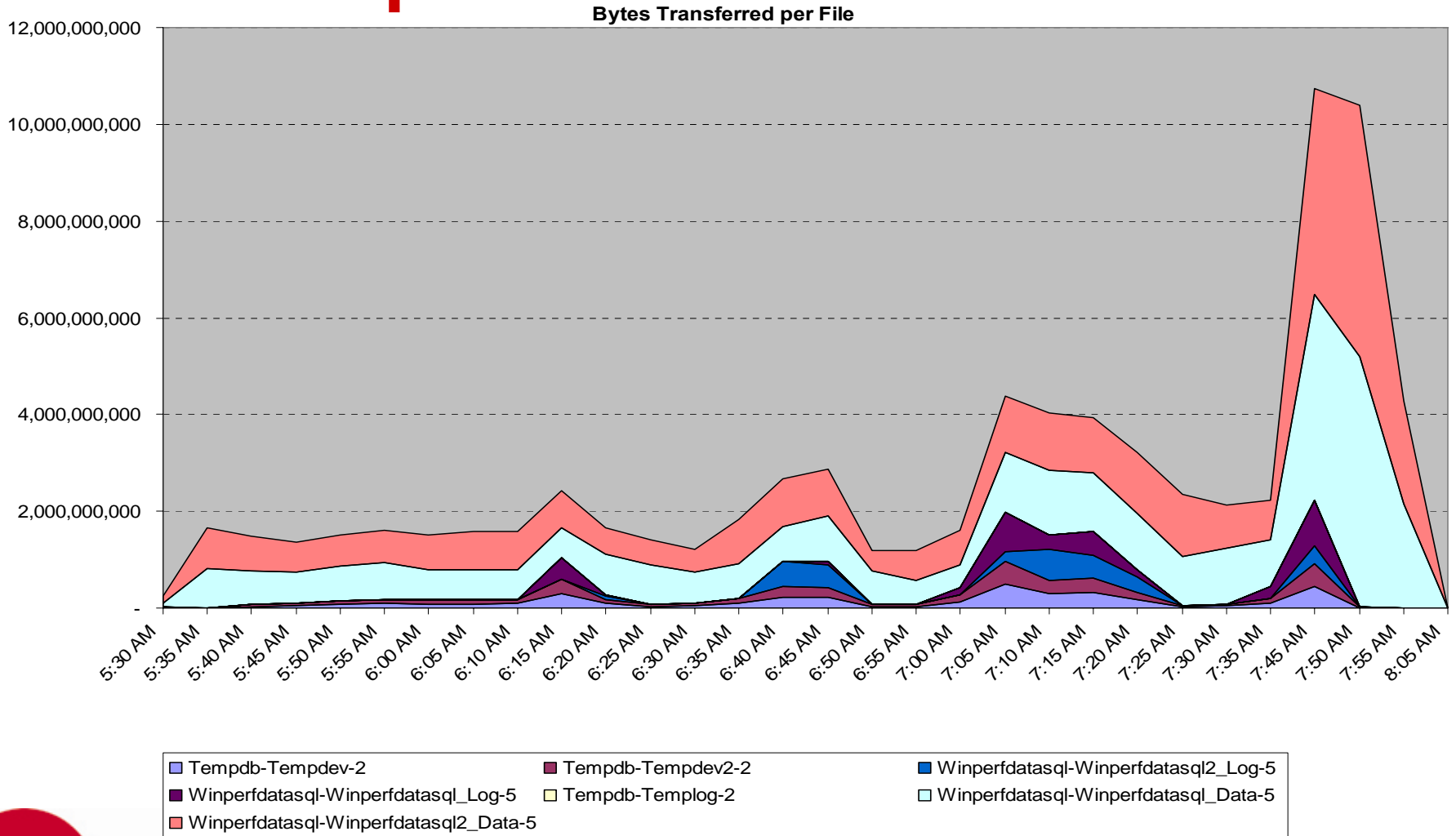
File Statistics – Data Example

Drv	File	Db	Reads	Writes	I/Os
F	XData	X	11,637,520	268,627	11,906,147
C	tempdev	tempdb	207,391	2,210,178	2,417,569
C	Y_Data	Y	831,723	981	832,704
C	Z_Data	Z	513,069	4,705	517,774
F	XLog	X	47,729	321,552	369,281
C	templog	tempdb	89	83,454	83,543
C	MSDBData	msdb	131,655	2,058	133,713
C	D_Data	D	46,408	7,880	54,288
C	master	master	8,429	3	8,432
F	XData	X_Training	7,060	0	7,060
F	XData	X_Test	6,248	0	6,248
C	E_Data	E	3,324	100	3,424

File Statistics – Data Example

Drv	File	Db	Bytes Read	Bytes Written
F	XData	X	277,438,513,152	3,209,388,032
C	tempdev	tempdb	9,222,414,336	136,900,911,104
C	Y_Data	Y	52,298,825,728	11,943,936
C	Z_Data	Z	33,369,636,864	39,960,576
F	XLog	X	5,678,792,704	2,961,001,472
C	templog	tempdb	4,951,040	4,975,552,000
C	MSDBData	msdb	3,032,276,992	17,367,040
C	D_Data	D	930,734,080	66,453,504
C	master	master	71,786,496	24,576
F	XData	X_Training	59,179,008	0
F	XData	X_Test	52,903,936	0
C	E_Data	E	49,840,128	835,584

sys.dm_io_virtual_file_stats Time Series Graph



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

SQL Profiler

- > **Bad reputation as a resource hog and performance killer need not be deserved**
- > **Excessive resource consumption caused by**
 - Requesting constantly occurring events
 - Examples: locks, scans, cache hit
 - Requesting too many events
 - Not using duration filter
 - Updating GUI on monitored machine
- > **Good for getting started, but trace definition T-SQL script is better**
- > **Trace captured locally regardless unless unusual arrangements have been made**



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Event Classes

> Categories of entities that are monitored

- Examples:
 - Cursors
 - Database
 - Locks
 - Errors and Warnings
 - Scans
 - Stored Procedures
 - Transactions
 - T-SQL

> **Lock:Cancel** and **Deadlock-related** events are **ONLY** lock-related events that should be monitored



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Starting Events

> Usually unnecessary

- Start time can be calculated from completed event records by subtracting duration from the ending time
- Stored Procedures event class
 - SP:Starting, SP:StmtStarting, RPC:Starting
- T-SQL event class
 - SQL:BatchStarting, SQL:StmtStarting

> Generally create extremely large trace files because usage filters do not apply

> May be **necessary** to determine statements that generate warnings or errors

Other Events

> Completed events

- **Imperative** to monitor **most, if not all**, of these
- Use with duration filter, especially when statement completions monitored
 - 200 milliseconds usually a good starting point

> Errors and Warnings events

- **Extremely** useful for highlighting inefficient sorts, missing statistics, inefficient joins
- Low overhead

> Performance events

- Potentially very useful
- **Execution Plan, Show Plan All , Show Plan Statistics , and Show Plan Text** can be difficult to decipher
- Creates **enormous** trace files so usage should be carefully limited



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Filters

- > Can greatly reduce data volume
- > Must use **native** units for 2005/2008 duration
- > Examples
 - ID
 - Database
 - Stored Procedure
 - Table
 - User
 - Duration
 - CPU time
 - Reads
 - Writes
- > Multiple “**or condition**” filters unreliable

Use `sp_trace` Instead of Profiler

- > **Most efficient to use `sp_trace` commands to capture trace information**
 - No GUI involved
 - Can be captured to files that compress very well
 - Pre-defined script can be used to generate trace with only most important information
- > **Remote Profiler session does not restart after network interruption**

Trace Script Commands

> **sp_trace_create**

- Defines a trace, but does not start it

> **sp_trace_setevent**

- Adds or removes an event or event column to a trace

> **sp_trace_setfilter**

- Applies a filter to a trace

> **sp_trace_setstatus**

- Modifies the current state of the specified trace, e.g., starts or stops trace

> **fn_trace_getinfo**

- Determines current status of a trace

Using SQL Server to Analyze Traces

> Traces can be imported easily into a SQL Server database using either

- T-SQL commands

Insert <table name> select * from ::fn_trace_gettable (<trace name>)

- Uses **Tempdb** in phases during load
- Can select specific record types (eventclass) during load

- SQL Profiler
- Newer versions can read older versions, but **not** the other way around

> Stored Procedures can be used to

- Summarize data
- Replace numeric IDs with meaningful text
- Locate offending queries
- Join trace data with other performance data



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Using SQL Server to Analyze Traces

> SQL Server 2005/2008 Profiler Note

- Although duration values stored in **microseconds** in trace itself, **SQL Profiler** converts these values to **milliseconds** before displaying them
- No other values stored in microseconds

> **Select * from ::fn_trace_gettable(<trace name>) uses RAW units**

- Must do conversion within analysis routines for SQL 2005/2008 traces

> **Parsing queries can eliminate parameters and facilitate collective-effect analysis**

- Examples around, but very difficult to do correctly and thoroughly



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Using SQL Server to Analyze Traces

> SQL Statements within Stored Procedures

- On SQL 2000, SP name precedes actual statement
 - `<sp_caller> cr/lf select * from ...`
- On SQL 2005/2008, object id column must be used instead
 - ObjectName sometimes will be valid

> Most useful when time units converted to seconds, minutes, or even hours to highlight them

- Aids in management justifications
- View collective effect of repeatedly-used query

Summary Query Statistics Example

Db	Count	Sum Duration (Secs)	Sum Duration (Hrs)	Sum Cpu (Secs)	Sum Cpu (Hrs)	Sum Reads	Sum Writes	Query
X	3	215,851	60	121,306	34	5,256,242,339	74,272	SP1
X	28	45,002	13	9,985	3	2,321,189,602	2,612,182	SP2
Y	199	43,550	12	6,768	2	2,244,209,617	5,726,806	SP3
Y	174	38,162	11	10,587	3	1,836,129,317	4,004,328	SP4
Z	60,221	64,273	18	111,787	31	1,297,532,251	6,408	SELECT * FROM X ORDER BY XYZ DESC
X	26	147,091	41	40,881	11	1,003,560,043	31,911	SP5

Creating a Repeatable Query Test Environment

- > Test on copy of database whenever possible
- > Do **NOT** use these DBCC commands on a production server
- > **Command sequence**
 - dbcc dropcleanbuffers with no_infomsgs
 - dbcc freeproccache with no_infomsgs
 - dbcc freesystemcache('ALL') with no_infomsgs
 - set statistics io on
 - set statistics time on
 - <insert query to be tested here> (use begin tran and rollback)
 - set statistics io off
 - set statistics time off

Interpreting Graphical Query Plans – Part 1

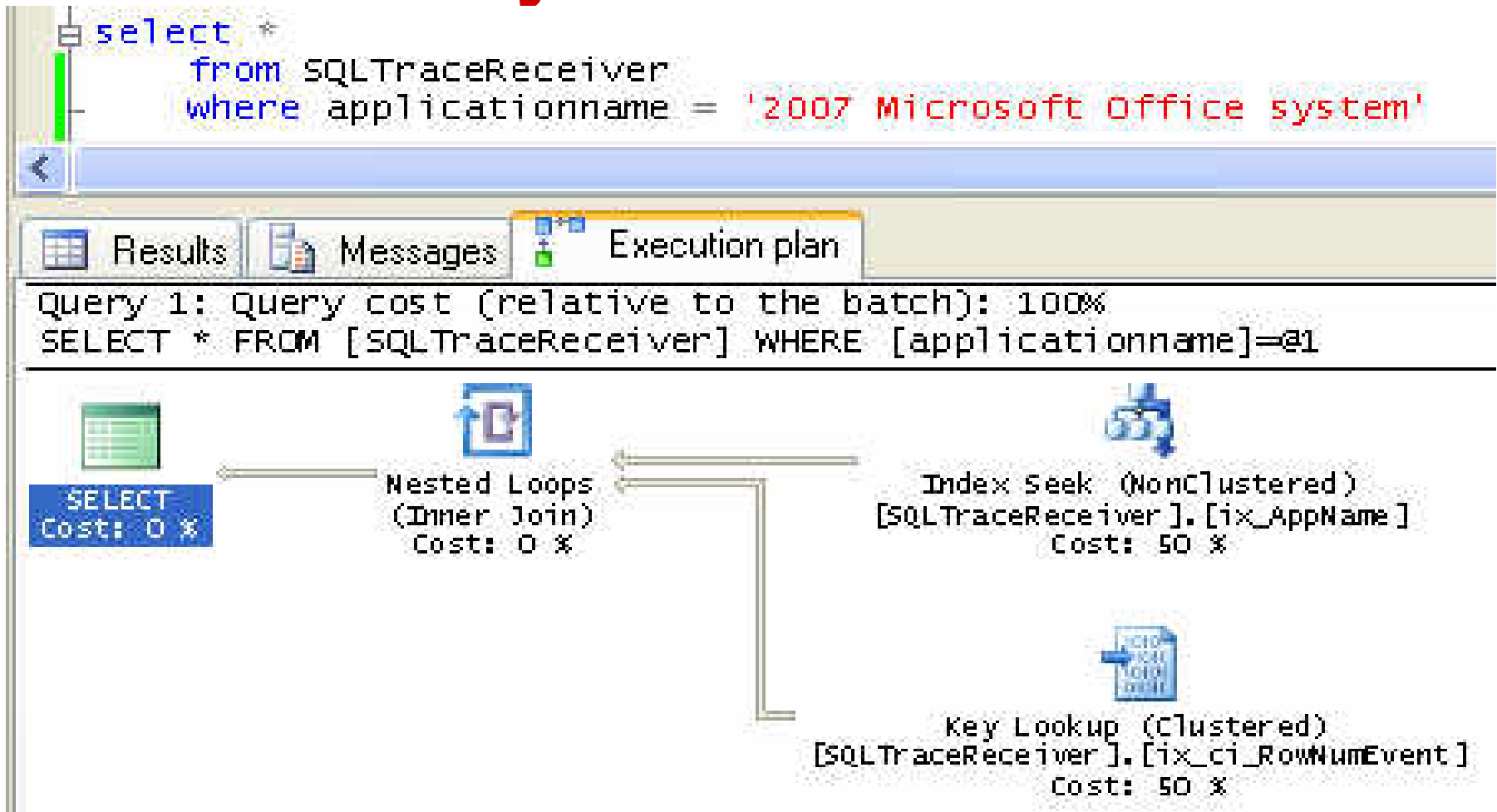
- **Each node in tree structure represented as an icon**
 - Specifies logical and physical operator used to execute that part of the query or statement
- **Each node is related to a parent node**
- **Child nodes of same parents drawn in same column**
 - All nodes in same column do not necessarily have same parent
- **Rules with arrowheads connect each node to its parent**
- **Operators shown as symbols related to specific parent**
 - When mouse placed over operator, a details popup appears
- **Arrow width proportional to number of rows**
 - Actual number of rows used when available. Otherwise, estimated number of rows used



Interpreting Graphical Query Plans - Part 2

- When query contains multiple statements, multiple query execution plans drawn
- Parts of tree structures determined by type of statement executed
- For parallel queries that involve multiple CPUs
 - Properties for each node in graphical execution plan displays information about operating system threads used
 - To view node properties, right-click the node, and then click Properties.

Efficient Query Plan



Efficient Query Plan Statistics

- > SQLTraceReceiver Table contained 20,815,160 rows
- > 243 rows returned by query
- > Table SQLTraceReceiver query statistics

• Scan count	1
• Logical reads	979
• Physical reads	136
• Read-ahead reads	0
• Lob logical reads	486
• Lob physical reads	86

> Execution Times

• CPU time	16 ms
• Elapsed time	2,194 ms



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Inefficient Query Plan

```
select *
  from SQLTraceReceiver
 where isnull(applicationname, parentname) = '2007 Microsoft Office system'
```

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

```
select * from SQLTraceReceiver where isnull(applicationname, parentname) = '2007 Microsoft Office system'
```



Inefficient Query Plan Statistics

> Table SQLTraceReceiver query statistics

- **Scan count** **3**
- Logical reads 1,941,682
- Physical reads 15,552
- Read-ahead reads 1,936,470
- Lob logical reads 486
- Lob physical reads 95

> Execution Times

- CPU time 21,531 ms (efficient one was 16 ms)
- Elapsed time 299,446 ms (efficient one was 2,194 ms)

Efficient vs. Inefficient Comparison

Category	Efficient Query	Inefficient Query	Difference
Logical Reads	979	1,941,682	1,940,703
Physical Reads	136	15,552	15,416
Read-ahead Reads	-	1,936,470	1,936,470
CPU Time (ms)	16	21,531	21,515
Elapsed Time (ms)	2,194	299,446	297,252

Hardware Usage Savings

21.5 **seconds** of CPU time saved

15,416 disk reads eliminated

If query run 100 times, 36 **minutes** of CPU time saved



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Query Plan Popups

```
no_intomsgs; dbcc t
statistics time on
```

Execution plan

```
to the batch): 100%
r] WHERE [applicationr
```



Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Number of Rows	243
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Number of Executions	1
Estimated Operator Cost	0.0032831 (50%)
Estimated Subtree Cost	0.0032831
Estimated Number of Rows	1
Estimated Row Size	67 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	1

Object

[WinPerfDataSQL].[dbo].[SQLTraceReceiver].
[ix_AppName]

Output List

[WinPerfDataSQL].[dbo].[SQLTraceReceiver].RowNumber,
[WinPerfDataSQL].[dbo].
[SQLTraceReceiver].ApplicationName, [WinPerfDataSQL].
[dbo].[SQLTraceReceiver].EventClass

Seek Predicates

Seek Keys[1]: Prefix: [WinPerfDataSQL].[dbo].
[SQLTraceReceiver].ApplicationName = Scalar Operator
(N'2007 Microsoft Office system')

Index Seek & Key Lookup Popups from Efficient Query Plan

Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Number of Rows	243
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Number of Executions	1
Estimated Operator Cost	0.0032831 (50%)
Estimated Subtree Cost	0.0032831
Estimated Number of Rows	1
Estimated Row Size	67 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	1

Object

[WinPerfDataSQL].[dbo].[SQLTraceReceiver].
[ix_AppName]

Output List

[WinPerfDataSQL].[dbo].[SQLTraceReceiver].RowNumber,
[WinPerfDataSQL].[dbo].
[SQLTraceReceiver].ApplicationName, [WinPerfDataSQL].
[dbo].[SQLTraceReceiver].EventClass

Seek Predicates

Seek Keys[1]: Prefix: [WinPerfDataSQL].[dbo].
[SQLTraceReceiver].ApplicationName = Scalar Operator
(N'2007 Microsoft Office system')

Key Lookup (Clustered)

Uses a supplied clustering key to lookup on a table that has a clustered index.

Physical Operation	Key Lookup
Logical Operation	Key Lookup
Actual Number of Rows	243
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Number of Executions	243
Estimated Number of Executions	1
Estimated Operator Cost	0.0032831 (50%)
Estimated Subtree Cost	0.0032831
Estimated Number of Rows	1
Estimated Row Size	9172 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	3

Object

[WinPerfDataSQL].[dbo].[SQLTraceReceiver].
[ix_ci_RowNumEvent]

Seek Predicates

Seek Keys[1]: Prefix: [WinPerfDataSQL].[dbo].
[SQLTraceReceiver].EventClass, [WinPerfDataSQL].[dbo].
[SQLTraceReceiver].RowNumber = Scalar Operator
([WinPerfDataSQL].[dbo].[SQLTraceReceiver].
[EventClass]), Scalar Operator([WinPerfDataSQL].[dbo].
[SQLTraceReceiver].[RowNumber])

Clustered Index Scan Popup from Inefficient Query Plan

Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Number of Rows	243
Estimated I/O Cost	1432.95
Estimated CPU Cost	11.4484
Number of Executions	2
Estimated Number of Executions	1
Estimated Operator Cost	1444.4 (100%)
Estimated Subtree Cost	1444.4
Estimated Number of Rows	1
Estimated Row Size	9231 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	2

Predicate

isnull([WinPerfDataSQL].[dbo].[SQLTraceReceiver].[ApplicationName],[WinPerfDataSQL].[dbo].[SQLTraceReceiver].[ParentName])=N'2007 Microsoft Office system'

Object

[WinPerfDataSQL].[dbo].[SQLTraceReceiver].[ix_ci_RowNumEvent]

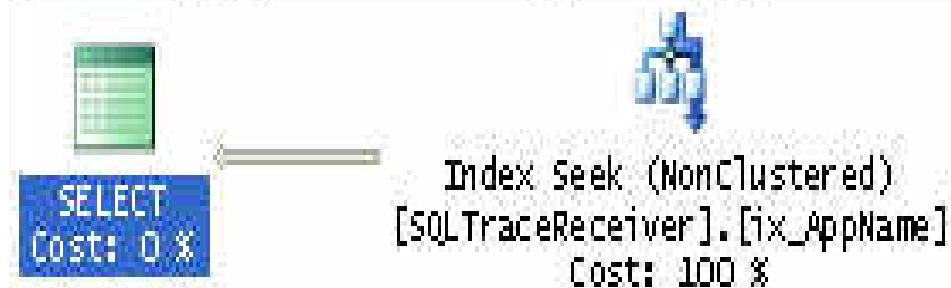


Covering Index Query Plan

```
select applicationname
  from SQLTraceReceiver
 where applicationname = '2007 Microsoft Office system'
```

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT [applicationname] FROM [SQLTraceReceiver] WHERE [applicationname]=@1



Covering Index Popup

Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Number of Rows	243
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Number of Executions	1
Estimated Operator Cost	0.0032831 (100%)
Estimated Subtree Cost	0.0032831
Estimated Number of Rows	1
Estimated Row Size	59 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	0

Object

[WinPerfDataSQL].[dbo].[SQLTraceReceiver].[ix_AppName]

Output List

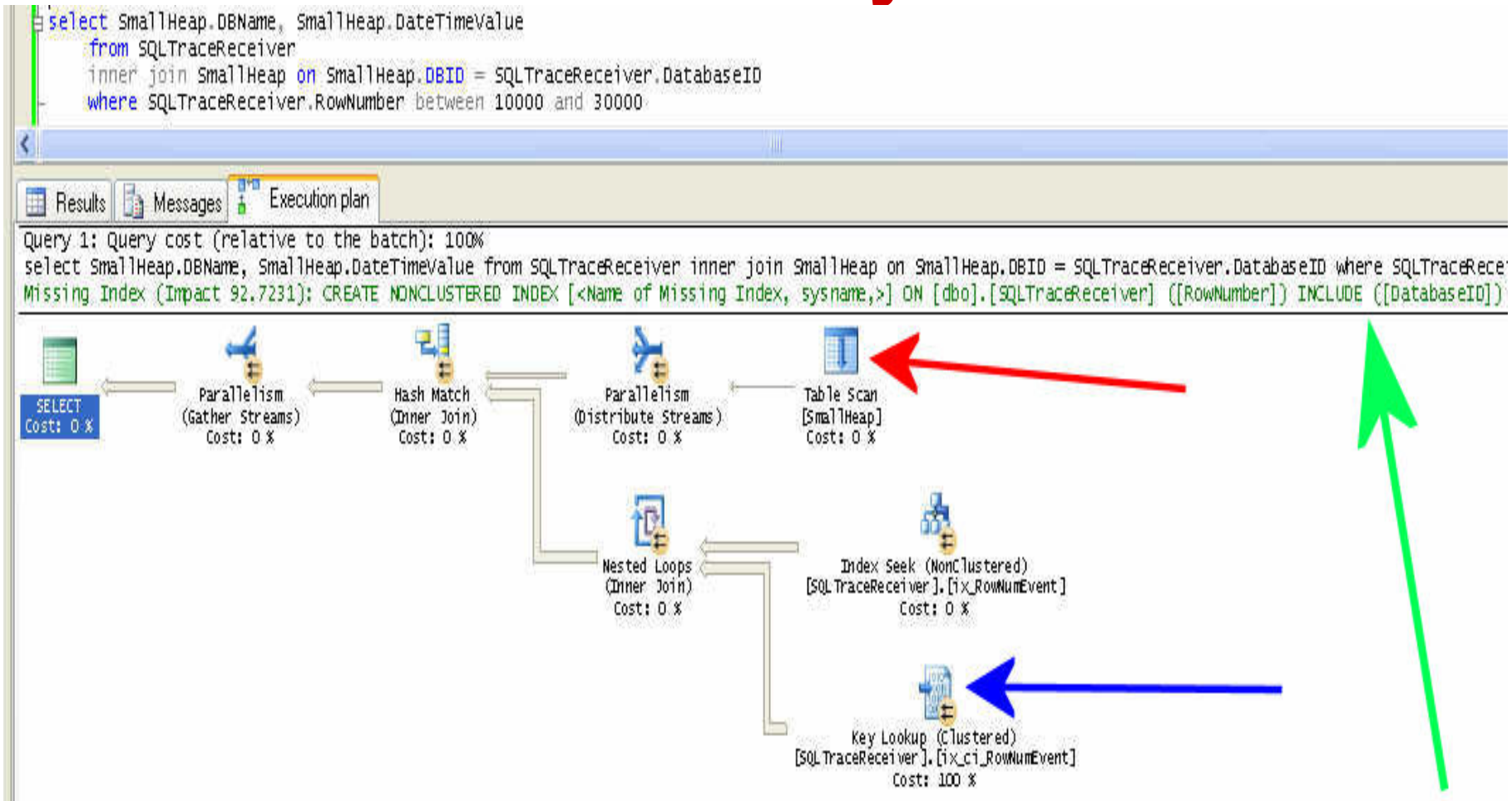
[WinPerfDataSQL].[dbo].
[SQLTraceReceiver].ApplicationName

Seek Predicates

Seek Keys[1]: Prefix: [WinPerfDataSQL].[dbo].
[SQLTraceReceiver].ApplicationName = Scalar Operator
(CONVERT_IMPLICIT(nvarchar(4000),[@1],0))



Small Table Scan Query Plan



Conclusions

- > Windows Performance Monitor should always be used to focus tuning efforts
- > **Extremely** important to combine Windows system performance and SQL Server information
 - Especially for processor, memory, and I/O



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Conclusions

> SQL Query and SQL Trace both provide extremely useful insights into

- How specific databases, queries, transactions, batches, and stored procedures perform
 - Display Estimated and Actual Execution Plan in Mgmt Studio
 - Trace records in Profiler
 - Trace analysis from loaded data

> Filtering enables one to collect trace data from a production system without damaging performance

- Filter duration (> 200 ms)
- Filter # of logical reads (> 1,000)



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Conclusions

- Lightweight T-SQL trace scripts can be used to gather very specific and inexpensive information regularly
- Importing SQL Trace output into SQL Server database greatly simplifies analysis
- Once specific queries or stored procedures have been identified as offenders, additional data can be gathered for just those entities



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com

Next Steps

1. Let us analyze your WORST performing SQL Server:

- a) SQLRx will perform a collection of your native SQL and Windows performance data.
- b) SQLRx will return a “Lite” Analysis with Recommendations.

2. Migrating to 2005 or 2008?

- a) Migrations and Upgrades (read about *ReplayableTrace™* at www.sqlrx.com). Don't migrate without using this valuable tool.



2008 DATA MANAGEMENT SOLUTIONS
PARTNER OF THE YEAR-WINNER
Database Management

SQLRx.com