SQL SERVER DAY 2009

Breaking the ETL world record with Integration Services

Henk van der Valk

IT Workload optimizer

UNISYS – *European Performance Center*

Agenda

- Part 1: Setting an ETL World record
  - Loading data: Bottlenecks & Optimizations found
  - SQL 2008 R2 & Solid State on 96 Cores

- Part 2: The data is all loaded, what's next?
  - Optimizing an Evil DWH query
    - Speeding up large table scans

Performance tuning tips & Quick wins to try @ home !

About the speaker

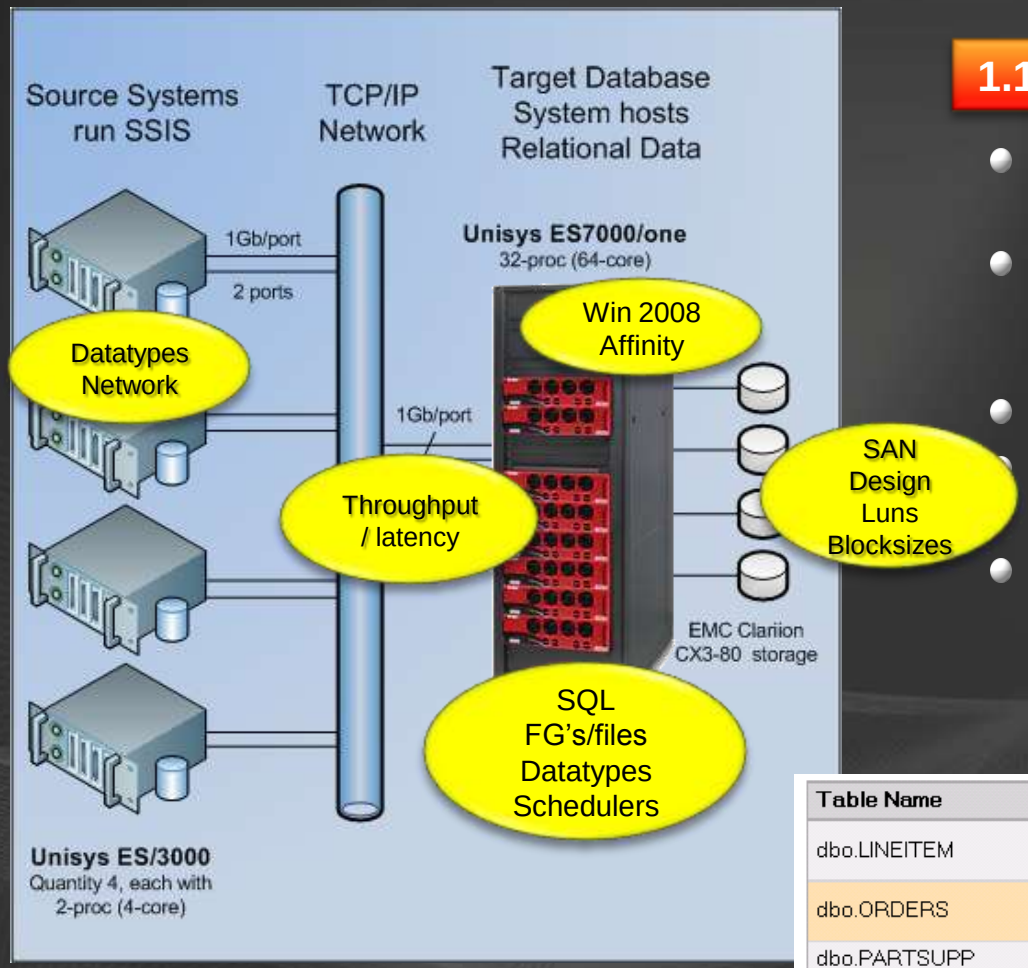
- Co-Founder Unisys ES7000 Performance Centers (2001)
- Over 5 year history of in depth SSIS / SQL product performance testing with the dev teams
- Performance troubleshooter & Workload optimizer
- 23+ years into computers…
- Deals with the largest & most demanding IT environments (in the world)
- Participates in Dutch + EU SQLPass & Performance SIG

A crazy idea!

- Someone once did 1 TB TPC-H load in 50 mins
  - 1900+ spindles
- Informatica has benchmark on 45 mins
- How many order lines will that be??!!
- That's 512 MB / sec both reading and writing
- Is that fast?

  … sounds terribly too slow to me … !
- Can we do this in 30 mins? (582+ MB/sec)
  - What will we need?

SQL Server Integration Services Record ETL Performance



1.18 TB of data loaded in 29 min 54 sec

- Data model reflects a wholesale supplier data warehouse
- Data is read from text files, sent over network, and stored in a single database image
- 8.5+ Billion records
- SSIS runs on source systems, reflecting a distributed environment
- Built upon
  - SQL Server 2008 Ent. Ed. February CTP
  - Windows Server 2008 Datacenter Ed.

Table Name	# Records	Reserved (KB)	Data (KB)
dbo.LINEITEM	5,999,989,709	679,400,512	679,394,408
dbo.ORDERS	1,500,000,000	182,977,472	182,972,144
dbo.PARTSUPP	800,000,000	125,819,072	125,813,888
dbo.PART	200,000,000	29,367,808	29,361,840
dbo.CUSTOMER	150,000,000	26,157,248	26,152,152
dbo.SUPPLIER	10,000,000	1,598,336	1,592,608

Microsoft SQL Server 2008

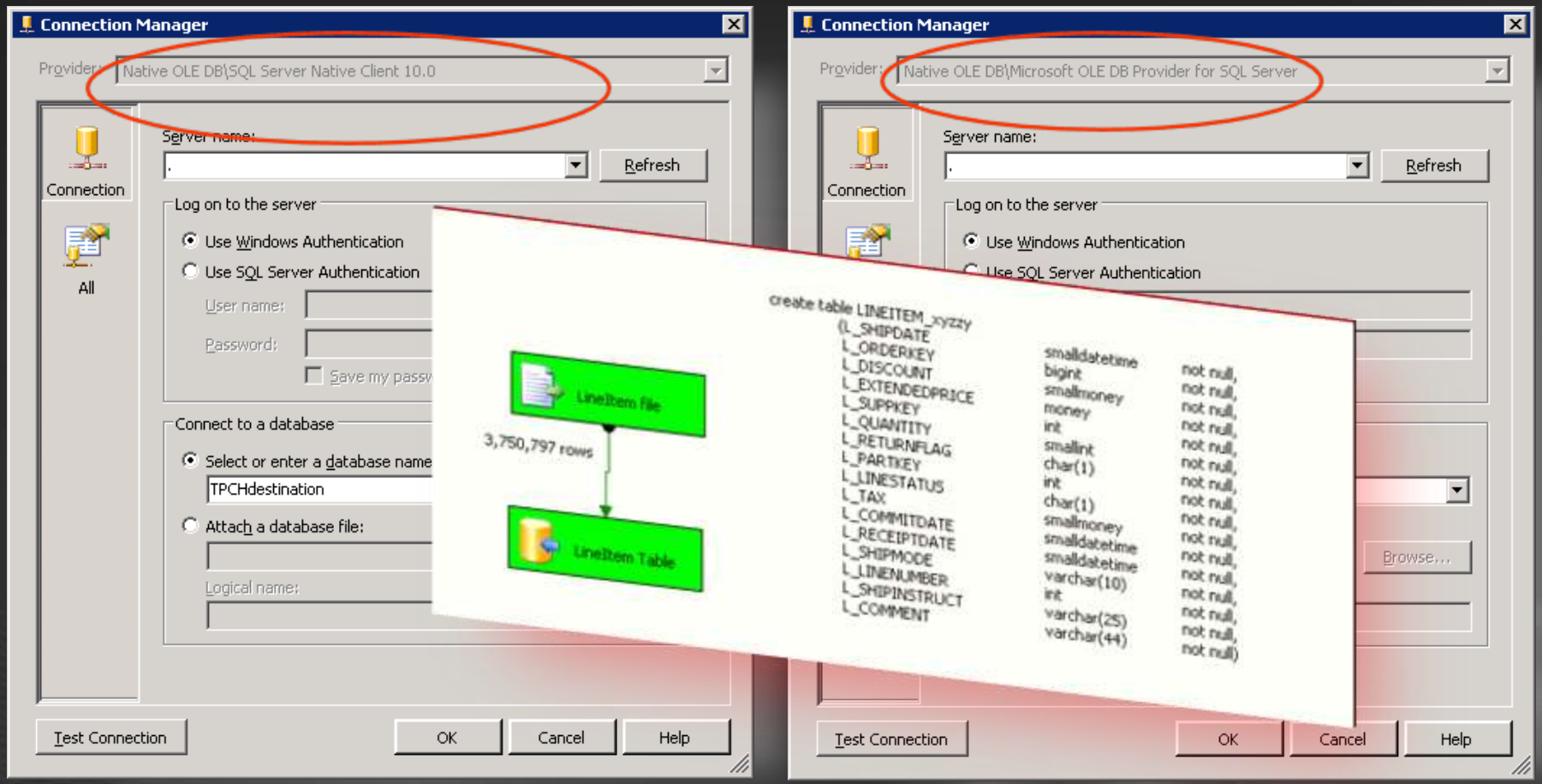
Question:

- If we have to load data from a flat file as ***<u>fast as possible</u>*** into SQLServer:

- What methods are there for Bulk load?
- Which will be fastest?
  - BCP
  - BULK INSERT
  - SSIS - Data Destinations

- Multiple systems for SSIS and SQL?

Functionality	Integration Services		BULK INSERT	BCP	INSERT … SELECT
	SQL Dest.	OLE DB Dest			
Protocol	Shared Memory	TCP/IP Named Pipes	In Memory	TCP/IP Shared Memory Named Pipes	In Memory
Speed	Faster / Fastest(4)	Fast / Fastest (1)	Fastest	Fast	Slow / Fastest (2)
Data Source	Any	Any	Data File Only	Data File Only	Any OLE DB
Bulk API Support	Not Native	Not ORDER Not Native	All	All	No Hints Allowed
Lock taken with TABLOCK hint on heap	BU	BU	BU	BU	X
Can transform in transit	Yes	Yes	No	No	Yes
I/O Read block Size	128 KB for text files	Depends(3)	64 KB	64 KB	Up to 512 KB
SQL Server Version	2005 and 2008	2005 and 2008	7.0, 2000, 2005, and 2008	6.0, 7.0, 2000, 2005, and 2008	2008
Invoked from	DTEXEC / BIDS	DTEXEC / BIDS	Transact-SQL	Command Line	Transact-SQL



3750797 rows
463 MB

```
BULK INSERT dbo.LINEITEM_1 FROM
'C:\Readers\R0\lineitem.tbl.1'
WITH (FIELDTERMINATOR = '|',
ROWTERMINATOR = '|\n',
TABLOCK)
```

Tip: Sharpen data type
Money type (10-20% improvement) (Still Applies to SQL2005)

- Use Money type instead of decimal columns
  - Storing as money (a 8-byte integer with implied 4 decimal digits). TDS (Tabular Data Stream) is the format SQL Server uses for transfer of data over the wire
  - Money, because it is fixed length, is alignment efficient for the CPU.

Column Name	Data Type	Allow Nulls
ProductID	int	☐
RecCnt	smallint	☐
Gross	decimal(11, 2)	☐
Quantity	decimal(9, 3)	☐
IsPledge	bit	☐
StornoType	tinyint	☐
AuditID	smallint	☐
BonNumber	int	☐
ReceiptDateID	int	☐
Total	decimal(11, 2)	☐
OrgID	int	☑
ReceiptTimeID	smallint	☐
BonPosID	tinyint	☐
		☐

Column Name	Data Type	Allow Nulls
OrgID	int	☑
BonNumber	int	☐
RecCnt	smallint	☐
ProductID	int	☐
ReceiptDateID	int	☐
ReceiptTimeID	smallint	☐
Gross	money	☐
Quantity	money	☐
Total	money	☐
BonPosID	tinyint	☐
IsPledge	bit	☐
StornoType	tinyint	☐
AuditID	smallint	☐
		☐

Loading Flat file data into a database

demo

Single flat file bulk insert results

- 463 MB / 3750797 rows

Method	Connection manager	Provider	Avg. Bulk copy rows/sec	CPU	Reads	Writes	Duration (millisec)
T-SQL Bulk Insert			150000	33446	165603	52763	50432
Native OLEDB \msft OLEDB provider for sql server (TCP)	localhost. TPCHDest 4 KB	SQLOLEDB.1	143000	29780	167120	53237	38444
Native OLEDB \msft OLEDB provider for sql server (TCP)	localhost. TPCHDest 32 KB	SQLOLEDB.1	160000	31684	167120	53235	37460
Native OLEDB \SQL Server Native client 10.0 (in memory)	DestinationDB - 0	SQLNCLI10.1	159000	30997	167259	53237	37951
Native OLEDB \SQL Server Native client 10.0 (in memory)	DestinationDB 32 KB	SQLNCLI10.1	**176000**	30825	167197	53233	33914

17% faster

The fastest method: SSIS with In memory connection (SNAC) + 32 KB packet Size

WRproject (Running) - Microsoft Visual Studio (Administrator)

File Edit View Project Build Debug Data Format SSIS Tools Window Help

Development

LineItemSOLED...dtsx [Design]* | LineItemSOLEDB....dtsx [Design]

Control Flow | Data Flow | Event Handlers | Package Explorer | Progress

Data Flow Task: LineItem

LineItem file

3,750,797 rows

LineItem Table

```
create table LINEITEM_xyzzy
        (L_SHIPDATE        smalldatetime   not null,
         L_ORDERKEY        bigint          not null,
         L_DISCOUNT        smallmoney      not null,
         L_EXTENDEDPRICE   money           not null,
         L_SUPPKEY         int             not null,
         L_QUANTITY        smallint        not null,
         L_RETURNFLAG      char(1)         not null,
         L_PARTKEY         int             not null,
         L_LINESTATUS      char(1)         not null,
         L_TAX             smallmoney      not null,
         L_COMMITDATE      smalldatetime   not null,
         L_RECEIPTDATE     smalldatetime   not null,
         L_SHIPMODE        varchar(10)     not null,
         L_LINENUMBER      int             not null,
         L_SHIPINSTRUCT    varchar(25)     not null,
         L_COMMENT         varchar(44)     not null)
```

Progress: Cleanup - 0 percent complete
Progress: Cleanup - 50 percent complete
Progress: Cleanup - 100 percent complete
Finished, 11:37:57 PM, Elapsed time: 00:00:33.914

Connection Manager

Provider: Native OLE DB\SQL Server Native Client 10.0

Connection | All

Auto Translate	True
Connection Plan	
Current Language	
DataTypeCompatibility	0
Failover Partner	
Failover Partner SPN	
Initial File Name	
MARS Connection	False
Network Address	
Network Library	
Old Password	
Packet Size	**32767**
Replication server name connect opti	
Server SPN	
Tag with column collation when possil	False
Trust Server Certificate	False
Use Encryption for Data	False
Use Procedure for Prepare	1
Workstation ID	UNISYS-ES7000

Provider
The name of the OLE DB Provider to use when connecting to the Data Source.

Test Connection | OK | Cancel | Help

Let's do the naïve thing

- Lets kick of 64 concurrent bulk Inserts with SQL
  - Single filegroup, single file
  - Let's look at CPU load

- Tools of the trade:
  - Taskmanager
  - Sys.dm_os_wait_stats
- Where do we spend the time?
- Interpretation?

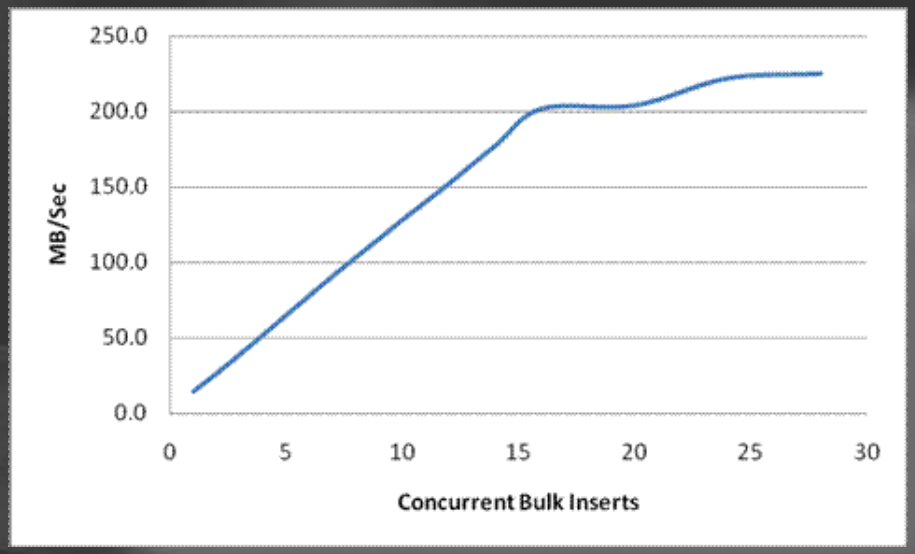
More I/O – easy !

	wait_type	waiting_tasks_co...	wait_time ...	max_wa
1	PAGEIOLATCH_UP	791	198197	396
2	PAGELATCH_UP	30016	66624	53
3	SOS_SCHEDULER_YIELD	52150	7806	8
4	PAGELATCH_SH	4244	6123	50
5	PAGEIOLATCH_EX	3	144	43
6	IMPPROV_IOWAIT	23	30	8
7	PAGEIOLATCH_SH	8	18	5

- Add more LUN's
  - (total 16+1)
- Spread out the data using SQL Server files
- Rerun...
- Tools of the trade
  - SQL Activity Monitor
    - What's happening?
  - Sys.dm_os_waiting_tasks

Bulk insert on single heap

- Up to 16 concurrent bulk operations
  - Otherwise the internal allocation data structure in SQL Server will start to become a bottleneck
- Check sys.dm_os_latch_stats
  For ALLOC_FREESPACE_CACHE
- For linear scale beyond this point, <u>partition the table</u>



Ehh… What are we waiting for?

- PAGELATCH_UP
  - What's this?
- Tools of the trade
  - Sys.dm_os_buffer_descriptors
- Interpretation?

unisys-es7000.TPCH_Contr...ts.sql | unisys-es7000.T...

```
DBCC SQLPERF ('sys.dm_os_wait_stats
GO
SELECT * FROM sys.dm_os_wait_stats
ORDER BY wait_time_ms DESC

/* what is wait, and for what? */
SELECT * FROM sys.dm_os_waiting_tas

/* What resource? */
SELECT * FROM sys.dm_os_buffer_desc
WHERE database_id = 5 AND file_id =
```

Results | Messages

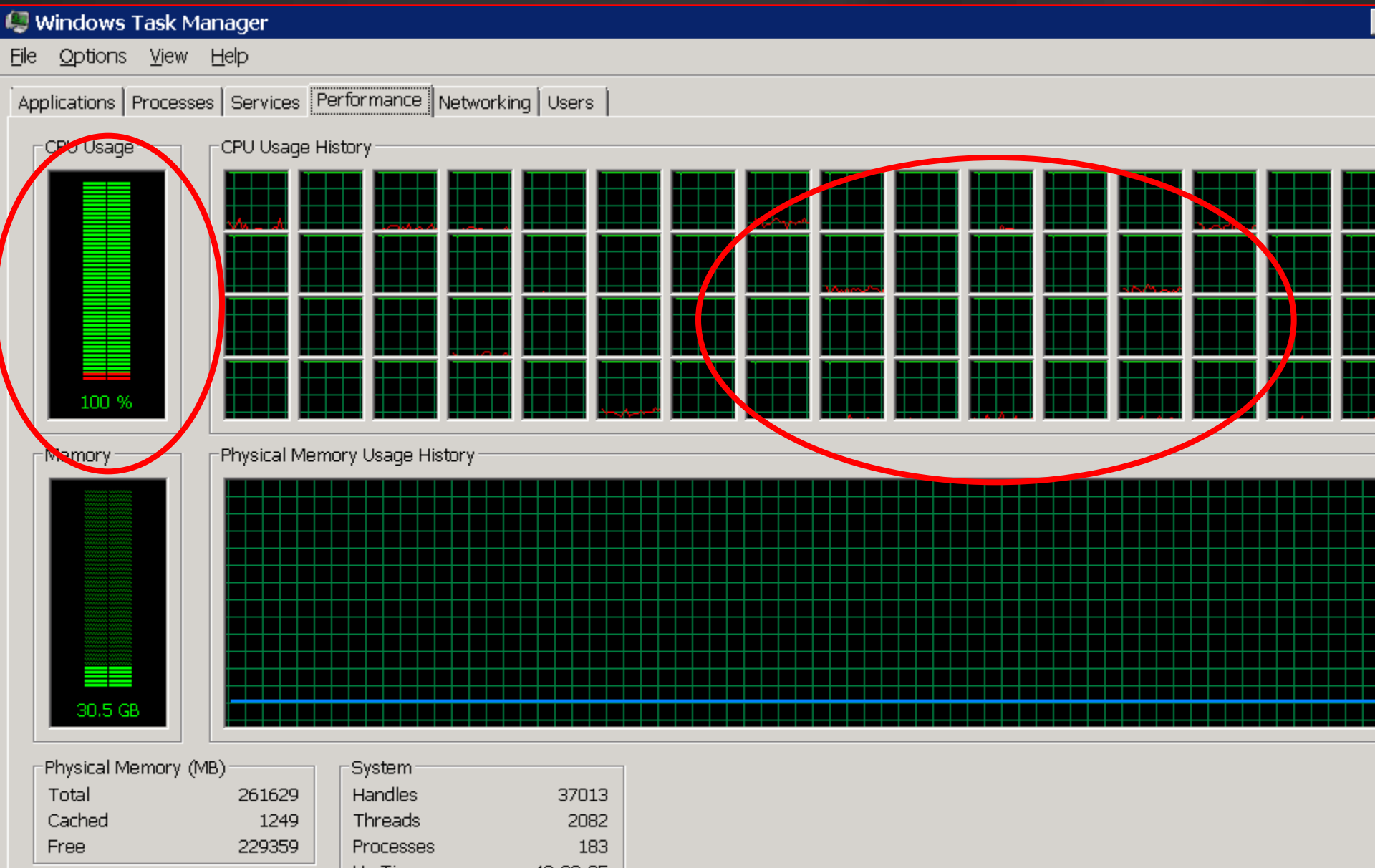
	wait_type	waiting_tas
1	PREEMPTIVE_COM_GETDATA	48239239
2	PAGELATCH_UP	242750
3	FT_IFTS_RWLOCK	2
4	LAZYWRITER_SLEEP	89
5	LOGMGR_QUEUE	139
6	SQLTRACE_BUFFER_FLUSH	22
7	REQUEST_FOR_DEADLOCK_SEARCH	17
8	SLEEP_TASK	24028

	session_id	exec_context_id	wait_type	resource_description
1	85	0	PAGELATCH_UP	6:1:2895504
2	122	0	PAGELATCH_UP	6:1:2895504
3	99	0	PAGELATCH_UP	6:1:2895504

How to fix PFS contention?

- Each data file has it's own PFS pages (**P**age **F**ree **S**pace)
- Solution: Add more database files!
- There is no free meal... (Or maybe?)
- Monitor:
  - Logical Disk – Avg Disk bytes / write

100% CPU load - Maximum performance ?



Initial Peak around 400 MB/sec read

Reliability and Performance Monitor

File Action View Favorites Window Help

- Reliability and Performance
  - Monitoring Tools
    - Performance Monito
    - Reliability Monitor
  - Data Collector Sets
    - User Defined
      - bulk
      - BULK_INSERT
    - System
    - Event Trace Session
    - Startup Event Trace
  - Reports

TSQL BulkInserts on 64 Core ES7000/one

100 80 60 40 20 0

6:06:28 PM 5:58:56 PM 6:00:16 PM 6:01:36 PM 6:02:56 PM 6:04:16 PM 6:06:27 PM

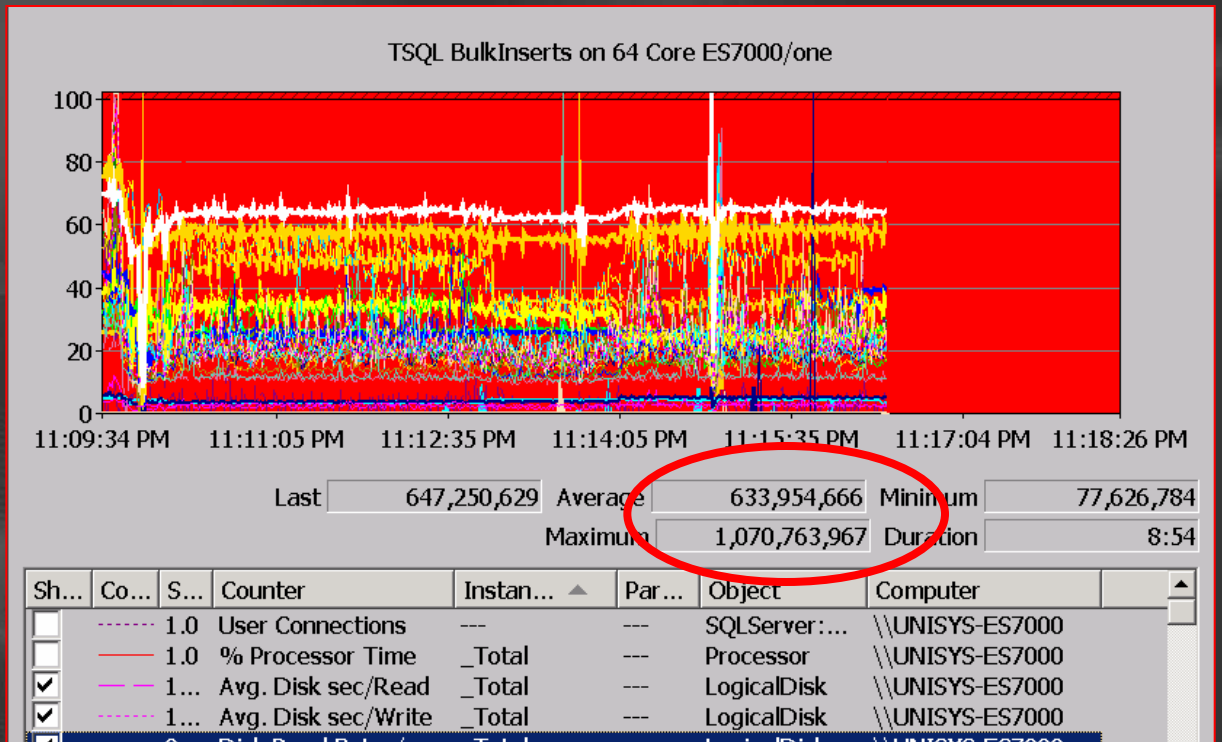
Last	0.000	Average	115,804,168
Minimum	0.000	Maximum	432,651,518
Duration	8:54		

Sh...	Co...	S...	Counter	Instance	Par...	Object	Computer
✓		1.0	User Connections	---	---	SQLServer:...	\\UNISYS-ES7000
		1.0	% Processor Time	_Total	---	Processor	\\UNISYS-ES7000
✓		1...	Avg. Disk sec/Read	_Total	---	LogicalDisk	\\UNISYS-ES7000
✓		1...	Avg. Disk sec/Write	_Total	---	LogicalDisk	\\UNISYS-ES7000
✓		0....	Disk Read Bytes/sec	_Total	---	LogicalDisk	\\UNISYS-ES7000
✓		0....	Disk Write Bytes/sec	_Total	---	LogicalDisk	\\UNISYS-ES7000
✓		1...	Avg. Disk sec/Read	C:	---	LogicalDisk	\\UNISYS-ES7000
✓		1...	Avg. Disk sec/Write	C:	---	LogicalDisk	\\UNISYS-ES7000

64 core / 64 Bulk Inserts with –x

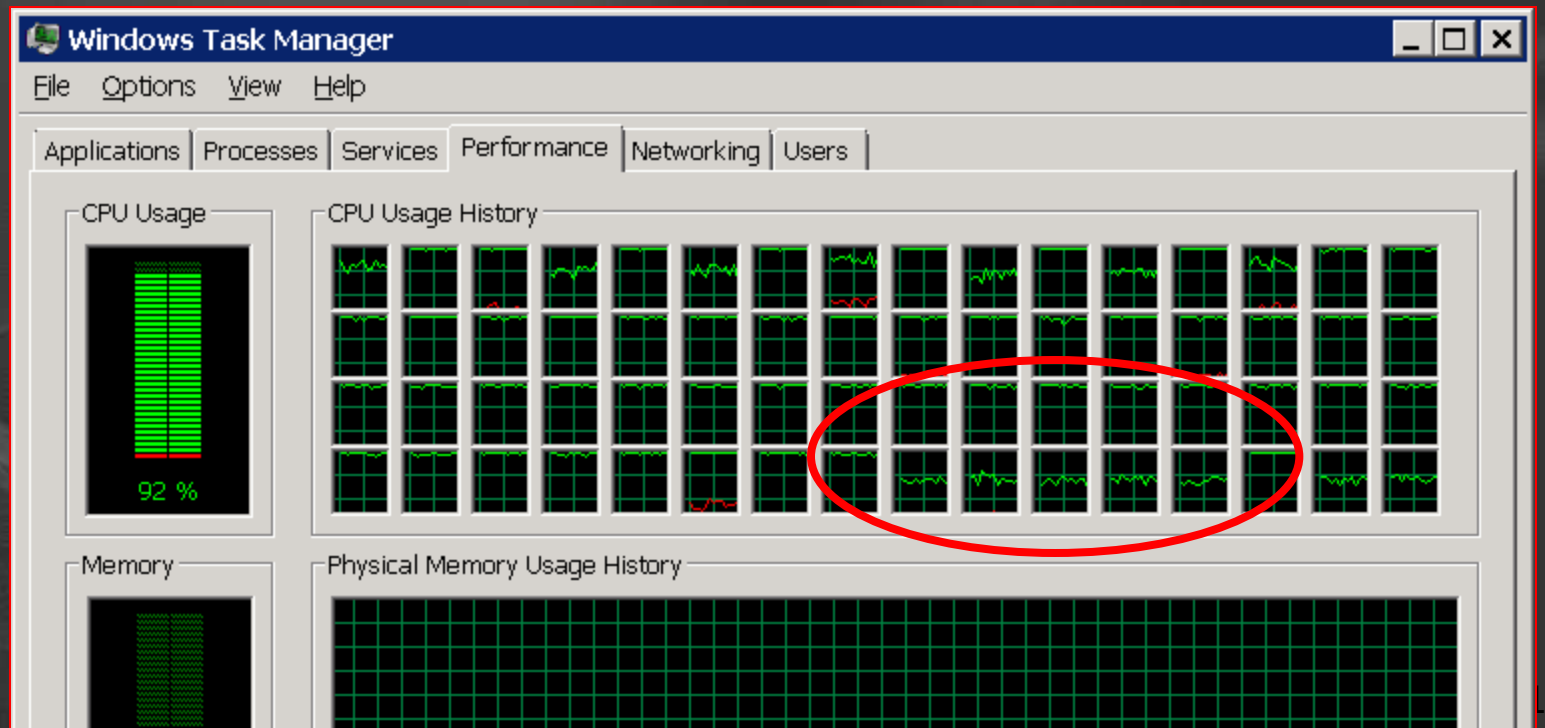
No worries:
Fixed in SQL2008 RTM

- Kernrate & Minidump analysis show lots of perf logging overhead
- Starting SQLServer with –x option boosts throughput:



Multiple Bulk Inserts tasks on same scheduler

- Can be overcome with Soft Numa / tcp port mappings
- Or simpler the ”Harakiri” SP



Soft Numa on 64 cores

- Assign BULK INSERT Tasks to dedicated CPU's (both SQL2005/2008)

- [HKEY_LOCAL_MACHINE\SOFTWA
Server\100\NodeConfiguration\Node6
- "CpuMask"=hex:00,00,00,00,00,00,00

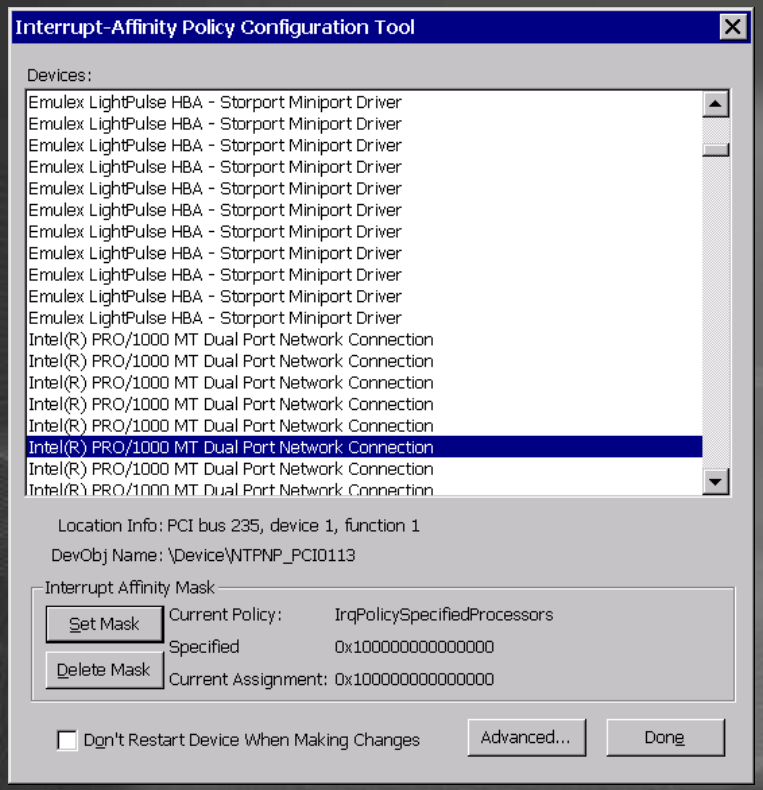
- [HKEY_LOCAL_MACHINE\SOFTWA
Server\MSSQL10.MSSQLSERVER\M
\Tcp]
- "ListenOnAllIPs"=dword:00000001

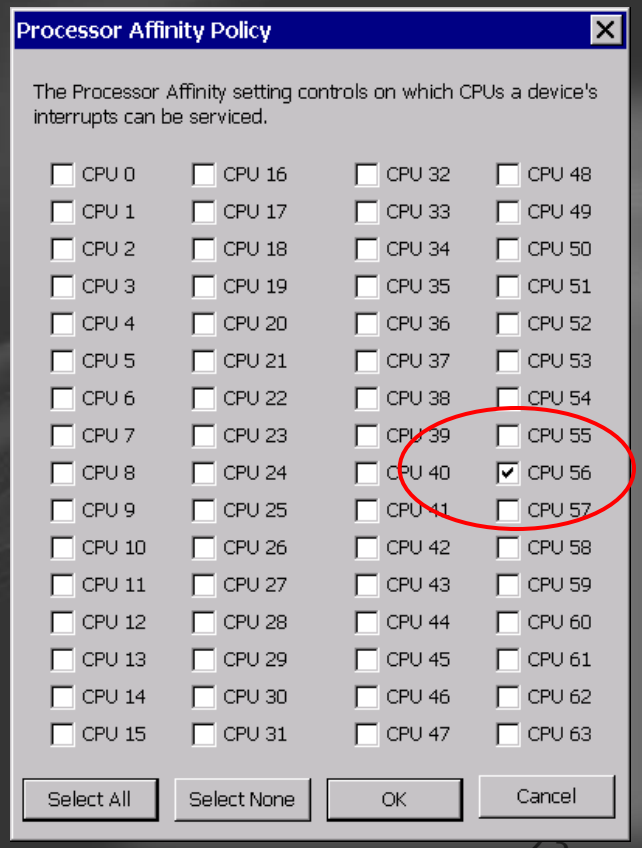
- [HKEY_LOCAL_MACHINE\SOFTWA
Server\MSSQL10.MSSQLSERVER\M
\Tcp\IPAll]
- "TcpPort"="2000[0x00000001],2001[0x00000002],

1/28/2008 1:25:00...	spid101	Using 'xpstar.dll' version '2007.100.1098' to execu
1/28/2008 1:24:53...	spid75s	Recovery is complete. This is an informational mes
1/28/2008 1:24:53...	Server	SQL Server is now ready for client connections. T
1/28/2008 1:24:53...	Server	The SQL Server Network Interface library could n
1/28/2008 1:24:53...	Server	Dedicated admin connection support was establis
1/28/2008 1:24:53...	Server	Server is listening on [127.0.0.1 <ipv4> 1434].
1/28/2008 1:24:53...	Server	SQL Server Network Interfaces initialized listeners
1/28/2008 1:24:53...	Server	Server is listening on ['any' <ipv4> 2001].
1/28/2008 1:24:53...	Server	SQL Server Network Interfaces initialized listeners
1/28/2008 1:24:53...	Server	Server is listening on ['any' <ipv4> 2063].
1/28/2008 1:24:53...	Server	SQL Server Network Interfaces initialized listeners
1/28/2008 1:24:53...	Server	Server is listening on ['any' <ipv4> 2062].
1/28/2008 1:24:53...	Server	SQL Server Network Interfaces initialized listeners
1/28/2008 1:24:53...	Server	Server is listening on ['any' <ipv4> 2061].
1/28/2008 1:24:53...	Server	SQL Server Network Interfaces initialized listeners
1/28/2008 1:24:53...	Server	Server is listening on ['any' <ipv4> 2060].
1/28/2008 1:24:53...	Server	SQL Server Network Interfaces initialized listeners
1/28/2008 1:24:53...	Server	Server is listening on ['any' <ipv4> 2059].
1/28/2008 1:24:53...	Server	SQL Server Network Interfaces initialized listeners
1/28/2008 1:24:53...	Server	Server is listening on ['any' <ipv4> 2058].
1/28/2008 1:24:53...	Server	SQL Server Network Interfaces initialized listeners
1/28/2008 1:24:53...	Server	Server is listening on ['any' <ipv4> 2057]

Handling the Network traffic

- IntPolicy tool for Interrupt Affinity
- On ES7000 SQL Server, assign NIC interrupts & DPC's onto dedicated CPU's



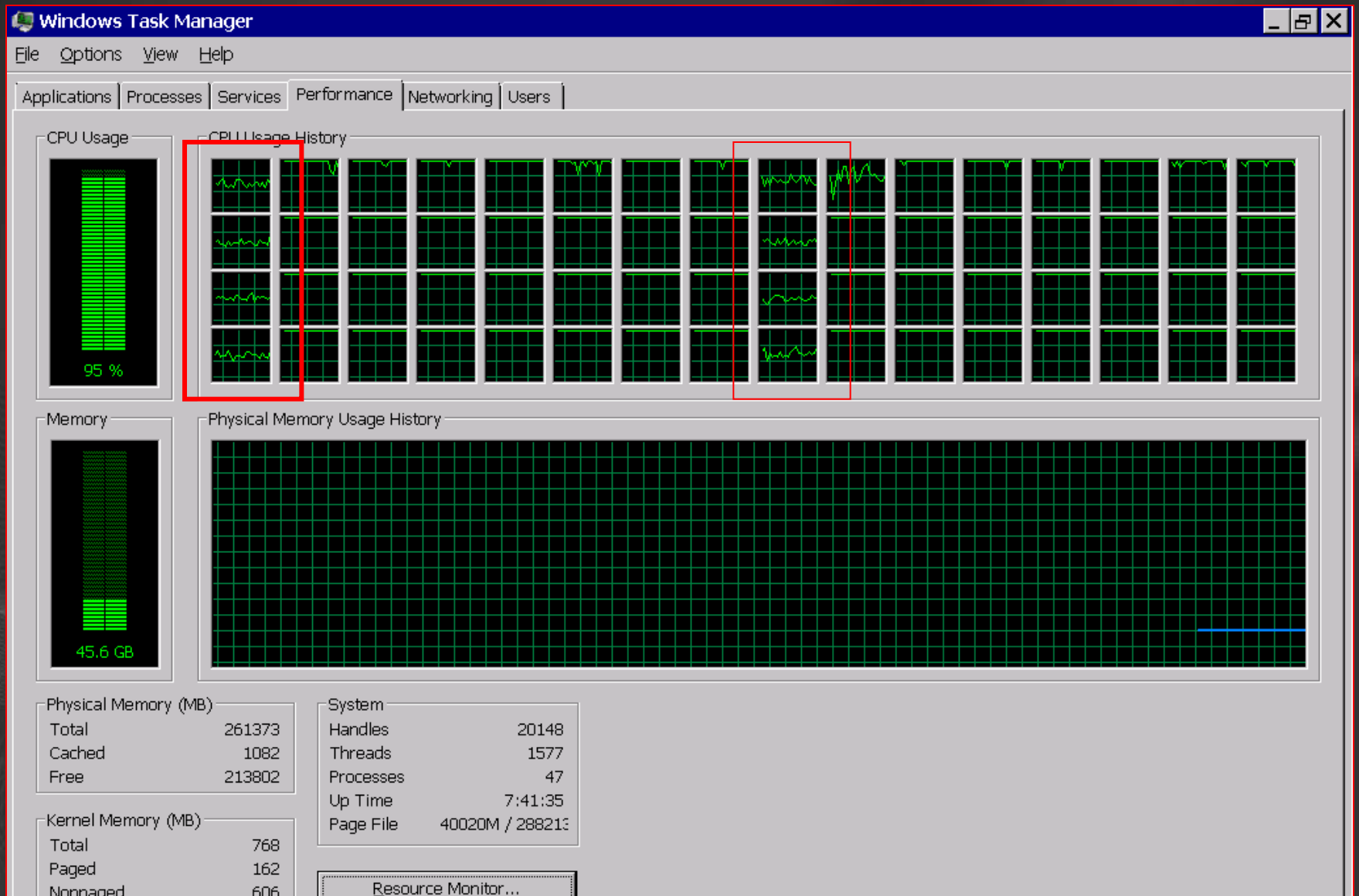


Network optimiziation - Intel Pro/1000 MT

Apply changes to each of the (16) network cards:

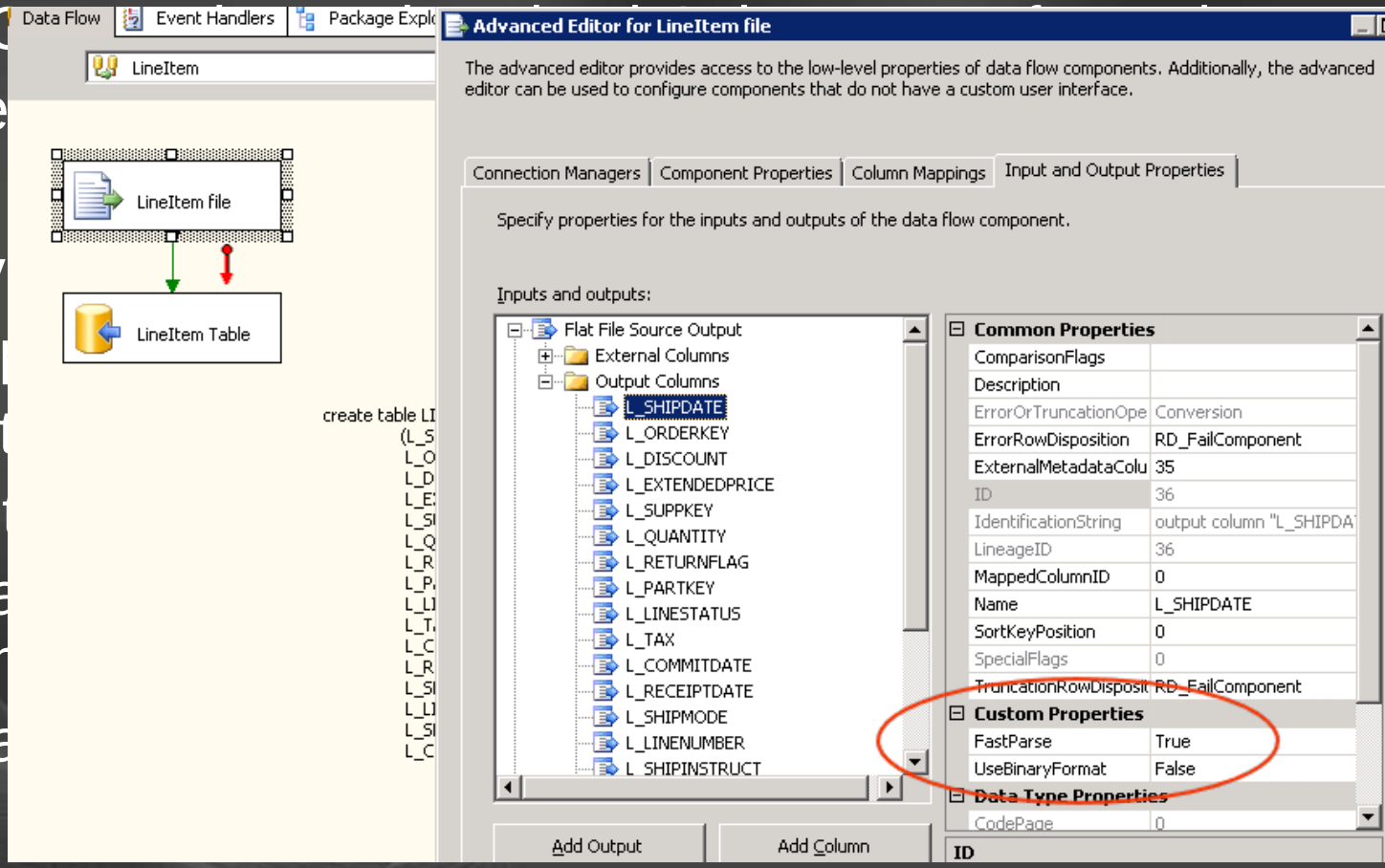
0) Adaptive Inter-Frame spacing disabled
1) Flow control = Tx & Rx enabled
   client & server Interrupt Moderation = Medium
2) Jumbo Packet = 9014 bytes enabled
3) Client & server Interrupt Moderation = Medium
   Coalesc buffers = 256
4) Set server Rx buffers to 512 and server Tx buffers to 512
5) Set client Rx buffers to 512 and client Tx buffers to 256
6) Link speed 1000mbps Full Duplex

Interrupt Affinity set for 8 network cards on 8 dedicated CPU's



Other SSIS flat file best practices

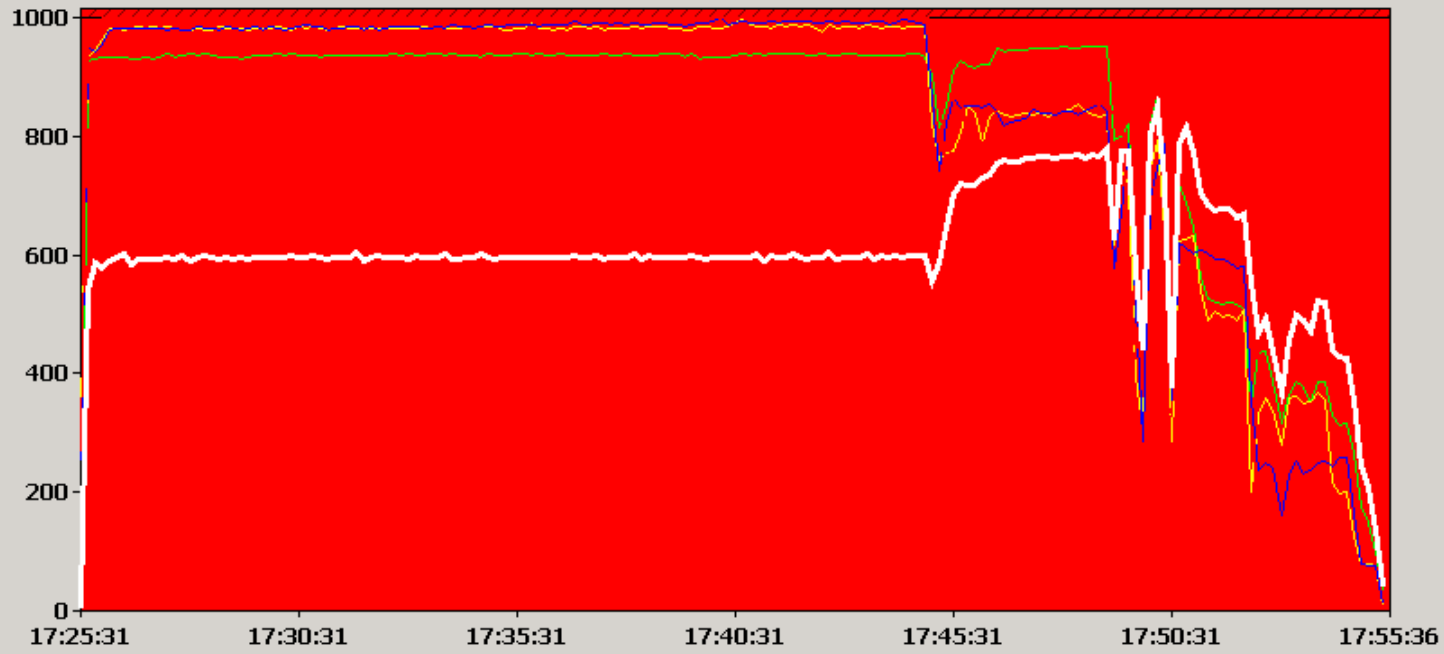
- Use Fast Parse option when possible:
  - Flat file source /destinations
  - Data C
  - Intege
- Reduce v
  - Don't
  - Condit
  - Do not
    - In a
      unr
    - Lea



Result: Packages completed in less then 1800 seconds!

Administrator: etlwr

Loading 1+ Terabyte into SQL Server 2008 on Unisys ES7000/One 64 core server within 30minutes.

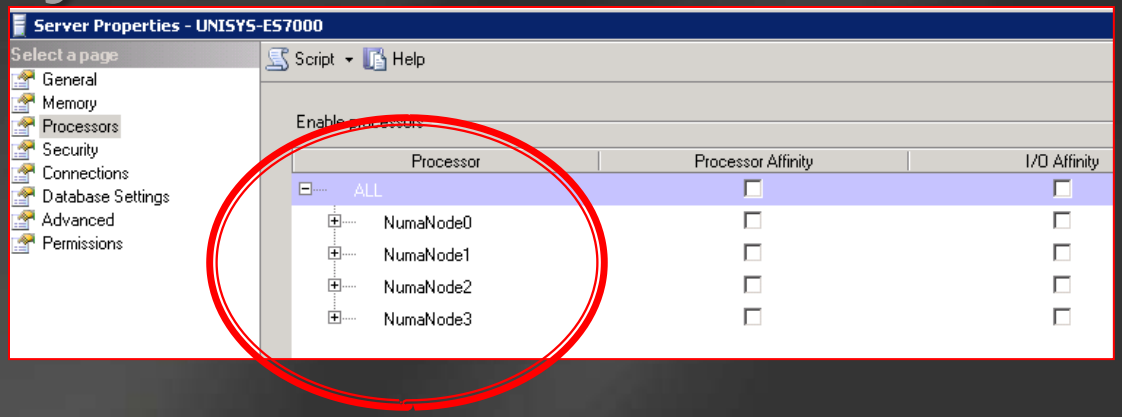


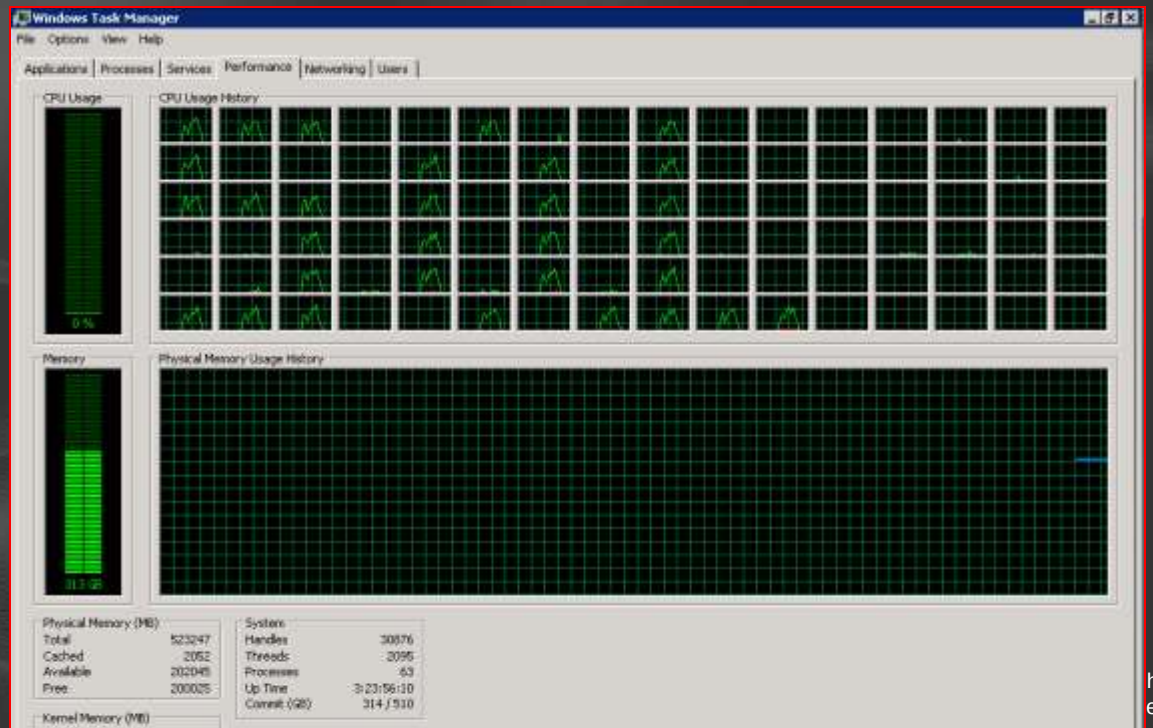
Last	---------	Average	599.986.295	Minimum	9.238.359	Maximum	848.250.009
						Duration	30:05

Sh...	Color	Scale	Counter	Inst...	Parent	Object	Computer
☑		0,0...	Disk Write Bytes/sec	_Total	---	LogicalDisk	\\UNISYS-ES7000
☑		10,0	% Processor Time	_Total	---	Processor	\\UNISYS-ES7000
☑		10,0	% Processor Time	_Total	---	Processor	\\UNISYS-C3
☑		10,0	% Processor Time	_Total	---	Processor	\\UNISYS-C2
☑		10,0	% Processor Time	_Total	---	Processor	\\UNISYS-C1

```
d:" C:\etlwr\log\_latest\800
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1609.2 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1609.18 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1609.17 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1722.77 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1722.75 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1609.13 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1698.24 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1609.38 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1609.06 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1609.01 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1698.07 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1608.89 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1697.34 seconds
2GB+Fiber NoFrg+LgCdPg+64FC+
:Elapsed:  1708.85 seconds
```

How about SQL 2008 R2 CTP on a 96 core Unisys ES7000 server?





+ Enterprise - High Speed DSI Solid State Storage

DSI3020

Solid State Disk
PCIe Flash Storage Expansion Card

Quick Specs

450 GB Flash Memory
700 MB/sec Read Bandwidth
RAID Protection
500 MB/sec Write Bandwidth

DSI3400

Solid State Disk System
Enterprise High Speed Disk

Quick Specs

Up to 512 GB Capacity
4.5 GB /sec Bandwidth
600,000 I/Os per second
Data Mirroring
Multiple Hot Swap Flash Drives
Up to 8 - 4Gb Fibre Channel Parts

DSI3500

Solid State Disk System
Enterprise High Speed Disk

Quick Specs

RAID3 Protection
Up to 2TB Flash Storage
100,000 I/Os per second
Up to 8 - 4Gb Fibre Channel Parts
Hot Swap Architecture
Chipkill Technology

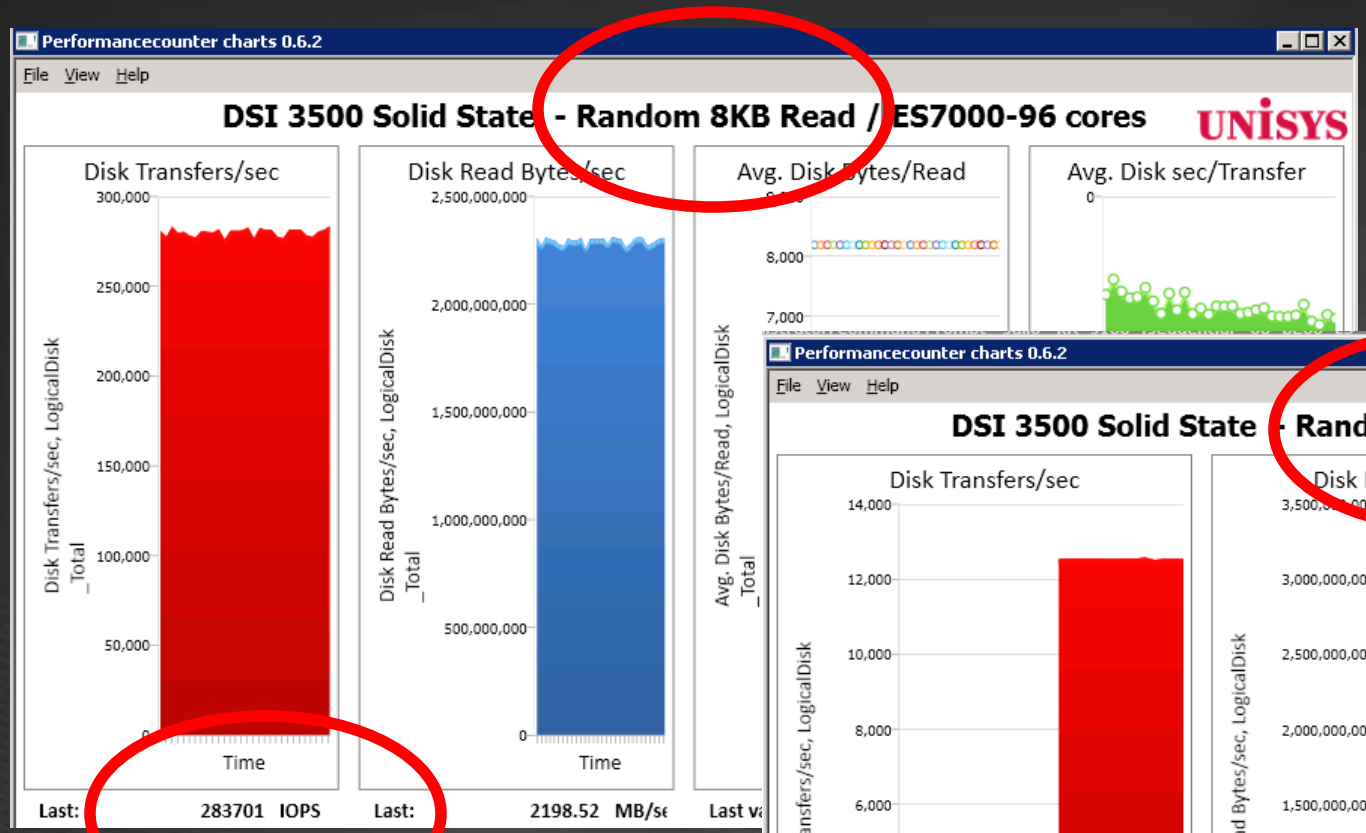
DSI3600

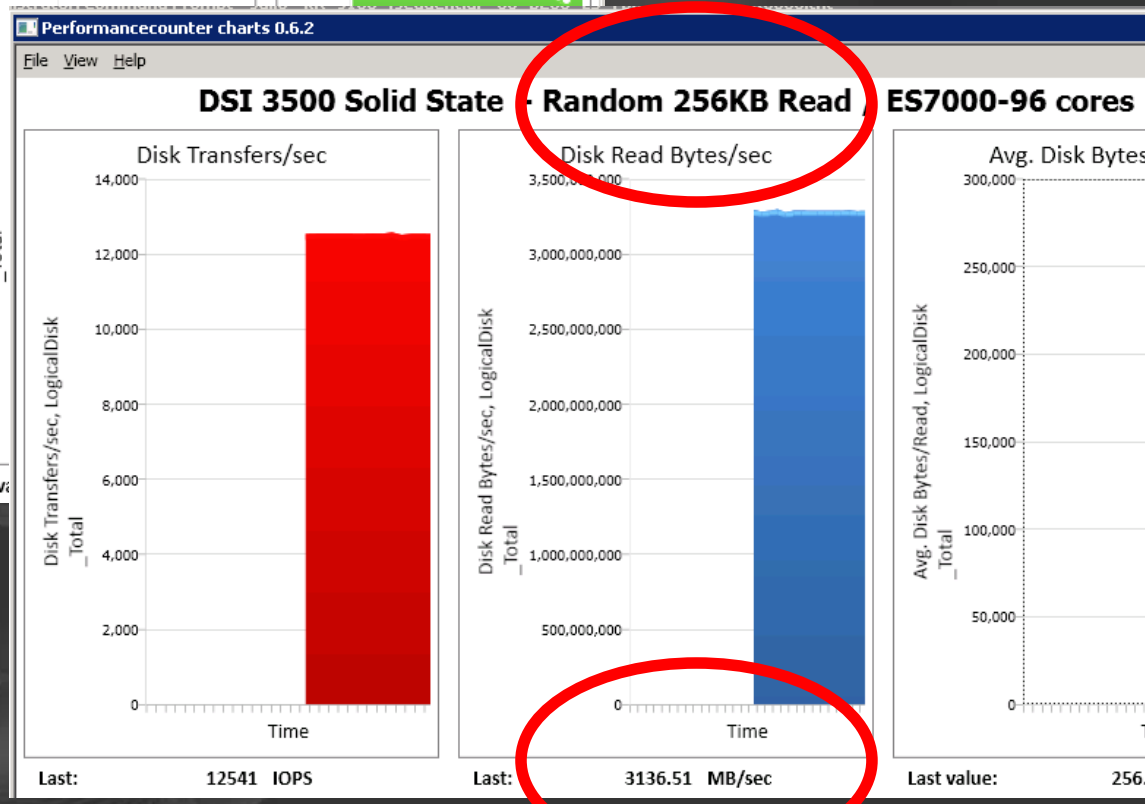
Solid State Disk System
Enterprise High Speed Disk

Quick Specs

Board Level RAID
Up to 5TB Flash Storage
250,000 I/Os per second
Up to 8 - 4Gb Fibre Channel Ports
Hot Swap Architecture
3 GB/sec of bandwidth

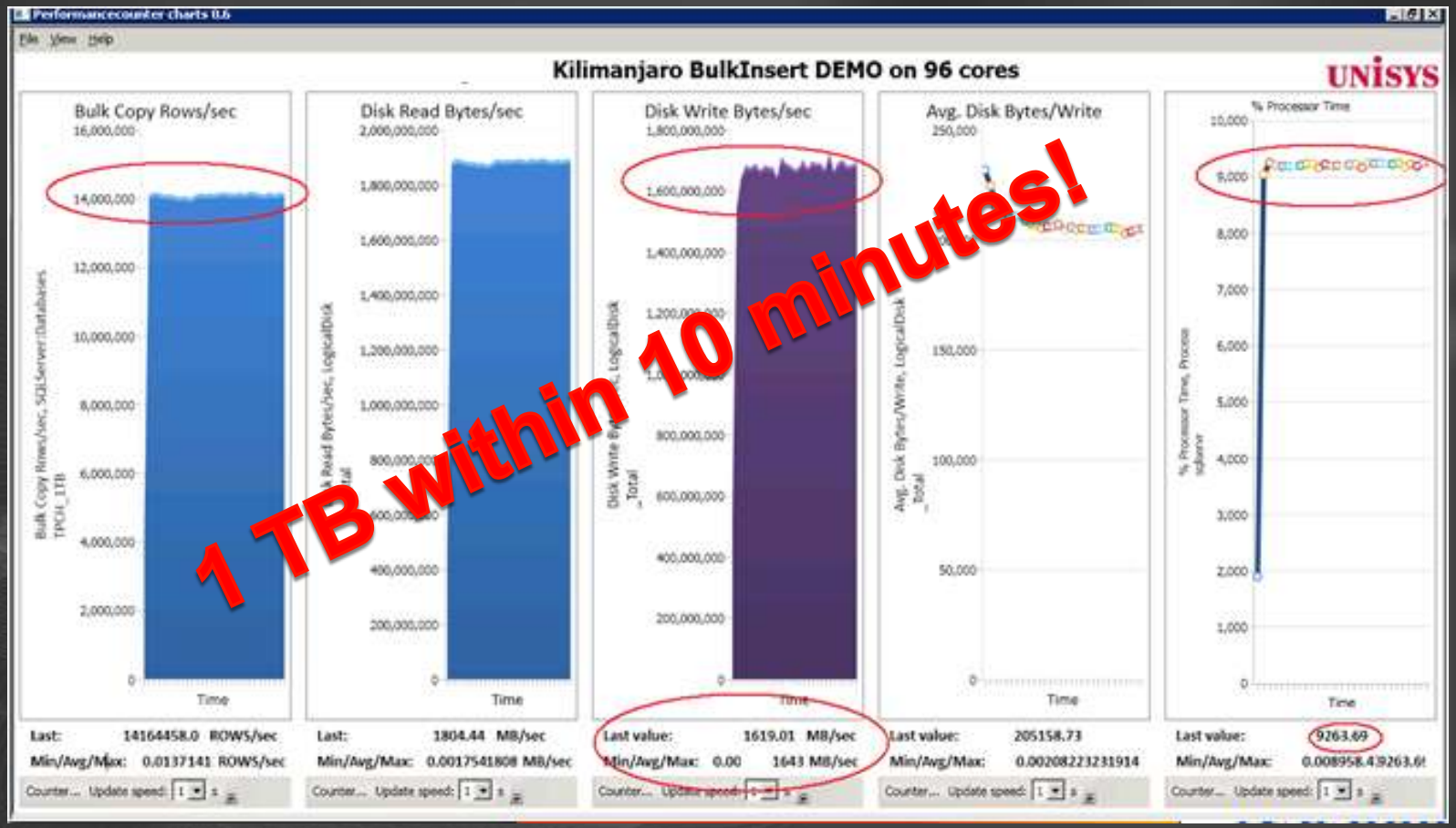
8 Fibers each


Performancecounter charts 0.6.2
File View Help
DSI 3500 Solid State - Random 8KB Read / ES7000-96 cores
UNISYS
Disk Transfers/sec
Disk Transfers/sec, LogicalDisk _Total
Time
Last: 283701 IOPS
Disk Read Bytes/sec
Disk Read Bytes/sec, LogicalDisk _Total
Time
Last: 2198.52 MB/se
Avg. Disk Bytes/Read
Avg. Disk Bytes/Read, LogicalDisk _Total
Avg. Disk sec/Transfer


Performancecounter charts 0.6.2
File View Help
DSI 3500 Solid State - Random 256KB Read / ES7000-96 cores
Disk Transfers/sec
Disk Transfers/sec, LogicalDisk _Total
Time
Last: 12541 IOPS
Disk Read Bytes/sec
Disk Read Bytes/sec, LogicalDisk _Total
Time
Last: 3136.51 MB/sec
Avg. Disk Bytes
Avg. Disk Bytes/Read, LogicalDisk _Total
Last value:

SQL2008 R2 96 Parallel Bulk Inserts

- 96 Core ES7000 with DSI Solid State disks
- Bulk Insert rate increases from
  4 Million rows/sec -> 14+ Million rows/sec!



Typical SQL Wait types while bulk loading

Wait type	Typical cause	Investigate / resolve
LCK_<X>	One process blocking another	Are you using non overlapping input streams?
		Correct TABLOCK used?
		Find the top blocker.
PAGEIOLATCH_<X>	Disk system too slow	Add more disk or tune I/O.
		See “Optimizing I/O and File Layout”.
IMPROV_IO	Text file data drive too slow	Optimize I/O on drive used for input files.
PAGELATCH_UP	Contention on PFS Pages	Make sure disk system is fast enough.
		See “PFS Contention”
		Run with –E flag.
ASYNC_NETWORK_IO	Network cannot keep up	See “Performance Counters Optimizing Network”.
WRITELOG	Transaction log cannot keep up	Verify that you are using minimally logged operations.
		Make sure the transaction log is on fast disk.
OLEDB	Input data too slow	Optimize speed of input data source.
SOS_SCHEDULER_YIELD	Scheduler contention	See “Scheduler Contention”.
ALLOC_FREESPACE_CACHE	Heap allocation contention (only found in sys.dm_os_latch_stats)	Too many threads are inserting into a heap at the same time. Consider partitioning table to get more heaps as insert targets.
PREEMPTY_COM_<X>	Nothing	These waits are normal and expected. Ignore them.

Wrap Up

- By executing this crazy idea and stress testing SQL2008 (R2), we found how to get amazing & constant throughput!
- We can easily load 1 Terabyte of data within 10- 30 minutes…
  - 4 - 14+ million row/sec !
- You have seen with common tools + queries how to identify typical bottlenecks , interpret the basic SQL Waitstats and how to resolve them
- SQL Performance optimizations applied:
  - Sharpen datatypes
  - Use Multiple filegroups
  - Use Soft Numa / Harakiri Stored Procedure
- Using newest CPU types + Solid State Storage increases Bulk insert operation significantly!

Additional Readings

- The Data Loading Performance Guide
  - http://msdn.microsoft.com/en-us/library/dd425070.aspx
- ETL World record
  - http://msdn.microsoft.com/en-us/library/dd537533.aspx

- Dynamic Solutions SSD Storage Solutions
  - DDR: http://www.dynamicsolutions.com/main-menu/dsi3400
  - FLASH: http://www.dynamicsolutions.com/main-menu/dsi3500
- Unisys ES7000 Enterprise Servers
  - http://www.unisys.com/products/enterprise__servers/high_d_end__servers/models/index.htm

Part 2

The data is all loaded, what's next?

Sample data and ETL operations

- ETL operations
  - Denormalize LINEITEM and ORDERS
  - Surrogate key lookup to PART, SUPPLIER and CUSTOMER
    - We will add a new dimension table: CLERK
  - Calculate date surrogate keys (INTEGER)

The Query

400Mill.

```
INSERT Sales  WITH (TABLOCK)
SELECT
/* Surrogate Key lookups */
                ISNULL(P.SK_Part, -1) AS SK_Part
                , ISNULL(C.SK_Customer, -1) AS SK_Customer
                , ISNULL(S.SK_Supplier, -1) AS SK_Supplier
                , ISNULL(CL.SK_Clerk, -1) AS SK_Clerk
/* Dates */
                , CAST(CONVERT(CHAR(8), O.O_ORDERDATE, 112) AS INT) AS SK_OrderDate
                , CAST(CONVERT(CHAR(8), L.L_SHIPDATE, 112) AS INT) AS SK_ShipDate
                , CAST(CONVERT(CHAR(8), L.L_COMMITDATE, 112) AS INT) AS SK_CommitDate
                , CAST(CONVERT(CHAR(8), L.L_RECEIPTDATE, 112) AS INT) AS SK_ReceiptDate
/* Measures */
                , L.L_Quantity AS Quantity
                , L.L_TAX AS Tax
                , L.L_DISCOUNT AS Discount
                , L_EXTENDEDPRICE AS Price
FROM ORDERS O
INNER JOIN LINEITEM L
                ON O.O_ORDERKEY = L.L_ORDERKEY
LEFT JOIN CUSTOMER C
                ON O.O_CUSTKEY = C.C_CUSTKEY
LEFT JOIN PART P
                ON L.L_PARTKEY = P.P_PARTKEY
LEFT JOIN SUPPLIER S
                ON S.S_SUPPKEY = L.L_SUPPKEY
LEFT JOIN CLERK CL
                ON O.O_Clerk = CL.CL_CLERK
```

80Mill.
400Mill.
150Mill.
200Mill.
10Mill.
10000

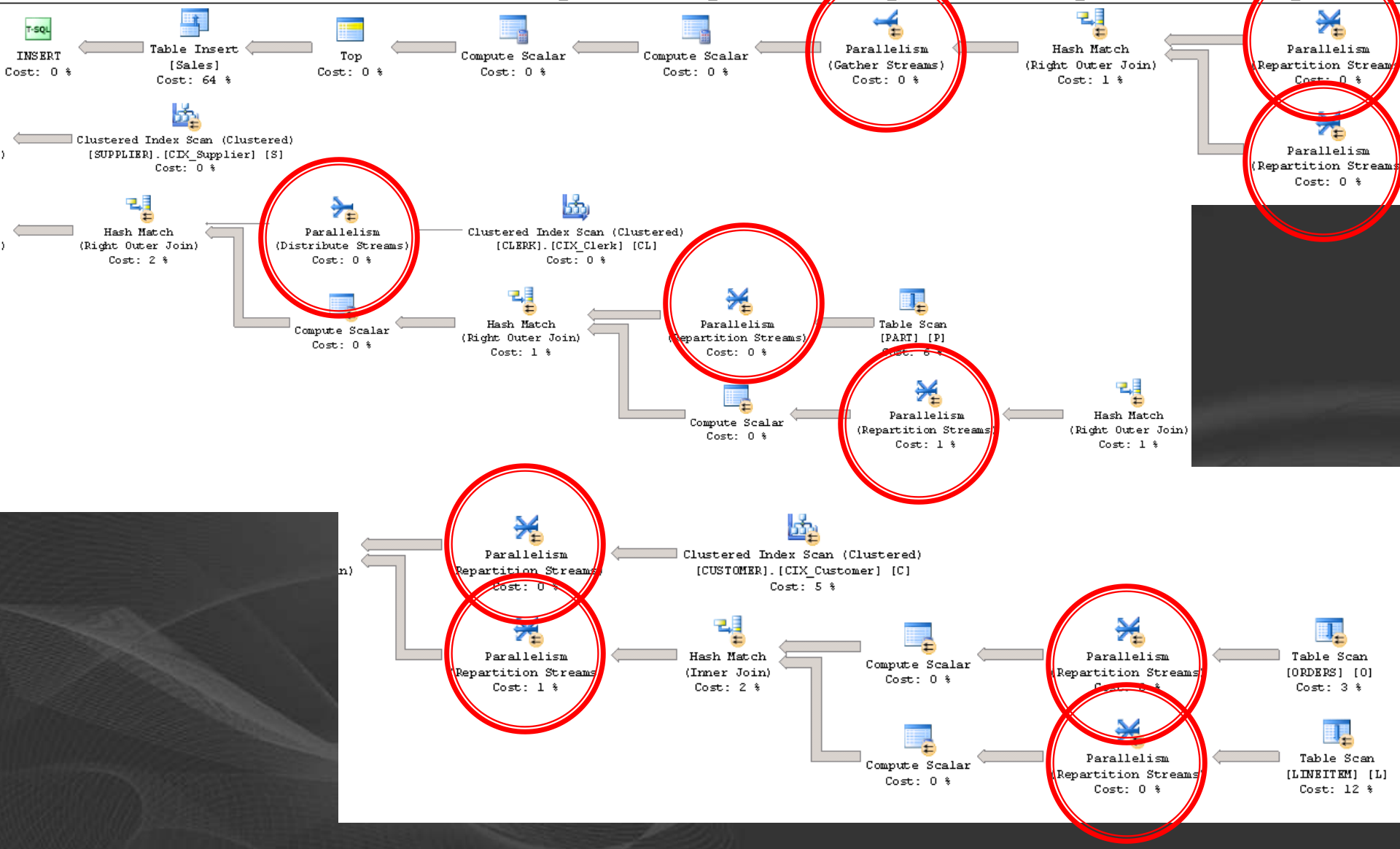
Surrogate key lookup (hash) joins

	SK_Part	SK_Customer	SK_Supplier	SK_Clerk	SK_OrderDate	SK_ShipDate	SK_CommitDate	SK_ReceiptDate	Quantity	Tax	Discount	Price
1	8246592	4422462	591302	11439	19950626	19950828	19950823	19950922	35	0.01	0.01	39955.65
2	8246592	4422462	591302	11439	19950626	19950828	19950823	19950922	35	0.01	0.01	39955.65
3	15216801	7026811	718021	15440	19920729	19920826	19920925	19920911	21	0.08	0.04	36057.84

Query Execution plan

first-EP.sqlplan | . - Activity Monitor | SQLQuery4.sql -...inistrator (63)) | SQLQuery3.sql ...istrator (61))* | SQLQuery2.sql -...istrator (54))* | SQLQuery1.sql -...istrator (60))* | Disk Usage by... UNISYS-ES7000

```
Query 1: Query cost (relative to the batch): 100%
INSERT Sales SELECT /* Surrogate Key lookups */ ISNULL(P.SK_Part, -1) AS SK_Part , ISNULL(C.SK_Customer, -1) AS SK_Customer , ISNULL(S.SK_Supplier, -
```



Step1 : Optimizing Table Scans

Measuring table scan speed

Table Scans - Reading from disk

- What should we measure?
- Tools of the trade :

1) Windows Performance Monitor

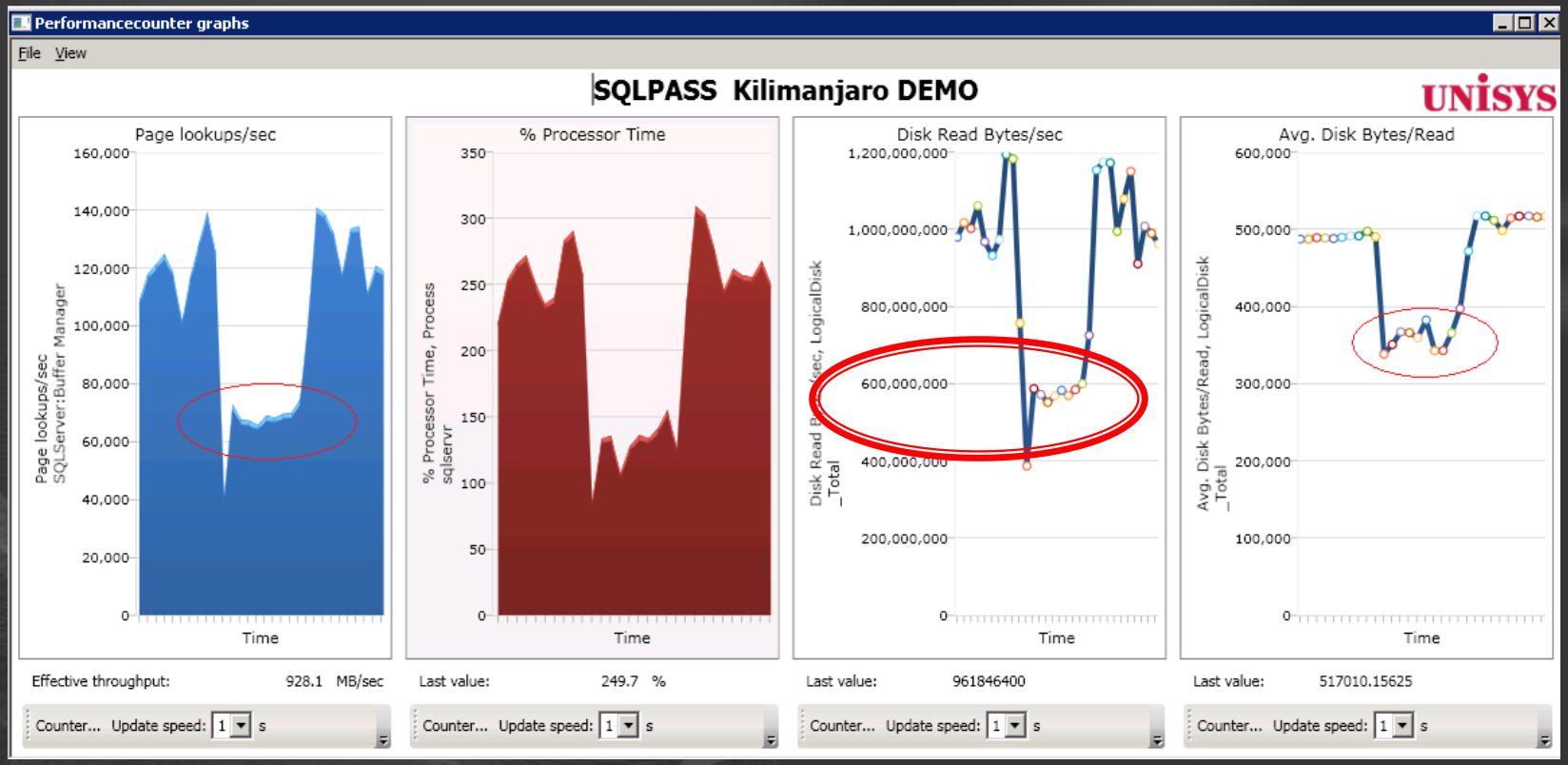
- **Logical Disk : Avg. Disk Bytes/Read**
- **Logical Disk: Disk Read Bytes/sec**
- **SQL Buffer Manager: Page Lookups/sec (8KB pages each)**

2) SQL waitstats :

```
SELECT * FROM sys.dm_os_wait_stats
ORDER BY wait_time_ms DESC
```

1) Non optimized Index: Read block sizes may vary

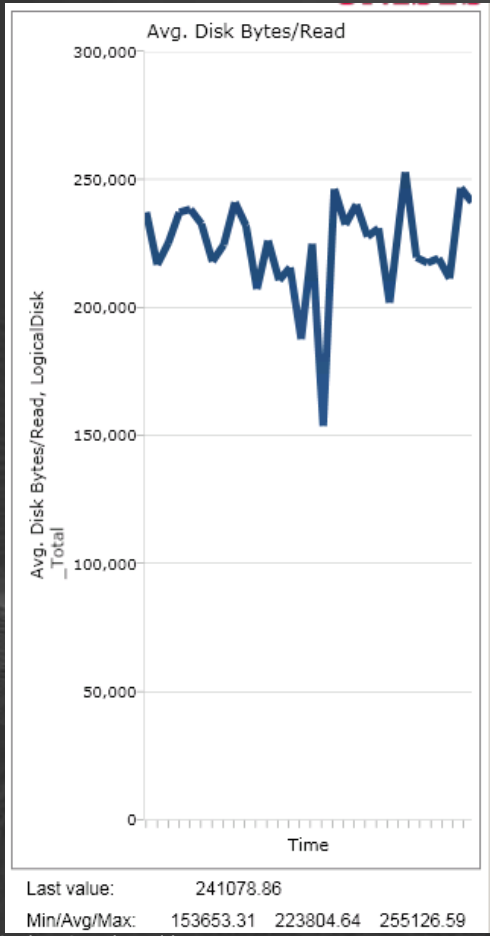
Direct correlation between blocksize and throughput



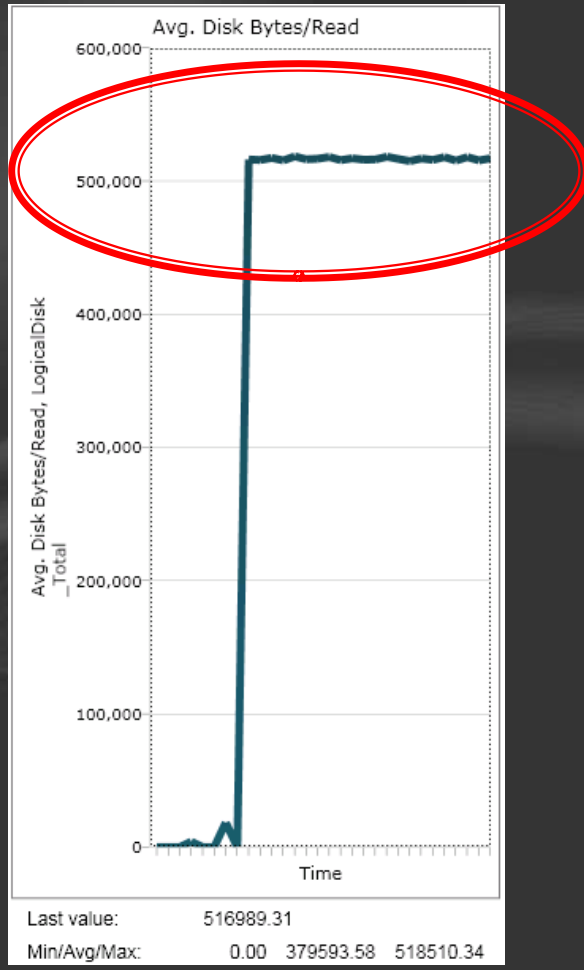
Select COUNT (*) from dbo.LINEITEM_DOP96

1) Optimizing Avg Disk Bytes per Read

- Out of the box: Up to 256 KB per Read

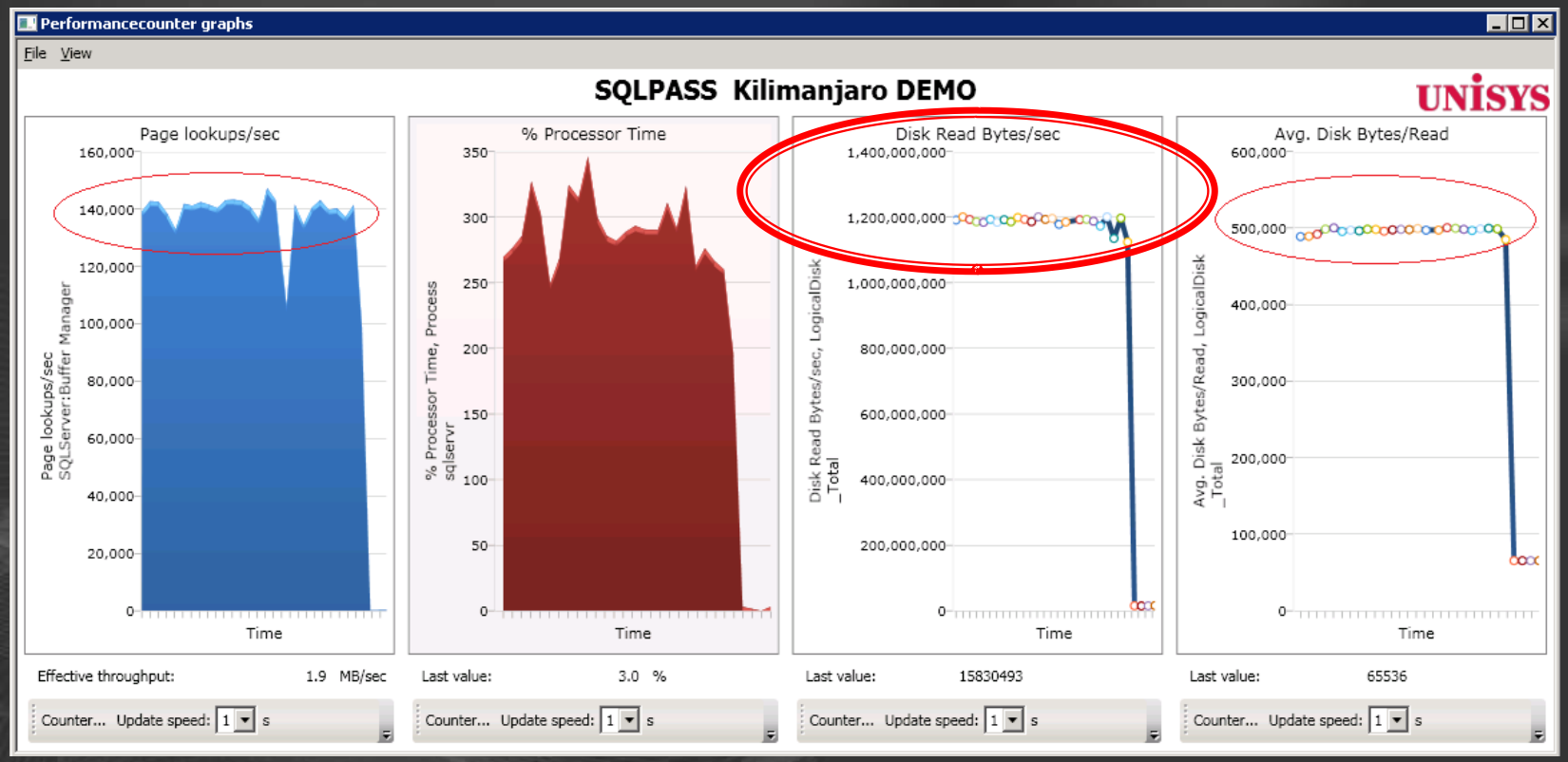


Optimized:
512 KB per Read



1) Index build with DOP1 – Sustained throughput

- ✓ **TIP1:** Build an Index with Option (DOP 1) brings 512KB Read IOPS
- ✓ Tradeoff... Index build time.... (don't worry...;-))



2) Optimizing DISK I/O- Adding Capacity

LineItem table contains 400 Million rows (46 GB)

Run : Select count (*) from LINEITEM

- ✓ Check what SQL is waiting for :

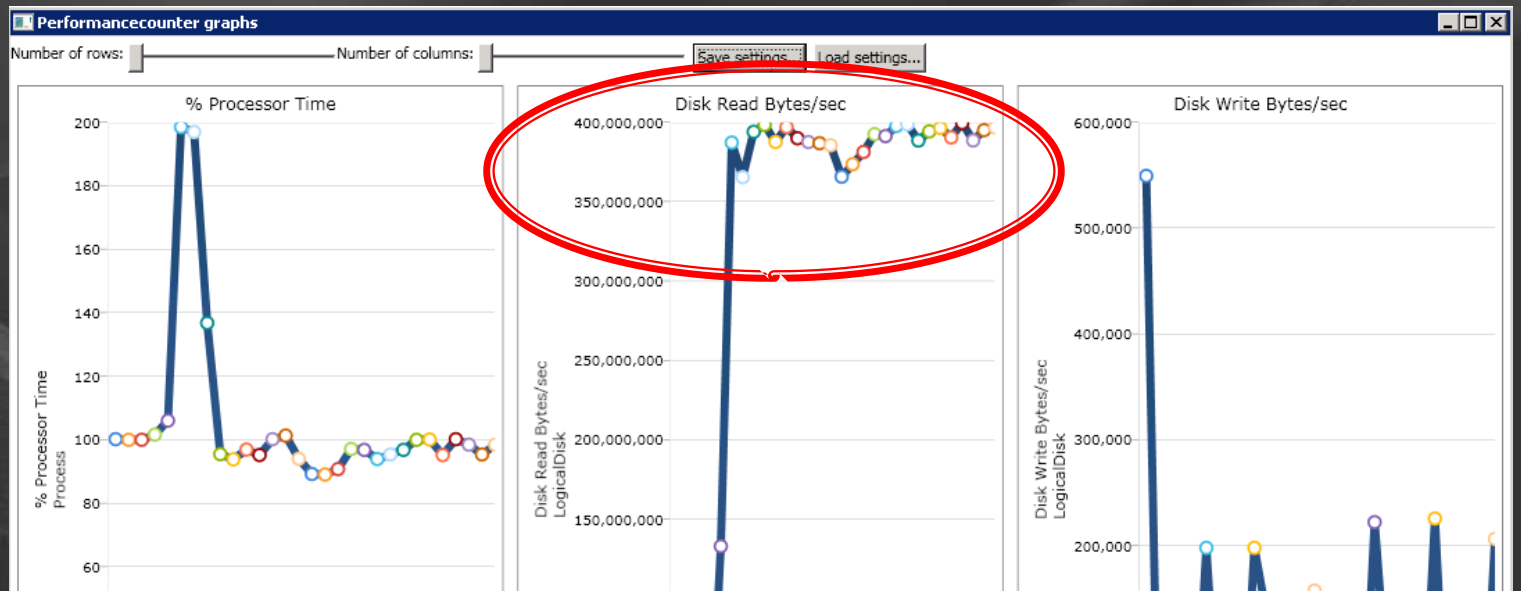
```
SELECT TOP 10 * FROM sys.dm_os_wait_stats
WHERE wait_type NOT IN ('LAZYWRITER_SLEEP','FT_IFTS_RWLOCK'
,'CHECKPOINT_QUEUE','REQUEST_FOR_DEADLOCK_SEARCH', 'BROKER_TASK_STOP'
,'LOGMGR_QUEUE', 'SQLTRACE_BUFFER_FLUSH', 'BROKER_TO_FLUSH', 'SLEEP_TASK', 'SLEEP_BPOOL_FLUSH')
AND wait_type NOT LIKE 'PREEMPTIVE%'
ORDER BY wait_time_ms DESC
```

- ✓
- ✓ If #1 wait is: PAGEIOLATCH_<X>

  Add more database files, LUNs, (Disk cabinets)

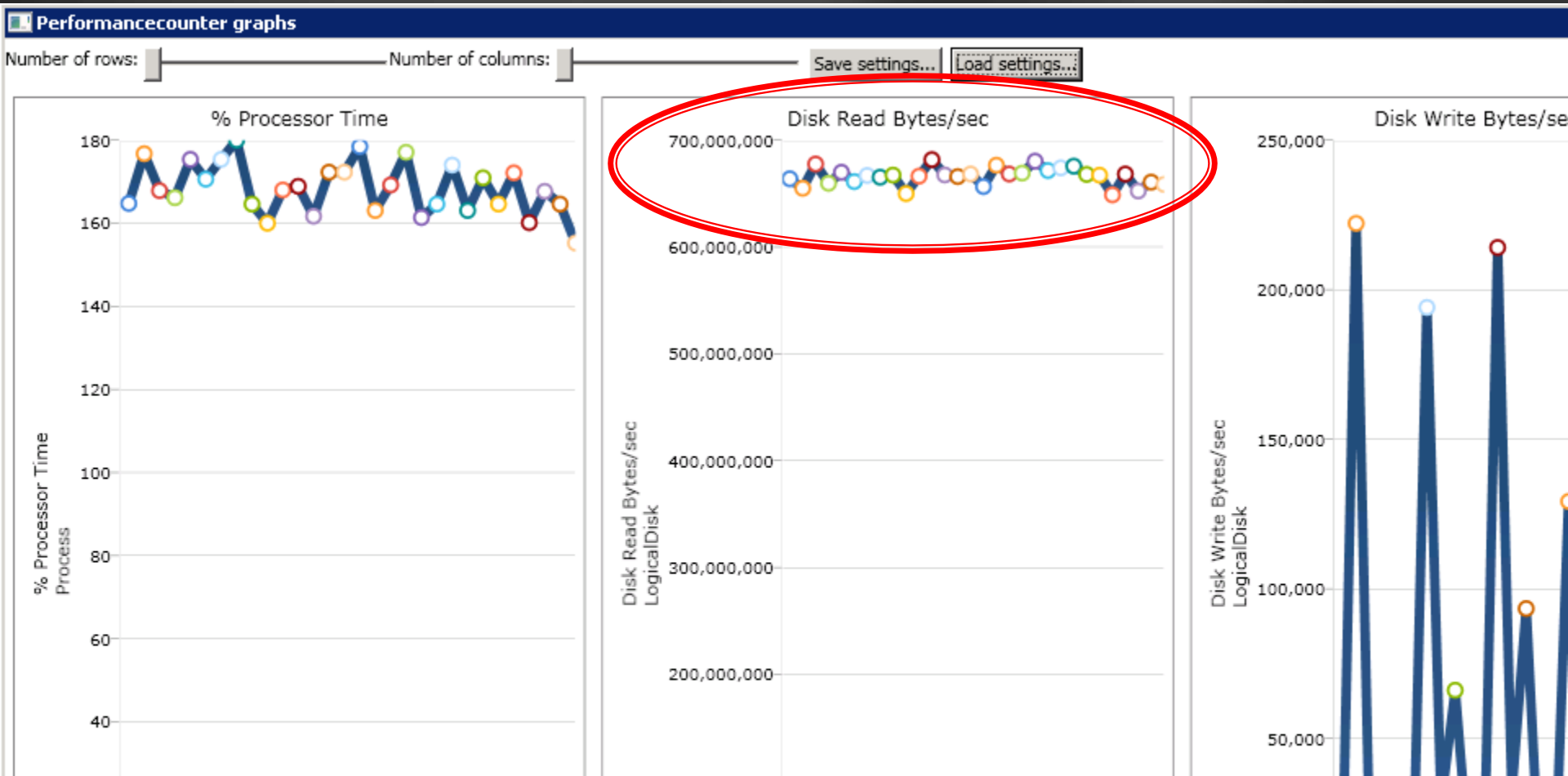
The Basics: Single table scan speed…

- Gettting the maximum out of a core / LUN with 512 KB reads:
  - ✓ With 2 NDF files per Raid-1 Lun: (SPB, no read cache) 250+ MB/sec
  - ✓ Read cache enabled: <u>390+ MB/sec</u>



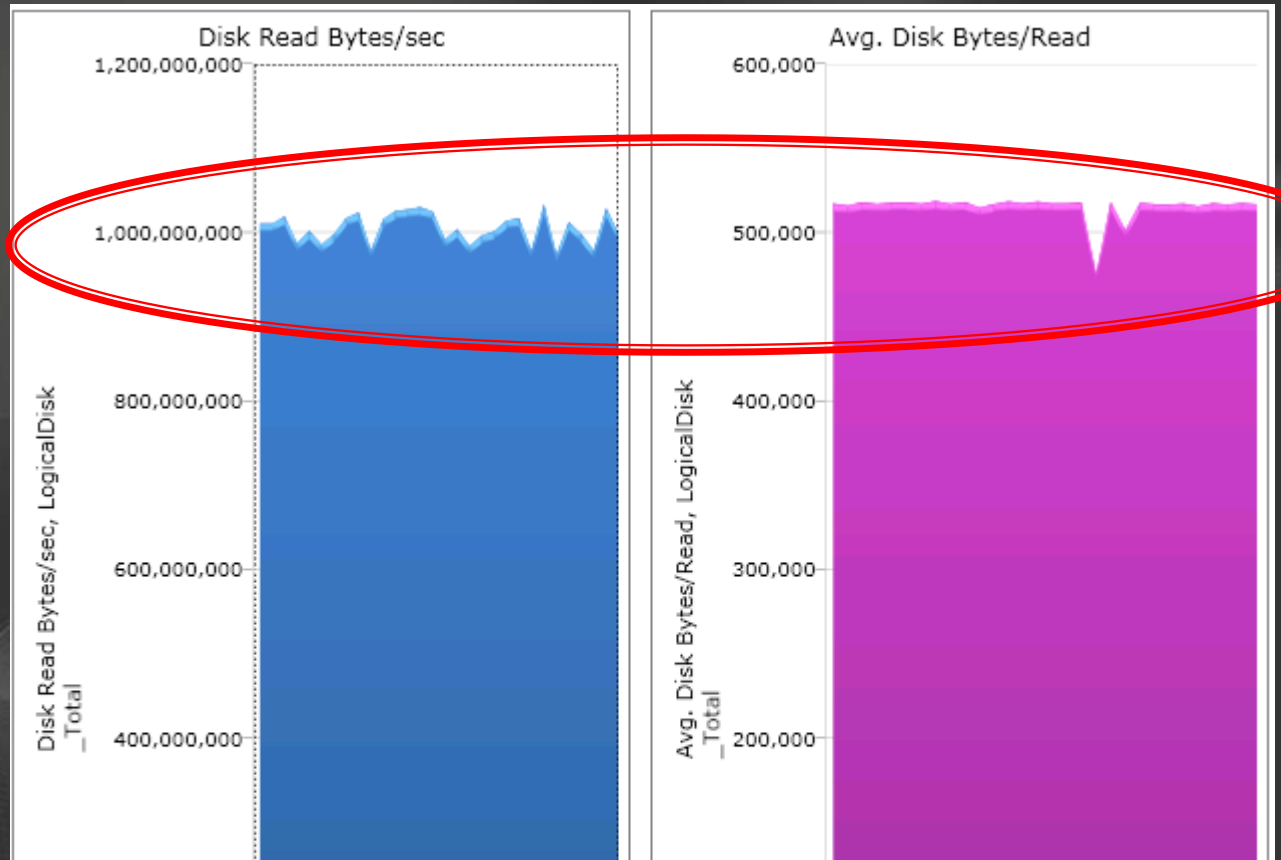
2 luns with 1 file each

- 660 MB read /sec



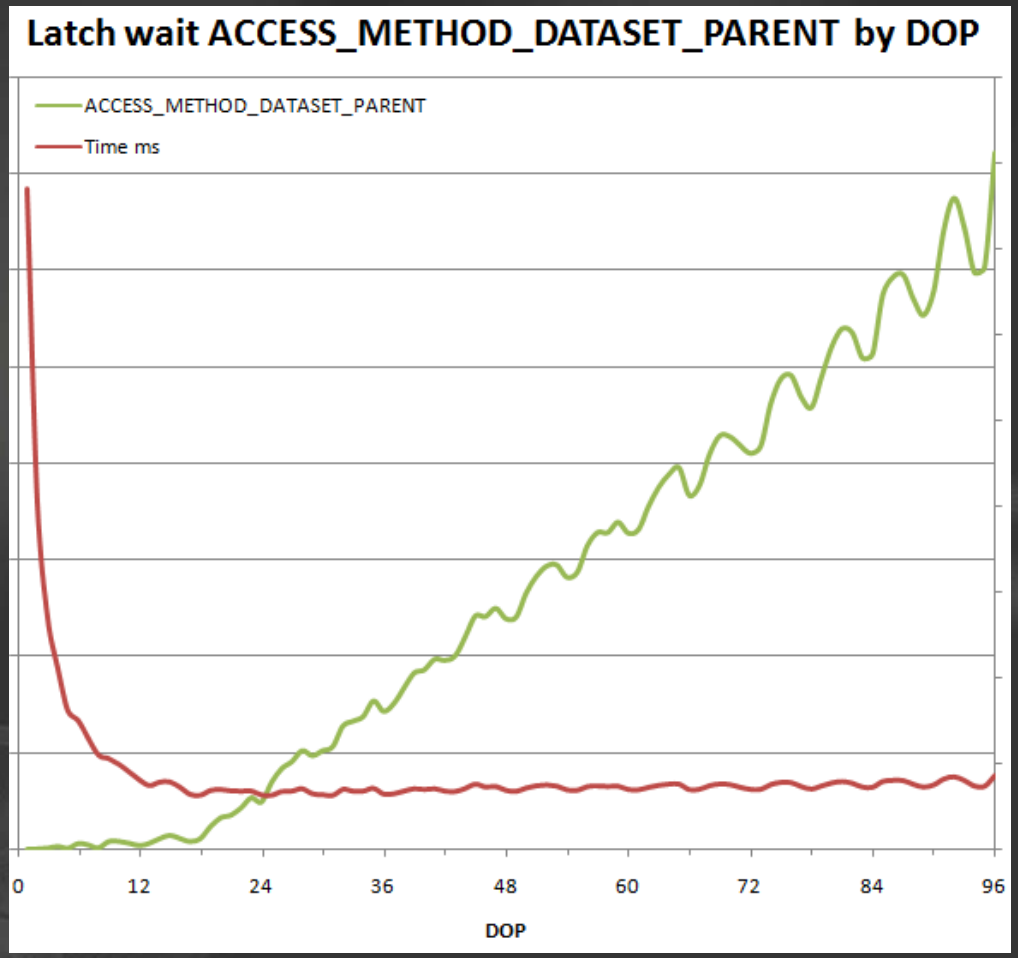
3 luns with 2 files each

- Striping the data across multiple luns
- 1 Filegroup, 6 files



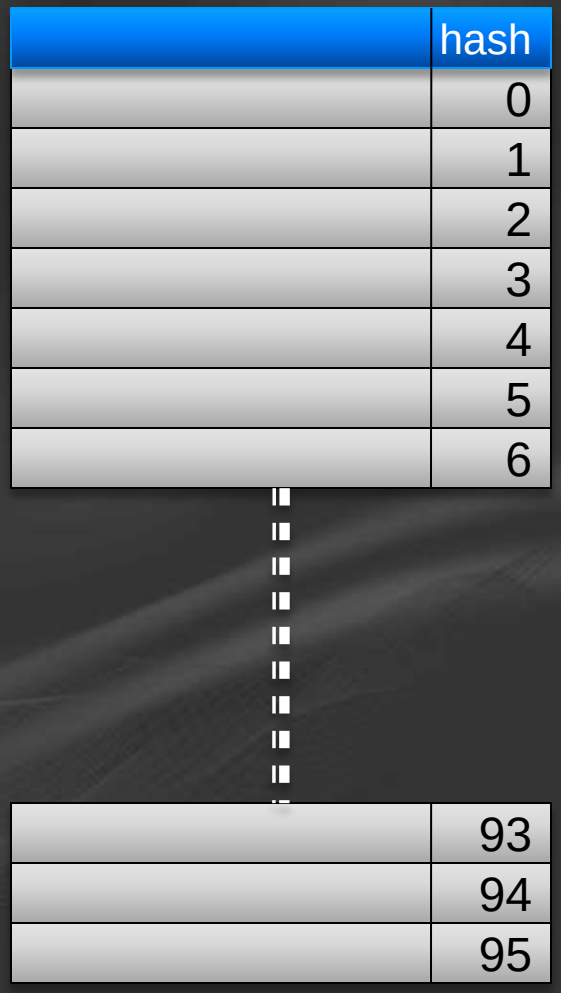
3) Scanning at max speed?

- Bring table to memory
  - Issue another SELECT COUNT(*)
  - How fast are we scanning memory?
- Check the waits, latches and spins:
  - Sys.dm_os_latch_waits show:
  - **ACCESS_METHODS_DATASET_PARENT**
- Hmm?.....



3) Implement Table Hash Partitioning

- Create filegroups hold the partitions
  - Equally balance over LUN using optimal layout (as described earlier)
- Use CREATE PARTITION FUNCTION command
  - Partition the tables into #cores partitions (96 in our case)
- Use CREATE PARTITION SCHEME command
  - Bind partition function to filegroups
- Add hash column to table (tinyint, just one byte per row)
  - Calculate a good hash distribution
  - For example, use hashbytes with modulo
- Rebuild the clustered index on the table on the partition scheme



Hashing

- Optimal :
  Number of partitions == Number of Cores (see later)
- The **ps_hash96()** partition function has 96 partitions
- The hash value is created in such a way that there is a near equal number of rows in all partitions

- ✓ Partion the table by the Hash column
- ✓ Re-Index each partition with DOP1

Pick and test hash function

```
SELECT ABS(binary_checksum(L_ORDERKEY) % 96) AS [Hash]
  , COUNT(*)
FROM [LINEITEM_DOP1]
GROUP BY ABS(binary_checksum(L_ORDERKEY) % 96)
ORDER BY [Hash]
```

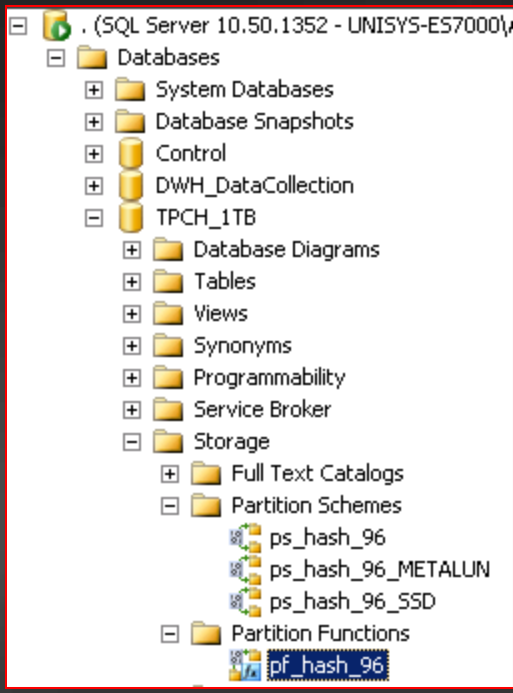
Only 24 buckets filled or uneven fill when hashing on all columns

```
SELECT ABS(hashbytes('MD5', CAST(L_ORDERKEY AS VARCHAR)) % 96) AS [Hash]
FROM [LINEITEM_DOP1]
GROUP BY ABS(hashbytes('MD5', CAST(L_ORDERKEY AS VARCHAR)) % 96)
```

96 almost equal sized partitions

Double check partitioned data for equal distribution afterwards

Select * from sys.partitions where object_name(object_id)='LINEITEM_HASH96key'



Results | Messages

	partition_id	object_id	index_id	partition_number	hobt_id	rows	filestream_filegroup_id	data_compression	data_compression_desc
88	72057594130202624	1029578706	1	88	72057594130202624	4169608	0	0	NONE
89	72057594130268160	1029578706	1	89	72057594130268160	4172864	0	0	NONE
90	72057594130333696	1029578706	1	90	72057594130333696	4174904	0	0	NONE
91	72057594130399232	1029578706	1	91	72057594130399232	4166568	0	0	NONE
92	72057594130464768	1029578706	1	92	72057594130464768	4171936	0	0	NONE
93	72057594130530304	1029578706	1	93	72057594130530304	4156308	0	0	NONE
94	72057594130595840	1029578706	1	94	72057594130595840	4175636	0	0	NONE
95	72057594130661376	1029578706	1	95	72057594130661376	4166728	0	0	NONE
96	72057594130726912	1029578706	1	96	72057594130726912	4172784	0	0	NONE
97	72057594130792448	1029578706	1	97	72057594130792448	4175304	0	0	NONE
98	72057594130857984	1029578706	1	98	72057594130857984	0	0	0	NONE

Kick off Insert

96 partitions

```
CREATE PARTITION FUNCTION [pf_hash_96](tinyint) AS RANGE RIGHT
        FOR VALUES (0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A,
         0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C,
        0x1D, 0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E,
        0x2F, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x40, 0x41,
        0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53,
        0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F, 0x60)
GO

CREATE PARTITION SCHEME [ps_hash_96_SSD] AS PARTITION [pf_hash_96]  TO
([EDGE],
[SSD3500_0], [SSD3500_1], [SSD3500_2], [SSD3500_3], [SSD3500_4], [SSD3500_5], [SSD3500_6], [SSD3500_7],
        [SSD3500_8], [SSD3500_9], [SSD3500_10], [SSD3500_11], [SSD3500_12], [SSD3500_13], [SSD3500_14],
        [SSD3500_15], [SSD3500_16], [SSD3500_17], [SSD3500_18], [SSD3500_19], [SSD3500_20], [SSD3500_21],
        [SSD3500_22], [SSD3500_23], [SSD3500_0], [SSD3500_1], [SSD3500_2], [SSD3500_3], [SSD3500_4],
        [SSD3500_5], [SSD3500_6], [SSD3500_7], [SSD3500_8], [SSD3500_9], [SSD3500_10], [SSD3500_11],
        [SSD3500_12], [SSD3500_13], [SSD3500_14], [SSD3500_15], [SSD3500_16], [SSD3500_17], [SSD3500_18],
        [SSD3500_19], [SSD3500_20], [SSD3500_21], [SSD3500_22], [SSD3500_23], [SSD3500_0], [SSD3500_1],
        [SSD3500_2], [SSD3500_3], [SSD3500_4], [SSD3500_5], [SSD3500_6], [SSD3500_7], [SSD3500_8], [SSD3500_9],
        [SSD3500_10], [SSD3500_11], [SSD3500_12], [SSD3500_13], [SSD3500_14], [SSD3500_15], [SSD3500_16],
        [SSD3500_17], [SSD3500_18], [SSD3500_19], [SSD3500_20], [SSD3500_21], [SSD3500_22], [SSD3500_23],
        [SSD3500_0], [SSD3500_1], [SSD3500_2], [SSD3500_3], [SSD3500_4], [SSD3500_5], [SSD3500_6], [SSD3500_7],
        [SSD3500_8], [SSD3500_9], [SSD3500_10], [SSD3500_11], [SSD3500_12], [SSD3500_13], [SSD3500_14],
        [SSD3500_15], [SSD3500_16], [SSD3500_17], [SSD3500_18], [SSD3500_19], [SSD3500_20], [SSD3500_21],
        [SSD3500_22], [SSD3500_23],
[EDGE])
        GO
```

```
CREATE TABLE [dbo].[LINEITEM_Hash96Key_SSD](
    [L_SHIPDATE] [date] NOT NULL,
    [L_ORDERKEY] [bigint] NOT NULL,
    [L_DISCOUNT] [smallmoney] NOT NULL,
    [L_EXTENDEDPRICE] [money] NOT NULL,
    [L_SUPPKEY] [int] NOT NULL,
    [L_QUANTITY] [smallint] NOT NULL,
    [L_RETURNFLAG] [char](1) NOT NULL,
    [L_PARTKEY] [int] NOT NULL,
    [L_LINESTATUS] [char](1) NOT NULL,
    [L_TAX] [smallmoney] NOT NULL,
    [L_COMMITDATE] [date] NOT NULL,
    [L_RECEIPTDATE] [date] NOT NULL,
    [L_SHIPMODE] [varchar](10) NOT NULL,
    [L_LINENUMBER] [int] NOT NULL,
    [L_SHIPINSTRUCT] [varchar](25) NOT NULL,
    [L_COMMENT] [varchar](44) NOT NULL,
    [hash] [tinyint] NOT NULL
)
GO
```

```
CREATE CLUSTERED INDEX [CIX] ON
    [dbo].[LINEITEM_Hash96Key_SSD]
(
    [L_ORDERKEY] ASC,
    [hash] ASC
    )
WITH  (
SORT_IN_TEMPDB = OFF,
ONLINE = OFF,
ALLOW_ROW_LOCKS  = OFF,
ALLOW_PAGE_LOCKS  = OFF,
MAXDOP=64,   (96?)
FILLFACTOR = 100)

ON ps_hash_96_SSD ([Hash])
GO
```

Hint: To optimally maintain index, you can switch out individual partitions and rebuild them with DOP1 each

Insert statement with the magic hash

```
INSERT INTO LINEITEM_Hash96Key_SSD WITH (TABLOCK)
SELECT [L_SHIPDATE]
      ,[L_ORDERKEY]
      ,[L_DISCOUNT]
      ,[L_EXTENDEDPRICE]
      ,[L_SUPPKEY]
      ,[L_QUANTITY]
      ,[L_RETURNFLAG]
      ,[L_PARTKEY]
      ,[L_LINESTATUS]
      ,[L_TAX]
      ,[L_COMMITDATE]
      ,[L_RECEIPTDATE]
      ,[L_SHIPMODE]
      ,[L_LINENUMBER]
      ,[L_SHIPINSTRUCT]
      ,[L_COMMENT]
   ,ABS (hashbytes ('MD5', CAST (L_ORDERKEY AS VARCHAR )) % 96)
  FROM [TPCH_1TB].[dbo].[LINEITEM_DOP1]
GO
```

Table Scan Optimization Results

- Table partitioning with Hashing speeds up reading from Disk & Memory . Add Hash key for maximum. performance

400 Million Rows / 46 Gbyte	Reading from **Disk** Duration (Sec)	Disk Scanspeed (MB/sec) avg.	Reading from **Memory** Duration (millisec)	Memory Scanspeed (MB/sec)
Out of the BOX : (1File 2spindles RAID-1)	202	250	6812	6800
2 Files (1File 2spindles RAID-1)		370		6800
1FG-48 FILES / 24 LUNS dbo.LINEITEM_DOP96	82	570	6768	6800
Non Partitioned table dbo.LINEITEM_DOP1	74	575	6762	6800
dbo.LINEITEM_Hash96_rev	38	1050	1811	42775
dbo.LINEITEM_Hash96Key_SSD	19	3200	1811	42775

Magic!

3x-10x faster

3-6x faster

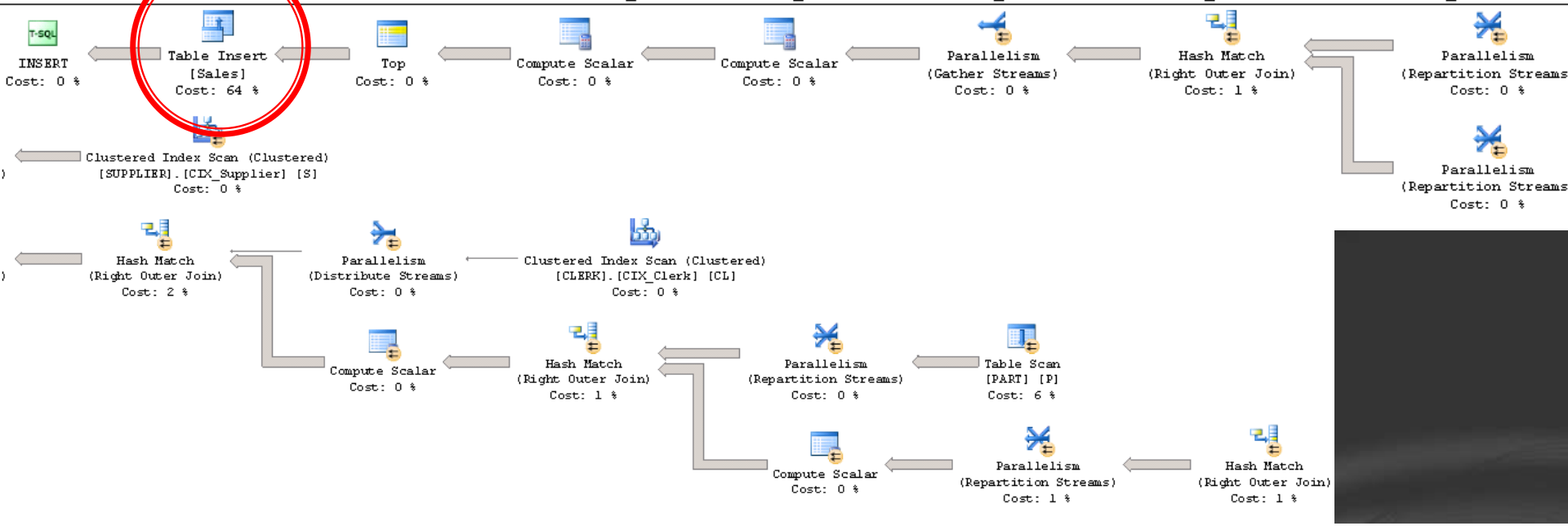
Summary – Optimizing Table Scan

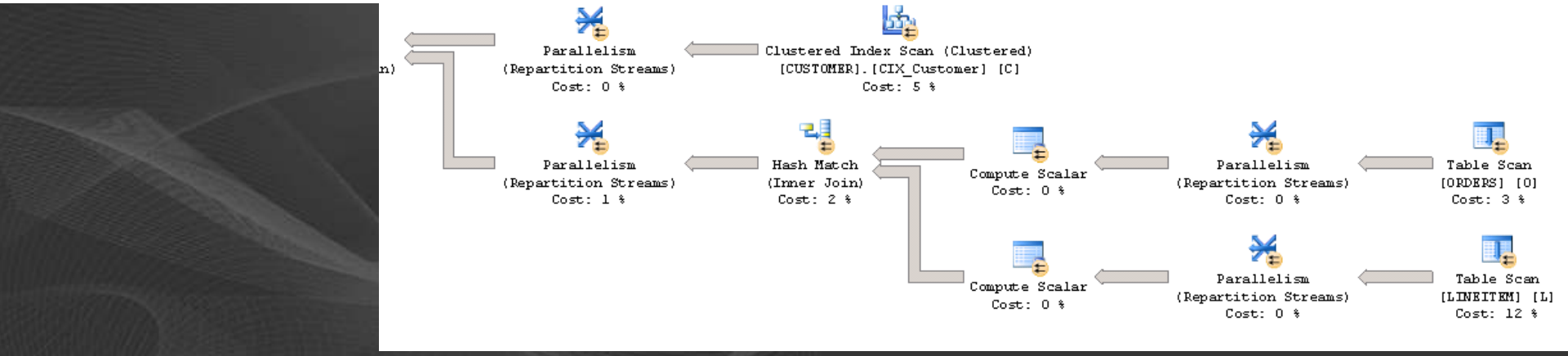
- Find the right LUN layout for your storage system
  - Experiment to find optimal No. of database files for each LUN
  - Add more LUNS
- Use DOP1 to get sustained / large 512KB Read Block sizes
- Use Hash Partitioning to increase memory scan speed
  - Will Balance load across LUNS
  - Avoid ACCESS_METHOD_DATASET_PARENT
- Disable Memory Prefetching in BIOS

Query Execution plan

first-EP.sqlplan | . - Activity Monitor | SQLQuery4.sql -...inistrator (63)) | SQLQuery3.sql ...istrator (61))* | SQLQuery2.sql -...istrator (54))* | SQLQuery1.sql -...istrator (60))* | Disk Usage by... UNISYS-ES7000

```
Query 1: Query cost (relative to the batch): 100%
INSERT Sales SELECT /* Surrogate Key lookups */ ISNULL(P.SK_Part, -1) AS SK_Part , ISNULL(C.SK_Customer, -1) AS SK_Customer , ISNULL(S.SK_Supplier, -
```

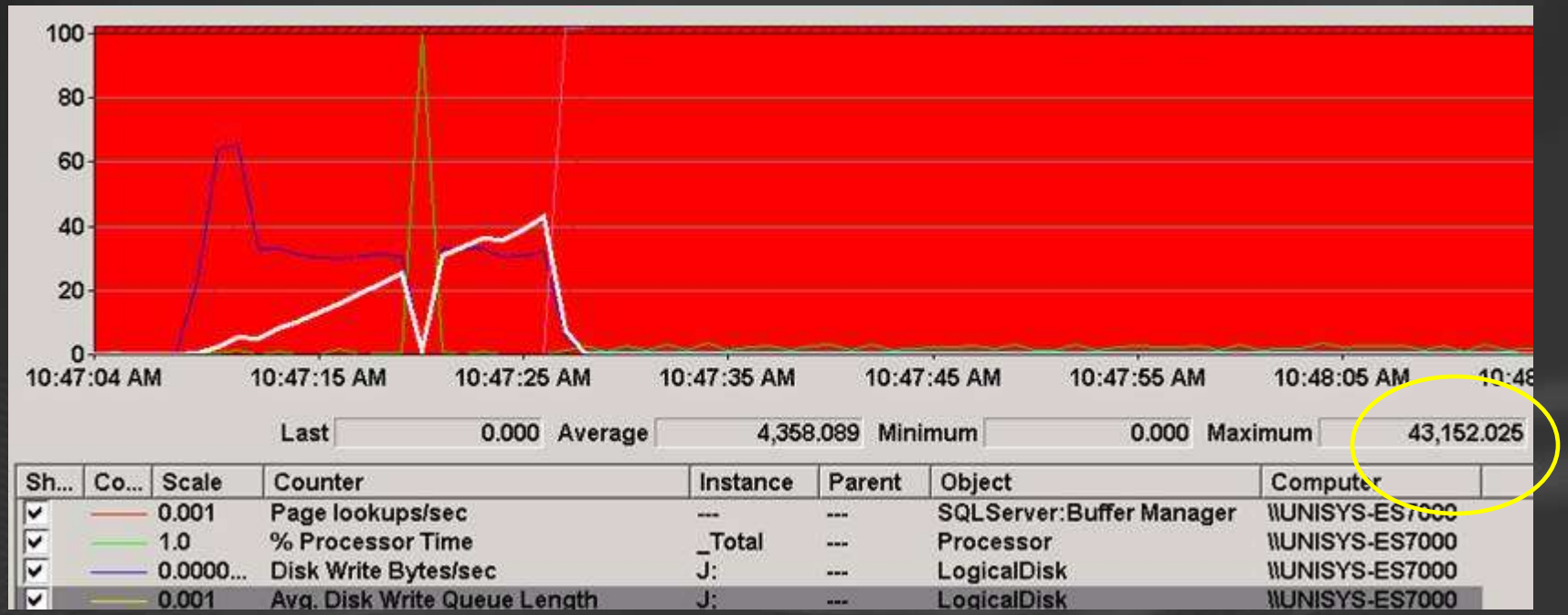


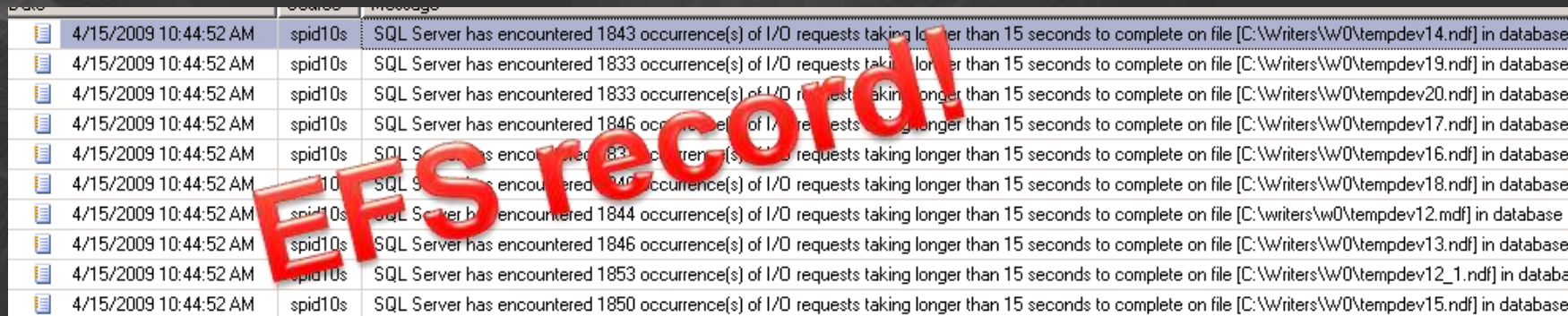


Optimizing The INSERT speed

32 GB RAM

- Tempdb on San / Query won't finish …





24 Core / 32 GB configuration

- With tempdb stored on SSD: avg 200MB/sec writ
- When system is short on memory; tempdb will be used heavily...
- Execution of single 1 Sales Query - with hash

Duration: 30 Minutes

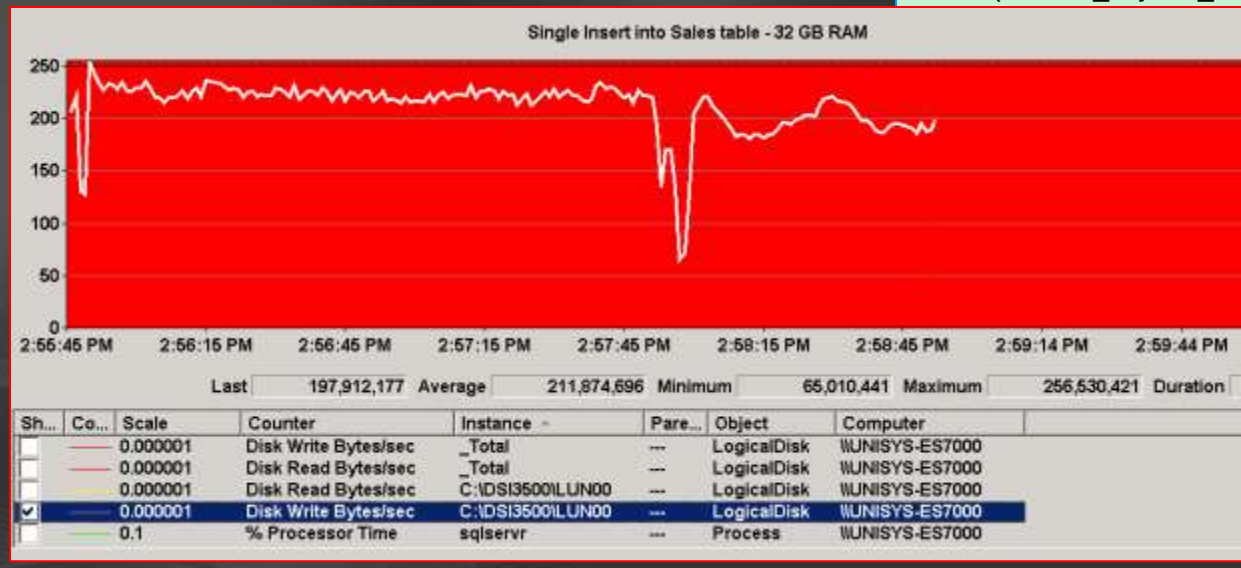
```
--- Tempdb usage by task.s
---
--- This script is provided "
warranties, and confers no
--- Use of included script s
the terms specified at
http://www.microsoft.com/i

SELECT t1.session_id,
(t1.internal_objects_alloc_
task_alloc) as allocated,
(t1.internal_objects_deallo
task_dealloc) as
 deallocated
from sys.dm_db_session_
(select session_id,
   sum(internal_objects_all
```

Results | Messages

	session_id	allocated	deallocated
1	54	31968816	28608384
2	60	1104	1120
3	55	24	16
4	56	0	0

This is 250 GB!



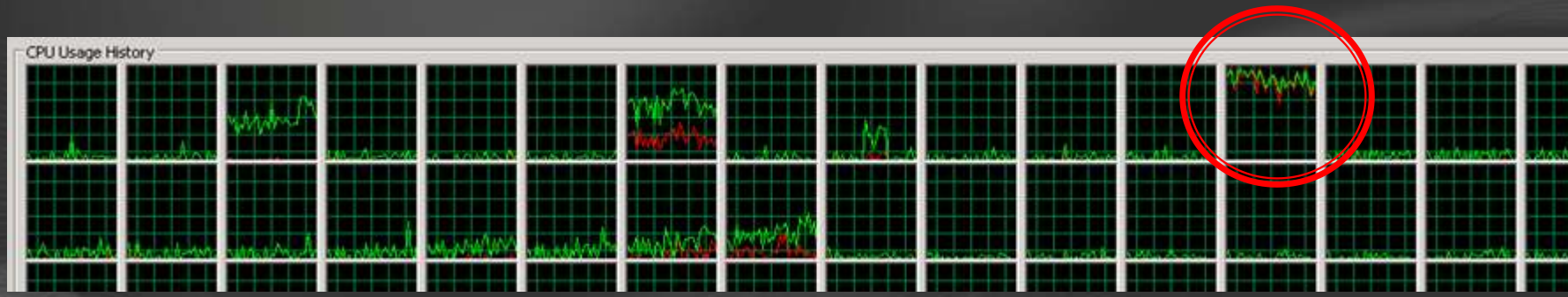
Data File I/O

Database	File Name	MB/sec Read	MB/sec Written	Response Time (ms)
tempdb	C:\DSI3500\LUN00\tempdb\tempdb2.ndf	2.4	2.0	0
tempdb	C:\DSI3500\LUN00\tempdb\tempdb21.ndf	2.5	2.4	0
tempdb	C:\DSI3500\LUN00\tempdb\tempdb22.ndf	2.3	2.4	0
tempdb	C:\DSI3500\LUN00\tempdb\tempdb23.ndf	2.5	2.4	0

Time to run the Query!

- First try with smaller 100 GB dataset

Writing out the Sales data into c:\writers\fact1.ndf with 7 MB/sec ????



What are we waiting for ?

Check with: Activity Monitor/ SELECT * FROM sys.dm_os_wait_stats

. - Activity Monitor | SQLQuery4.sql -...inistrator (63)) | SQLQuery3.sql -...istrator (61))* | SQLQuery2.sql -...istrator (54))*

Processes

Se ID	Us Pr	Login	Datab.	Task State	Comm.	Applic.	Wait Time (ms)	Wait Type	Wait Resource	Bl By	He Bl	Memor Use (KB)	Host Name	Workl Group
62	1	UNISYS-...	TPCH_1...	SUSPEN...	INSERT	Microsoft...	386	CXPACKET	exchangeEvent id...	62		16	UNISYS-...	default
62	1	UNISYS-...	TPCH_1...	SUSPEN...	INSERT	Microsoft...	384	CXPACKET	exchangeEvent id...	62		16	UNISYS-...	default
62	1	UNISYS-...	TPCH_1...	SUSPEN...	INSERT	Microsoft...	391	CXPACKET	exchangeEvent id...	62		16	UNISYS-...	default
62	1	UNISYS-...	TPCH_1...	SUSPEN...	INSERT	Microsoft...	390	CXPACKET	exchangeEvent id...	62		16	UNISYS-...	default
62	1	UNISYS-...	TPCH_1...	SUSPEN...	INSERT	Microsoft...	384	CXPACKET	exchangeEvent id...	62		16	UNISYS-...	default
62	1	UNISYS-...	TPCH_1...	SUSPEN...	INSERT	Microsoft...	392	CXPACKET	exchangeEvent id...	62		16	UNISYS-...	default
62	1	UNISYS-...	TPCH_1...	SUSPEN...	INSERT	Microsoft...	376	CXPACKET	exchangeEvent id...	62		16	UNISYS-...	default
62	1	UNISYS-...	TPCH_1...	SUSPEN...	INSERT	Microsoft...	381	CXPACKET	exchangeEvent id...	62		16	UNISYS-...	default
62	1	UNISYS-	TPCH_1	SUSPEN	INSERT	Microsoft	385	CXPACKET	exchangeEvent id	62		16	UNISYS-	default

Resource Waits

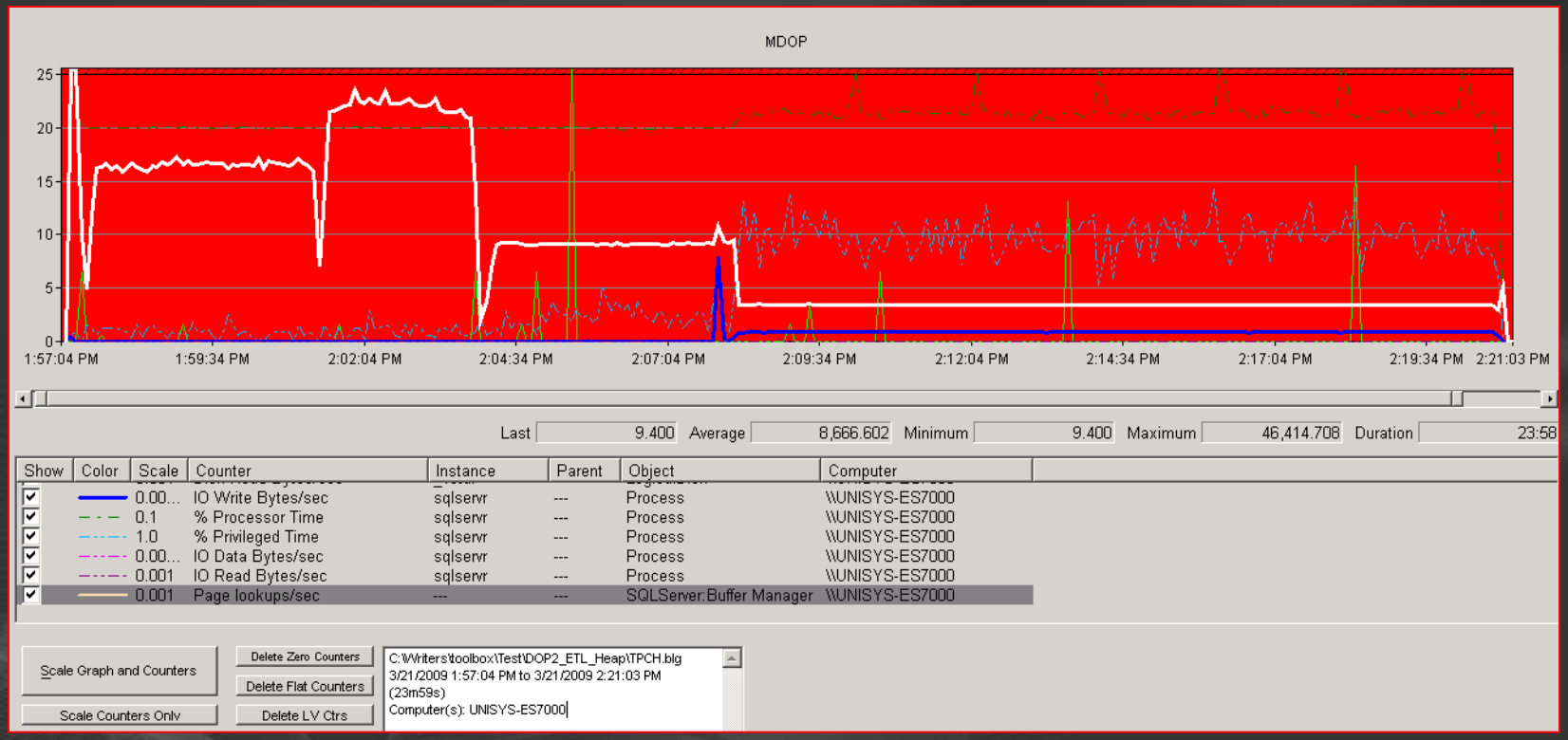
Wait Category	Wait Ti
CPU	
Latch	
Logging	
Memory	
Network I/O	
Other	
Buffer I/O	
Buffer Latch	

Results | Messages

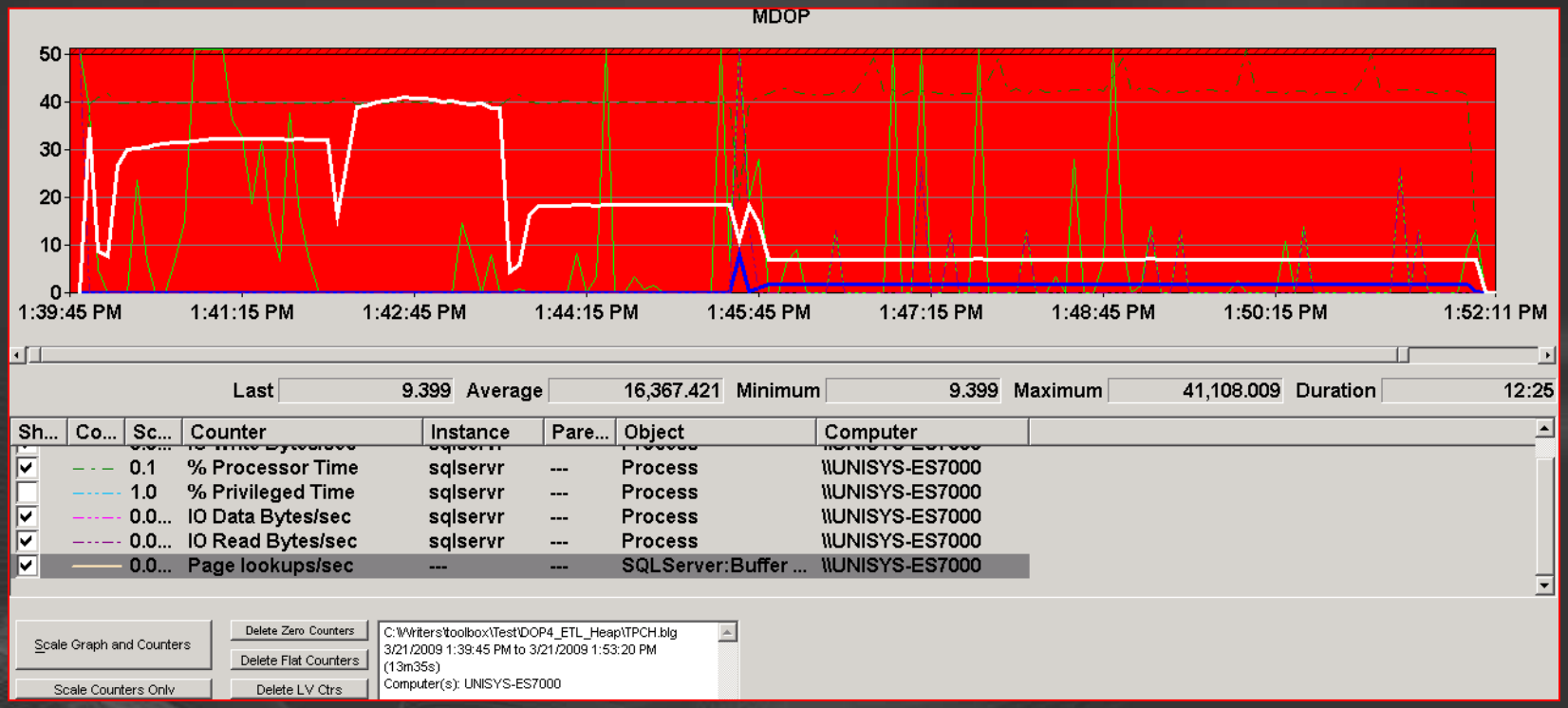
	wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
1	CXPACKET	10434437	64040006	87207	1062128
2	LATCH_EX	98166	1995221	1172	30166
3	LAZYWRITER_SLEEP	9747	1794929	1347079	3426
4	PAGEIOLATCH_SH	9068	1381906	3909	3394
5	SOS_SCHEDULER_YIELD	203359	332617	712	332048
6	XE_TIMER_EVENT	10	300027	30018	300027
7	LOGMGR_QUEUE	45	294413	28165	19
8	CHECKPOINT_QUEUE	10	288620	48007	2
9	SQLTRACE_BUFFER_FLUSH	72	288251	4019	36
10	REQUEST_FOR_DEADLOCK_SEARCH	57	285118	5018	285118
11	BROKER_TO_FLUSH	139	143686	1049	64

Executing the query with MDOP 2

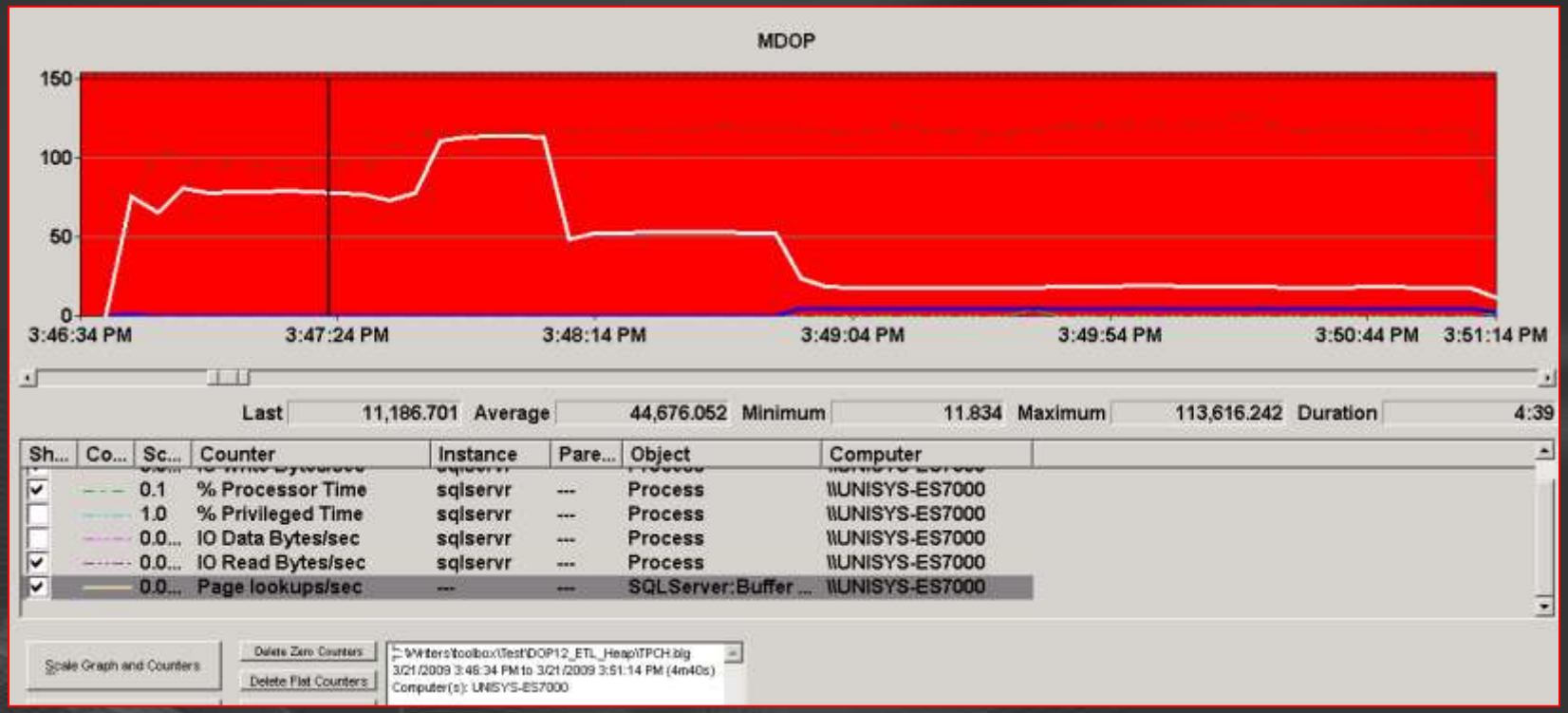
Impact of setting Max Degree of Parallelism:



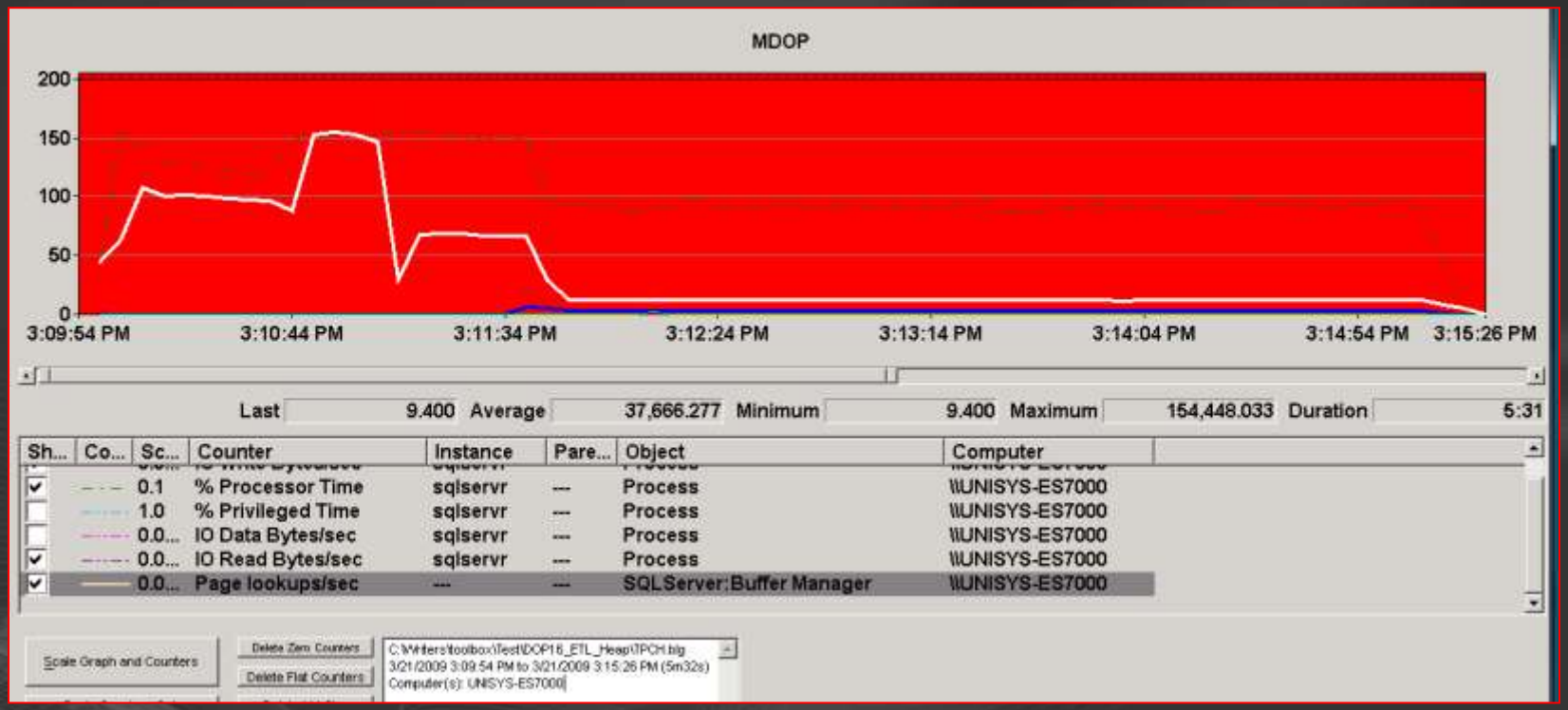
Executing the query with MDOP 4



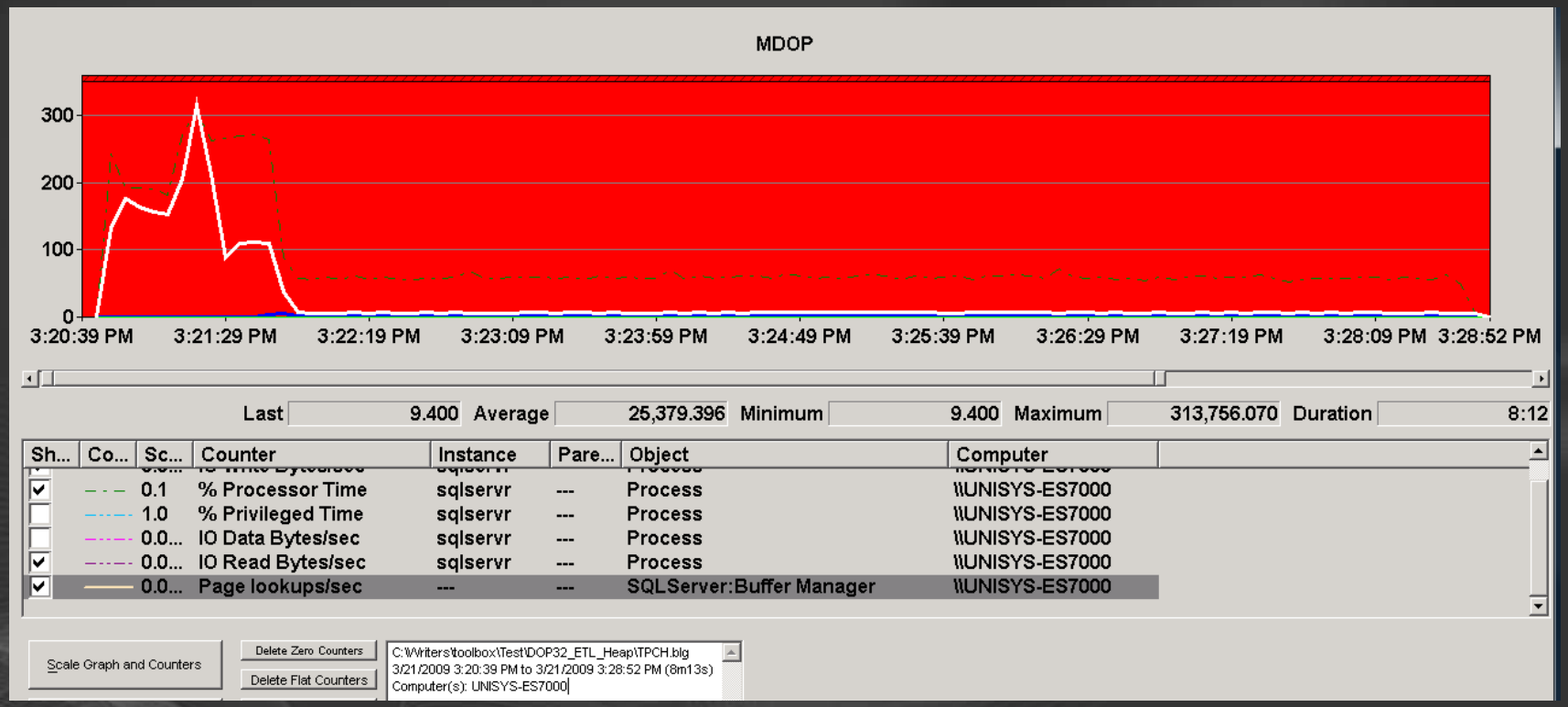
Executing the query with MDOP 12



Executing the query with MDOP 16



Executing the query with MDOP 32



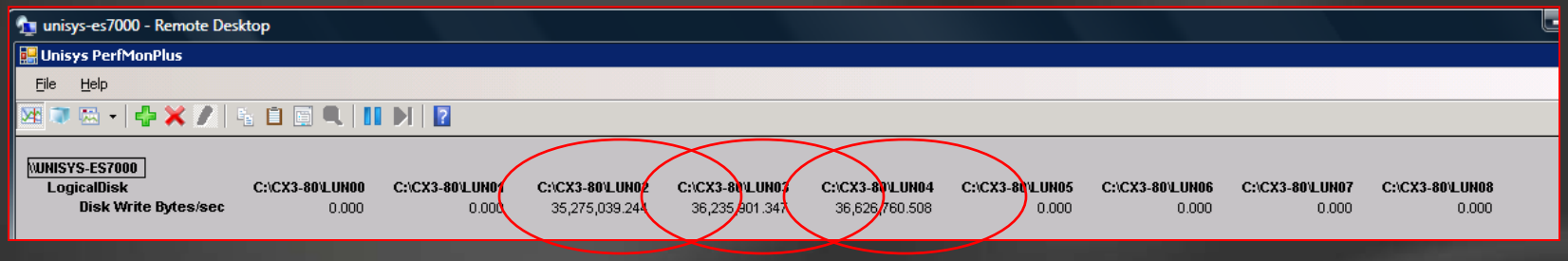
Tuning the slowest part: writing into Sales table

- (MDOP12) gives most effective query execution
- Now time to tune the writing part; with the Brute force approach :

Run multiple instances in parallel

Starting 8 Queries in parallel

- Goal: Let each query write 1/8th of the data
- By default only 3 out of the 8 statements are effectively writing out any data:



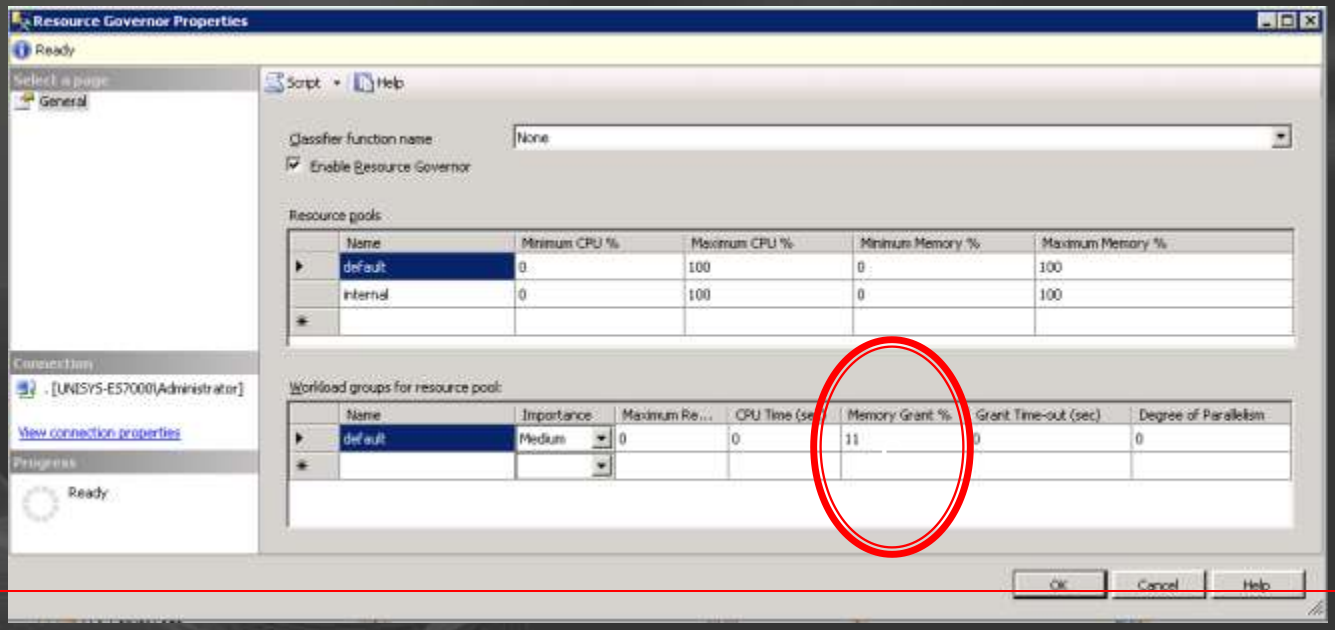
wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms
NULL	126790370	255123774	1465845	63801992
CXPACKET	124460054	240603768	195985	61255269
PAGEIOLATCH_SH	170758	3836430	345	43122
RESOURCE_SEMAPHORE	10	2628736	878041	0
SOS_SCHEDULER_YIELD	1865773	2333791	228	2332614
LATCH_EX	245606	1740525	1283	170074

Use Resource Governor to increase Parallelism

Change "Memory Grant %" Default value of 25% into 10%:

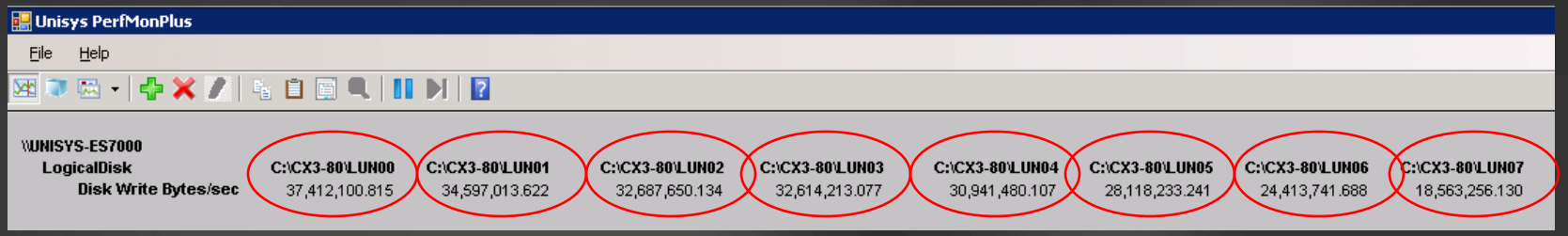
```
ALTER WORKLOAD GROUP [default] WITH
(Request_max_memory_grant_percent=10)

ALTER RESOURCE GOVERNOR
RECONFIGURE;
GO
```

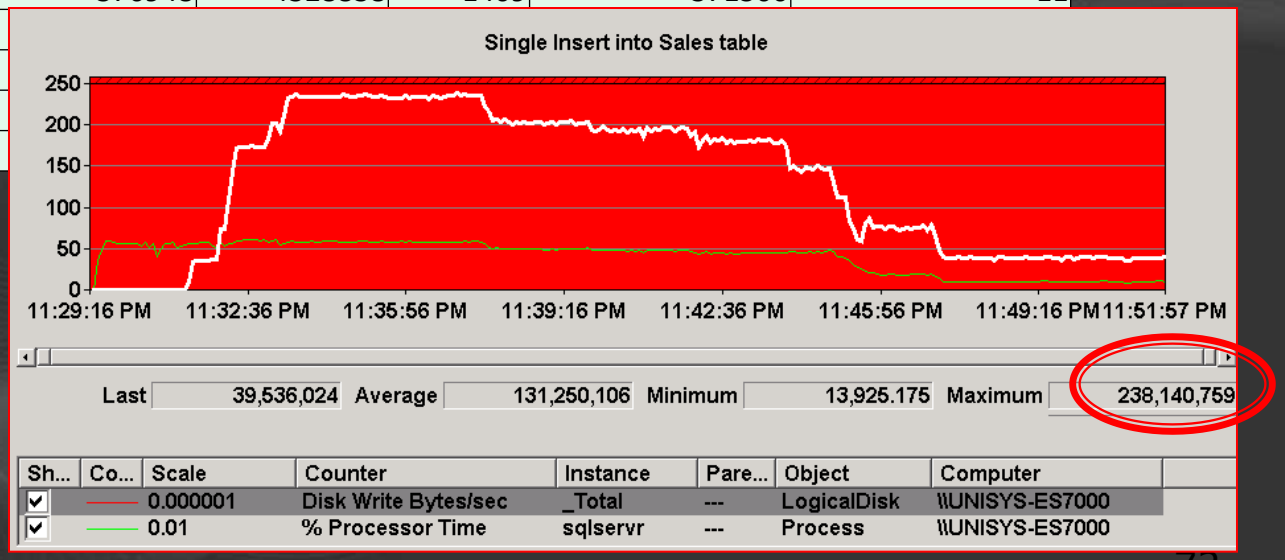


Set RG- Mem. Grant to10%

All 8 queries active



wait_type	waiting_tasks_count	wait_time_ms	max_wait_time_ms	signal_wait_time_ms	Avg_wait_time_ms per waittype req.
NULL	166888823	472763302	382377	107944524	13612
CXPACKET	163998768	458891617	236539	99493911	2
SOS_SCHEDULER_YIELD	2498300	8080208	506	8078462	3
LATCH_EX	376948	4328838	1409	371500	11
LOGMGR_QUEUE					
PAGEIOLATCH_UP					
EXECSYNC					
IO_COMPLETION					



RESOURCE_SEMAPHORE is gone.

8 x 1/8 Inserts to reduce runtime

- Write in parallel portions of the sales output data into separate tables to reduce overall query runtime

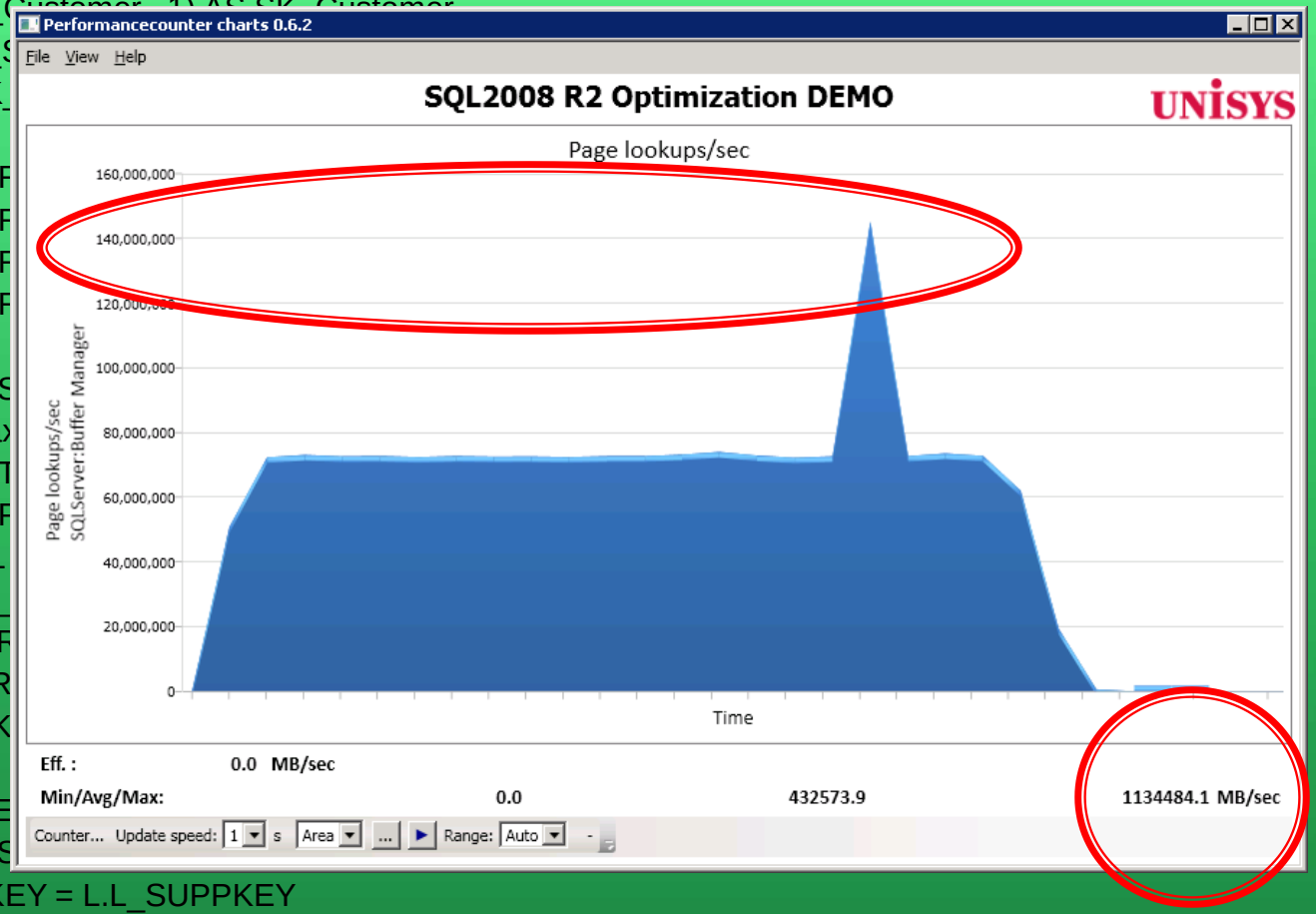
- Use Hash value to define ranges:
  - Sales_00 Range 0 – 16
  - Sales_01 Range 17-32 ...

Summary – Tuning INSERT

- Forcing MAXDOP helps increase throughput
  - Find the best value
- Multiple copies of INSERT statements will speed up ETL speed dramatically
  - Factor 13 in our case
- Use Resource Governor to increase Parallelism
  - Avoid RESOURCE_SEMAPHORE on concurrent workload
  - But: Beware of tempdb pressure

```
SELECT  top 1000000
        /* Surrogate Key lookups */
                ISNULL(P.SK_Part, -1) AS SK_Part
                , ISNULL(C.SK_Customer, -1) AS SK_Customer
                , ISNULL(S.SK_S
                , ISNULL(CL.SK_
        /* Dates */
                , CAST(CONVER
                , CAST(CONVER
                , CAST(CONVER
                , CAST(CONVER
        /* Measures */
                , L.L_Quantity AS
                , L.L_TAX AS Tax
                , L.L_DISCOUNT
                , L_EXTENDEDP
        FROM ORDERS_DOP1
        INNER JOIN LINEITEM_
                ON O.O_ORDER
        LEFT JOIN CUSTOMER
                ON O.O_CUSTK
        LEFT JOIN PART P
                ON L.L_PARTKE
        LEFT JOIN SUPPLIER S
                ON S.S_SUPPKEY = L.L_SUPPKEY
        LEFT JOIN CLERK CL
                ON O.O_Clerk = CL.CL_CLERK

        Select * from PART  where sk_part = '999999'
         OPTION ( MAXDOP 96, LOOP  JOIN )
```



SQL SERVER DAY 2009

Prizes to win

- 1 year hosting by Hostbasket
- 1 Microsoft ARC mouse by Microsoft

Please visit the UNISYS booth

&

join the conversation

SQLUG.BE

www.HenkvanderValk.com

Microsoft UNISYS ORDINA KOHERA QUEST SOFTWARE Smart Systems Management