



white paper

Data Warehouse Performance: Unisys and Microsoft Achieve Record-setting Benchmark

Author:

Sanjay Aggarwal, Unisys Limited (UK)

Henk van der Valk, Unisys Netherlands

Special Thanks for Technical Contributions:

Microsoft Germany

Eric Jacobsen, Microsoft Corp.

Thomas Kejser, Platon (Denmark)

Success

Table of Contents

Executive Summary	3
Our Data Warehouse Proof of Concept Challenge	4
Business Challenge	4
Test Scenario	5
Test Methodology	5
Test Environment Configuration	5
Software Systems	5
Server Configuration	5
Storage Configuration	7
ETL Data Load Test	8
ETL Setup and Optimization	8
ETL Test Result	8
OLAP Cube Processing Test	8
Relational Design Impact on Cube Processing	8
Cube Setup and Optimization	9
Processed Data – Sizes	11
32-Core System Cube Processing Results	12
64-Core System Cube Processing Result	13
Conclusion	16
About the Unisys Performance Centre for Business Intelligence	16
About Unisys	16
About Microsoft	16
About Platon	16

Unisys and Microsoft collaborate to prove Microsoft® SQL Server® 2005 database software supports enterprise-scale data warehouses based on Unisys® ES7000™/one Enterprise Servers. We deliver high levels of performance for all typical processes of building a data warehouse. This white paper summarizes the results of testing SQL Server 2005 running on a single Unisys ES7000/one server with 32 dual-core Intel® Xeon® 7140M–x64 processors.

Executive Summary

Unisys and Microsoft collaborated on a proof of concept to demonstrate that the Microsoft SQL Server 2005 product suite will support even the most demanding enterprise business intelligence initiatives. The proof of concept shows that SQL Server 2005 can deliver the same or better results for a high-performance enterprise-class data warehouse than the more-costly alternatives.

We delivered an optimized data warehouse solution using SQL Server 2005, SQL Server 2005 Integration Services, and SQL Server 2005 Analysis Services on a Unisys ES7000/one Enterprise Server with x64 processors. We conducted a series of tests for extraction, transformation, and load (ETL) operations and online analytical processing (OLAP) cube building. These tests measured throughput, scalability, and performance as the amount of data and execution parallelism were increased. SQL Server 2005 demonstrated the ability to use all available CPUs and memory for both the bulk data load and cube processing testing. In this proof of concept we outperformed the direct competition (Oracle-based solutions) by at least 45 percent. The test results documented in this paper prove that the SQL Server 2005 suite running on the Unisys ES7000 server can take advantage of parallelism and multiple processors to deliver extremely high rates of ETL and OLAP cube processing throughputs. Our team of 12 specialists from Unisys, Microsoft (U.S. and Germany), and Platon implemented this fully-functional data warehouse in less than three weeks.

The testing used the data and business scenario of a large European supermarket chain. The point of sale data spanned over five years. The load process was able to load over 3.5 million rows per second, which equates to loading a billion rows into a data warehouse in less than five minutes. On the cube processing side, we first tested using the 32-core capability on the Unisys ES7000 server. SQL Server 2005 Analysis Services was able to process 1.3 billion rows in approximately 21 minutes, which equates to over 1 million rows per second. We then scaled up to the full 64-core capability of the ES7000 server, using the SQL TCP/IP connection between the SQL Server engine and the Analysis Services. In these conditions the full processing of a 2.1 billion row cube took over 46 minutes. However, when we ran the SQL Server and Analysis Services engines under the same operating system, the processing time was dramatically cut down to just over 19 minutes. This equates to a throughput of over 1.8 million rows processed per second.

These tests are part of a series of collaborative proofs of concepts, head-to-head (benchmark) testing, and engineering. Through this series, Unisys and Microsoft validate optimal performance and establish best practices in system sizing and configuration. As a result, we reduce your risk of implementation and help you achieve high performance levels.

Our Data Warehouse Proof of Concept Challenge

We modeled this proof of concept for a large European supermarket chain that wanted to evaluate and benchmark several common data warehouse solutions. Our solution was based on the SQL Server 2005 capability for building an enterprise-wide data warehouse. Some of the key challenges from the proof of concept include

- final solution: 360 billion rows at the lowest level
- 30+ TB data
- 600+ users
- ad-hoc access through Microsoft PerformancePoint™ Server, Microsoft ProClarity® business analysis software, and Microsoft Office Excel® 2007 spreadsheet software
- initial full and incremental data loads within defined load windows
- full and incremental cube processing within defined load windows
- duration of the proof of concept: only three weeks

This proof of concept needed to demonstrate that the data could be loaded efficiently and then made available

in a format that end users could query in a variety of ways using their tool of choice. For this purpose we tested the two key stages in this process, ETL and the OLAP cube and its reporting capabilities.

ETL is the foundation for business intelligence. It encompasses the processes of collecting, reformatting and cleansing data from multiple sources, and loading it into a data store for analysis or operational purposes. Applications depend on ETL processes to populate these data stores with accurate and timely information. Once the consolidated and cleansed data is available in the repository, it then needs to be summarized and aggregated to quickly answer the most frequent queries. One of the ways to achieve this is by creating an OLAP cube where data is organized for faster querying and some of the values are pre-computed.

Business Challenge

Any successful business intelligence implementation needs to provide fast, efficient ETL, and data available in any formats required by the business users, all within an acceptable time window. Today data warehouses need to support business globalization and 24x7 operations; the processing windows are shrinking and updates have to be completed in much smaller timeframes (i.e., within a few hours versus a few days). Data warehouse performance is a critical success factor for enterprise business intelligence as demonstrated in Figure 1. The blue dotted line represents the information feedback loop, which is a critical component of continuously improving the business performance.

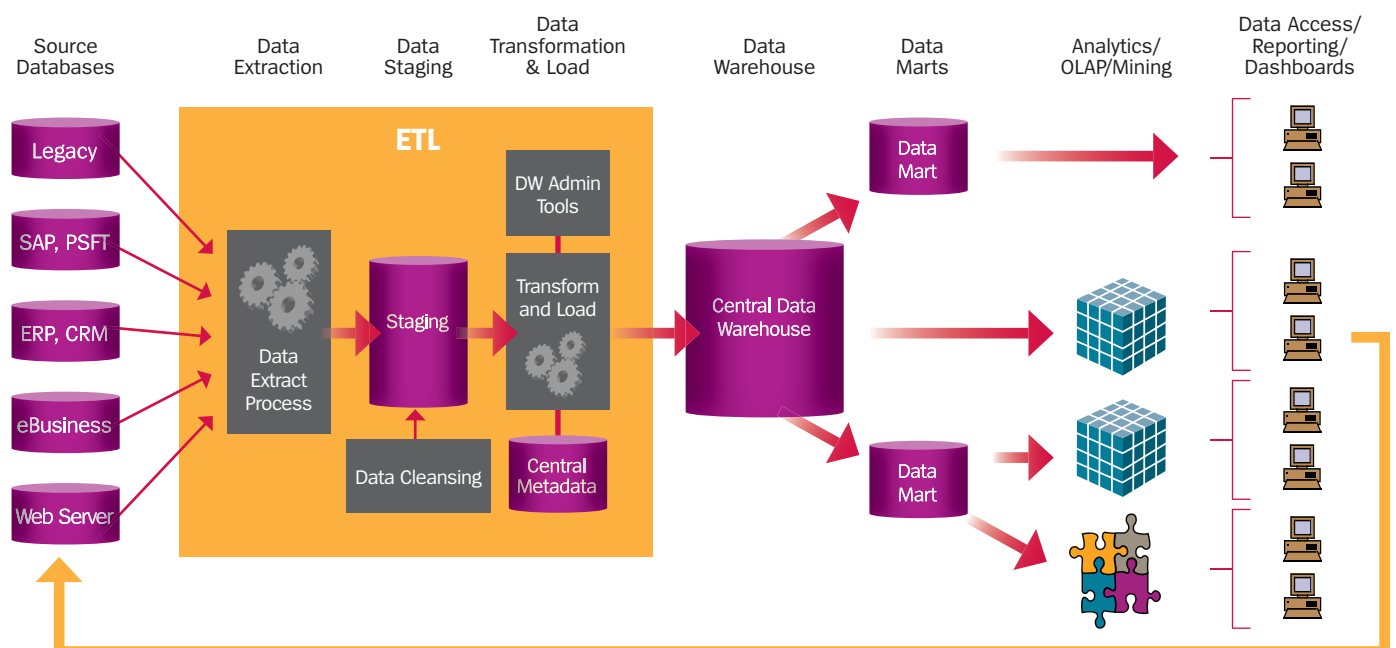


Figure 1: Data warehouse - the foundation of enterprise business intelligence

Test Scenario

Unisys, Microsoft and Platon conducted the testing at the Unisys Performance Centre in Amsterdam, Netherlands. We designed the test methodology around realistic customer scenarios that required creating a new data warehouse and loading it with all the existing data in bulk as well as loading new data incrementally on a daily basis. The test system used the SQL Server 2005 product suite on Unisys ES7000 servers. We designed a single OLAP cube for the customer querying requirement and chose the multidimensional OLAP (MOLAP) storage method for the best performance. We processed the cube by reading the data from the relational data warehouse, summarizing and aggregating it, and then writing it under the multidimensional cube files. The MOLAP storage made the detail level data available in the OLAP cube itself and resulted in a much better query performance. The processing performance and resource utilization was measured at each stage.

The point of sale data consisted of line item order data in a variety of files, some with more than 100 million rows consisting of approximately 20 GB of data each. The data needed to be loaded and optimized for use with OLAP cubes. We faced the challenge of updating the data warehouse in a short batch update window. Based on this scenario, the testing included key aspects of the ETL process: reading data in from flat files, transforming and cleaning it up, formatting it uniformly, and then writing it to the target repository to be exploited.

The three series of scalability tests included

1. loading the point of sale data into a staging database
2. populating the cube
3. validating the cube data by showing several reports and their manageability capabilities

Test Methodology

We masked the data for the testing. We used the T-SQL BULK INSERT command for bulk loading file sources into the data warehouse. We gradually increased the number of commands being run in parallel in line with the available server resources until the throughput peaked.

The cube processing was split into two phases.

- Process data: rows were read from the relational source and compressed
- Process index: aggregates were built to support user queries

We used the standard SQL Server Management Studio and Analysis Services tools to automate the processing, and no additional coding was necessary to achieve the results documented here.

Test Environment Configuration

Software Systems

- Microsoft Windows Server® 2003 R2 Datacenter x64 Edition SP2 operating system
- SQL Server Enterprise Edition V9.0.3175
 - Microsoft High Availability Cluster Service
 - SQL Server and Analysis Services both installed on same ES7000 partition, running under the same OS instance
- EMC Powerpath V 4.5.1.2

Server Configuration

Unisys ES7000/one server with the following configuration was used for all tests.

- 32x dual-core Intel 7140M –x64
- 256 GB RAM
- 8 cells
 - 4 socket & 32 GB per cell.
 - The cells were connected with Unisys proprietary flexbar interconnect which maintained the CMP performance
 - The single physical server was capable of running multiple hard partitions (independent operating systems) within the same cabinet
- 2 partitions (clustered)
 - 7 active cells production / 1 cell passive for failover
 - Up to 28 sockets (56 cores) per partition
- 40 I/O slots
- 4 x 4 gigabit HBAs
 - (16 dual-port HBAs installed)
- 16 dual-port 1GB NICs
- 2 service processors
 - Separate 1U units in the cabinet for server configuration and management

Figure 2 represents the ES7000 test configurations, including the EMC storage subsystem used for the proof of concept.

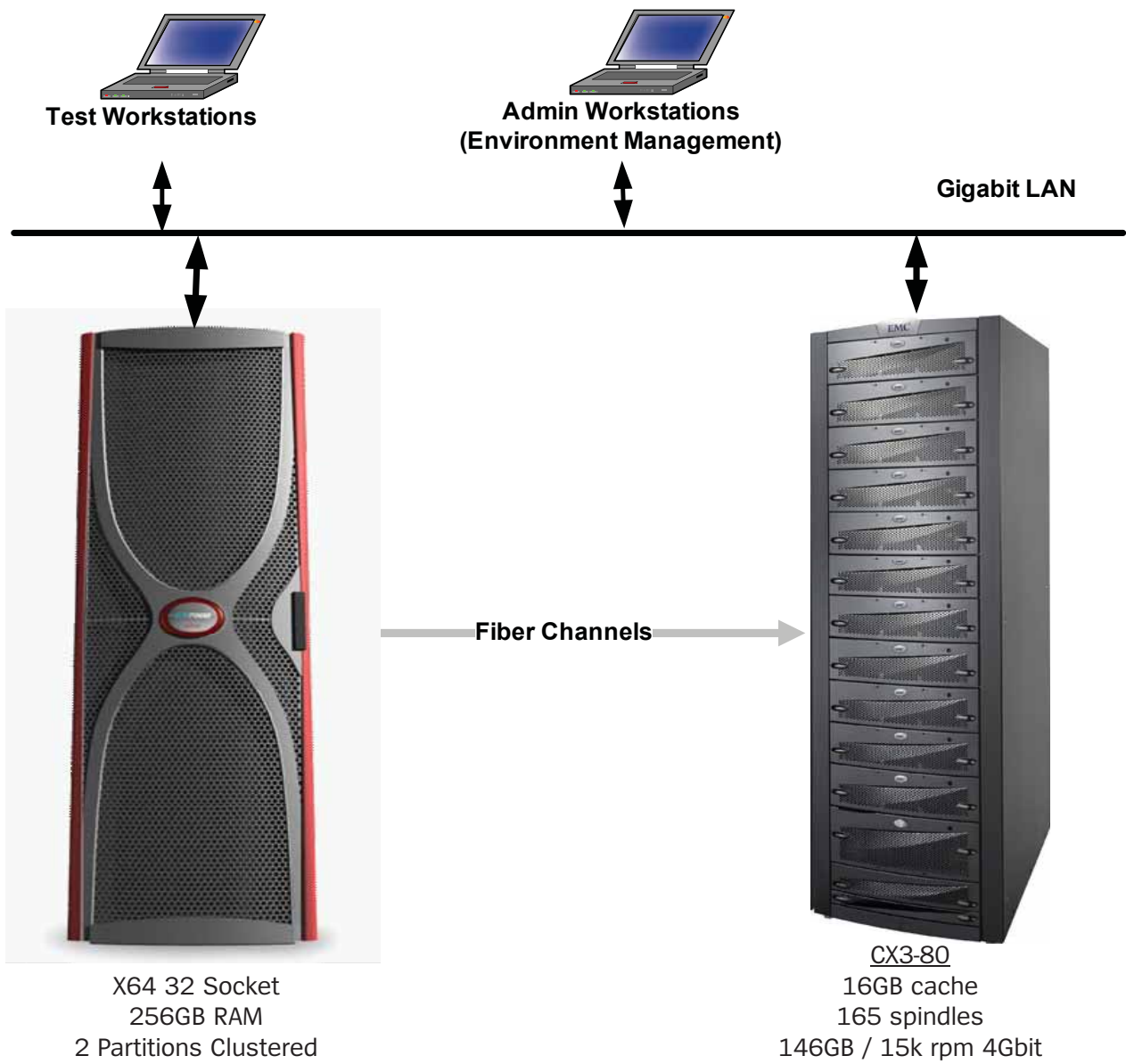


Figure 2: Test systems hardware configurations

Storage Configuration

The storage subsystem used for testing was EMC with following details.

- EMC Clariion CX3-80
- 16 GB cache
- 165 spindles, 146GB / 15k rpm, 4 gigabit each
- LUNS based on 14 spindles RAID-10 RAID groups

Table 1 shows the storage layout and the various RAID levels used for the testing.

RG	LUN	RAID Protection	Size (GB)	Drive Letter	Function
1	10	1+0	600	M	Database
1	11	1+0	335	G	Analysis
2	20	1+0	600	Q	Database
2	21	1+0	335	J	Analysis
3	30	1+0	600	N	Database
3	31	1+0	335	H	Analysis
4	40	1+0	600	R	Database
4	41	1+0	335	I	Analysis
5	50	1+0	600	O	Database
5	51	1+0	250	F	Flat Files (metahead)
6	60	1+0	600	S	Database
6	61	1+0	1	Y	Quorum
6	62	1+0	10	X	System Databases
6	61	1+0	250	F	Metamember
7	70	1+0	600	P	Database
7	71	1+0	250	F	Metamember
8	80	1+0	600	T	Database
8	81	1+0	250	F	Metamember
9	90	1+0	650	L	Logs
9	91	1+0	250	F	Metamember
10	110	1+0	935	E	Backup
11	111	1+0	935	D	Backup

Table 1: Storage layout and RAID levels

Figure 3 shows the views of the various volumes available to the host system as presented by the storage subsystem.

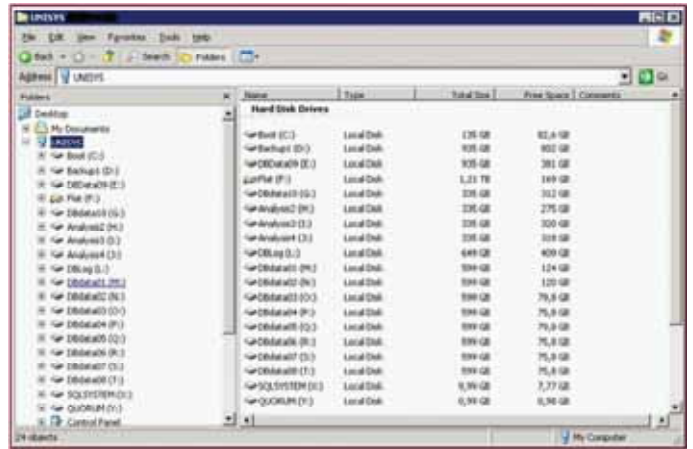


Figure 3: Storage configuration

After we set up the storage, we used the SQL IO tool to test the baseline performance of the IO subsystem. The SQL IO tool simulated reading and writing in configurable batch sizes similar to normal SQL Server operations. As an example, in the multipath latency test the SQL IO tool yielded a 1+ GB/sec cache hit reading from a single volume. (Figure 4) The SQL IO tool also tested both sequential and random read/write operations.

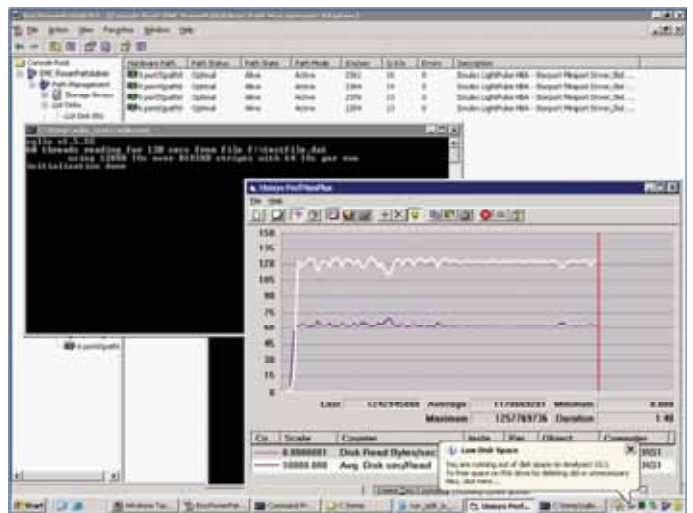


Figure 4: Sample of measuring the storage SAN connectivity / latency using SQL IO tool

ETL Data Load Test

ETL Setup and Optimization

We used the T-SQL BULK INSERT command to load the data as this provided all the required functionality and performance. Also this made the bulk load operation parallelism easy to control for the proof of concept. Some of the key optimization techniques used for data load include

- removing indexes on target table while loading. Index build is vastly improved in SQL Server 2005 and rebuilding the indexes after the data load is much faster overall.
- using TABLOCK hint. This allows SQL Server to put bulk update locks. The bulk update locks allow concurrent multi-thread loading into the same table but prevent other non bulk load processes from accessing the table.
- disabling check constraints during the load. Checking the constraint as a separate pass will improve the processing significantly.

The BULK INSERT can be started from the command line to enable starting multiple instances of import in parallel, e.g.

```
SQLCMD ES7000\SQLTest -E -dstagingDB -Q"exec dbo.spDoWork"
```

where the stored procedure looks like:

```
CREATE PROCEDURE [dbo].[spDoWork]
AS
DECLARE @DivNo CHAR(3)
DECLARE @FileName NVARCHAR(255)
DECLARE @FormatFile NVARCHAR(255)
EXEC spGetNext @DivNo OUTPUT, @FileName OUTPUT, @FormatFile OUTPUT
IF @DivNo IS NOT NULL BEGIN /* There is work to be done */
    DECLARE @Sql NVARCHAR(MAX)
    SET @Sql = 'BULK INSERT IMPORT_BC_DAT_BC_DAT_' + @DivNo
    + ' FROM "f:\POC_DATA\' + @FileName
    + '" WITH (FORMATFILE = 'C:\PROJECT\' + @FormatFile+ ',
    TABLOCK, FIRSTROW=2, MAXERRORS=20)'
    PRINT @Sql
    EXEC sp_executeSql @Sql
    EXEC spDoWork
END
```

The source and related format file list can be managed within SQL Server 2005 to enable better logging, error handling and restart capabilities. Figures 5 and 6 show the database table used for managing the file and a sample format file.

FileName	DivNo	Done	FormatFile	TimeTakenFromQueue
ora15_d_801Payment_Type.txt	115	True	Payment_Type.fmt	10:51:57
ora16_d_801Bc_Dat.txt	116	True	Bc_Dat.fmt	10:52:04
ora16_d_801Bc_Dat_Product.txt	116	True	Bc_Dat_Product.fmt	10:52:19
ora16_d_801Bc_Dat_Product_Details.txt	116	True	Bc_Dat_Product_Details.fmt	10:52:42
ora16_d_801Bc_Dat_Tax.txt	116	True	Bc_Dat_Tax.fmt	10:52:42
ora16_d_801Bc_Dat_Tax_Details.txt	116	True	Bc_Dat_Tax_Details.fmt	10:52:45
ora16_d_801Payment_Type.txt	116	True	Payment_Type.fmt	10:52:46
ora17_d_801Bc_Dat.txt	117	True	Bc_Dat.fmt	10:52:47
ora17_d_801Bc_Dat_Product.txt	117	False	Bc_Dat_Product.fmt	NAE1
ora17_d_801Bc_Dat_Product_Details.txt	117	False	Bc_Dat_Product_Details.fmt	NAE1
ora17_d_801Bc_Dat_Tax.txt	117	False	Bc_Dat_Tax.fmt	NAE1
ora17_d_801Bc_Dat_Tax_Details.txt	117	False	Bc_Dat_Tax_Details.fmt	NAE1

Figure 5: Database table used for file organization

Figure 6: Content of one of the BULK INSERT command format files

ETL Test Result

We executed the tests with varying number of parallel data loads to get the most optimum load rate. SQL Server handled increasing amounts of data with consistent performance. The number of input files were increased until the CPUs were becoming a bottleneck and execution time started to increase. We were able to scale running 32 bulk insert operations in parallel on a 32-core system. The data load throughput peaked at 3.5 million rows per seconds, providing a mechanism to load a billion rows into the data warehouse in under 5 minutes.

OLAP Cube Processing Test

OLAP cube processing roughly involves three main phases: reading data from the source file (in our case the relational data warehouse); summarizing the data and computing aggregation; and then writing the required data in the OLAP cube structure. The testing cube was designed as MOLAP for the best querying performance.

Relational Design Impact on Cube Processing

The design of the source relational database can have a significant impact on the overall cube performance. Some standard best practices such as the right level of indexes and data types should be evaluated when creating a cube.

Some of the optimizations done to the relational data warehouse during the testing are listed below.

- All surrogate keys in the fact table are integers
 - Don't use int when smallint is enough
- Dimension tables are heavily indexed
 - All indexes: FILLFACTOR = 100
 - Use included columns to support every dimension process query
- Cluster index added in the fact table to match cube partitions
 - Fillfactor = 100
- Use partitioned tables or partition the data as separate tables
- Measure columns were originally created as decimal
 - Use money data type where possible as the performance difference listed later
- SQL latch wait time can be minimized by creating views. For example, a simple view like the following was used during the testing:

```
CREATE VIEW [Cube].[Fakt_Bc_Dat_Product_Details]
AS
SELECT [OrgID]
      ,[Bc_Number]
      ,[RecCnt]
      ,[ProductID]
      ,[ReceiptDateID]
      ,[ReceiptTimeID]
      ,[Gross]
      ,[Quantity]
      ,[Total]
      ,[Bc_PosID]
      ,[IsPledge]
      ,[StornoType]
      ,[AuditID]
FROM [dbo].[Fakt_Bc_Dat_Product_Details] WITH (NoLock)
```

- Use 32 KB network packet size for SQL connection, as shown in Figure 7

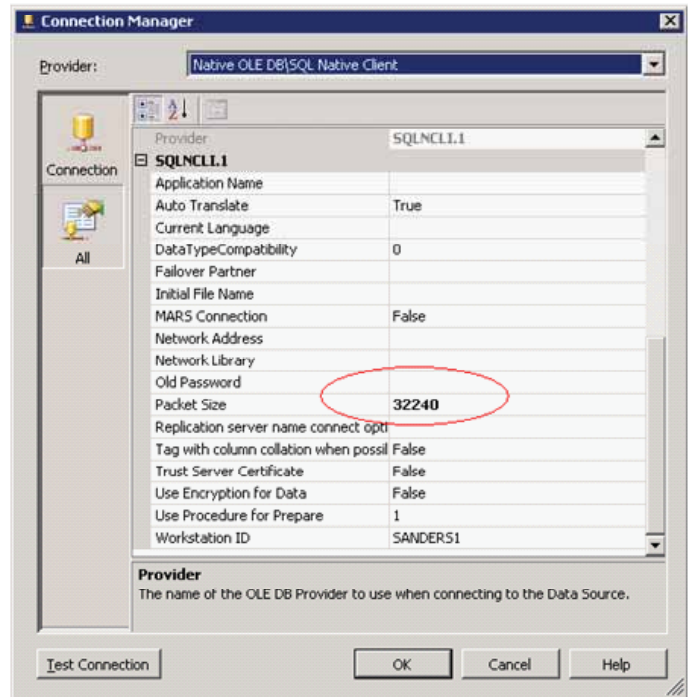


Figure 7: Large network packet size for connections to SQL Server

Cube Setup and Optimization

The OLAP cube was designed to meet all the business requirements and allow users to perform fast queries. Some of the design techniques used for the cube include

- Attribute relationships in dimension defined to support all hierarchies
- UDM build on top of views not tables
 - Allows implementing optimization tweaks without changing cubes
- Ensure dimension attribute counts updated
- Applied aggregation wizard combined with usage based
 - Aggregation storage about 25% of total cube storage
- Using Money data type instead of decimal yielded a 13% improvement
 - Storing as money (a 8-byte integer with implied 4 decimal digits)

SQL Server uses the Tabular Data Stream (TDS) format to transfer data over the wire, and it does not support decimal or numeric. We could change the relational database, or cast or convert during the SQL query that SSAS sends (and create a view if that would help manageability).

Column Name	Data Type	Allow Nulls
ProductID	int	<input type="checkbox"/>
RecCnt	smallint	<input type="checkbox"/>
Gross	decimal(11, 2)	<input type="checkbox"/>
Quantity	decimal(9, 3)	<input type="checkbox"/>
IsPledge	bit	<input type="checkbox"/>
StornoType	tinyint	<input type="checkbox"/>
AuditID	smallint	<input type="checkbox"/>
BonNumber	int	<input type="checkbox"/>
ReceiptDateID	int	<input type="checkbox"/>
Total	decimal(11, 2)	<input type="checkbox"/>
OrgID	int	<input checked="" type="checkbox"/>
ReceiptTimeID	smallint	<input type="checkbox"/>
BonPosID	tinyint	<input type="checkbox"/>

➔

Column Name	Data Type	Allow Nulls
OrgID	int	<input checked="" type="checkbox"/>
BonNumber	int	<input type="checkbox"/>
RecCnt	smallint	<input type="checkbox"/>
ProductID	int	<input type="checkbox"/>
ReceiptDateID	int	<input type="checkbox"/>
ReceiptTimeID	smallint	<input type="checkbox"/>
Gross	money	<input type="checkbox"/>
Quantity	money	<input type="checkbox"/>
Total	money	<input type="checkbox"/>
BonPosID	tinyint	<input type="checkbox"/>
IsPledge	bit	<input type="checkbox"/>
StornoType	tinyint	<input type="checkbox"/>
AuditID	smallint	<input type="checkbox"/>

Figure 8: Conversion from Decimal to Money data type

We used the following approaches to maximize the processing performance.

- Data partitioning: data was partitioned on the relational data warehouse and the cube was defined with a similar aligned partition
 - Find the optimal maximum number of rows for each partition and create partitions accordingly
 - Three partitions per month were implemented in the cube for a total of 180 partitions
- Tests were conducted to determine the maximum rows/sec throughput overall for the available server resources
- The ProcessIndex phase defaulted to only three partitions to be processed simultaneously
 - With the following msmdsrv.ini changes, 14 active partitions were processed in parallel

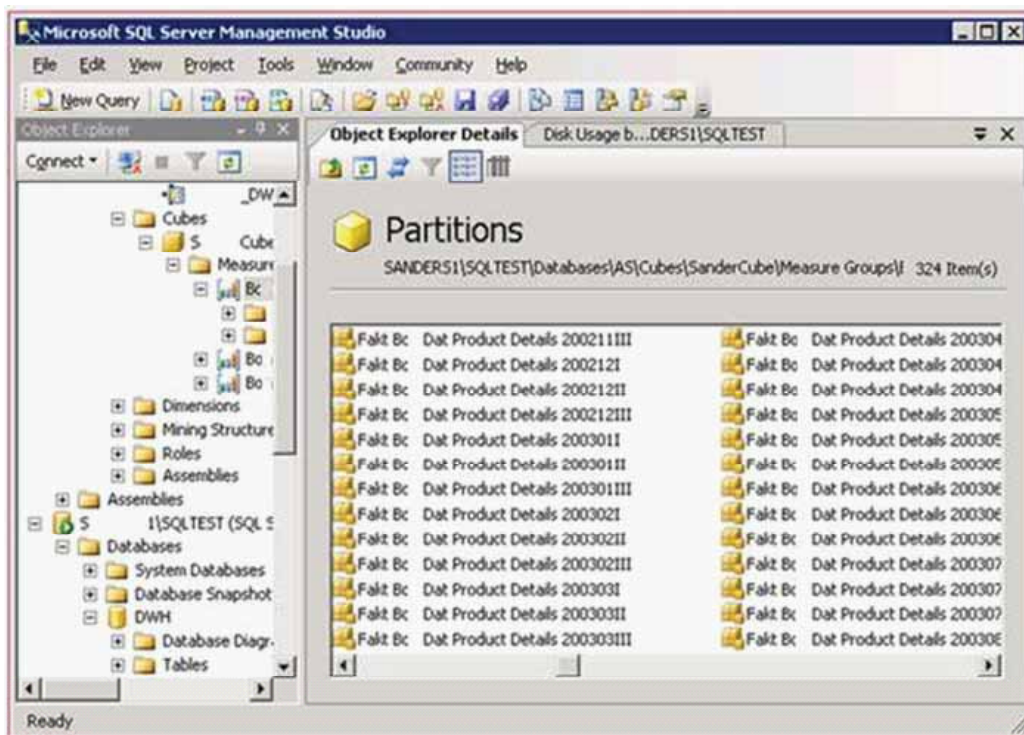


Figure 9: Partition of the OLAP cube

Parameter	Optimized	Original value
CoordinatorExecutionMode	-8	-4
Memory\LowMemoryLimit	70	75
OLAP\Process\ AggregationMemoryLimitMin	1	10
OLAP\Process\ AggregationMemoryLimitMax	5	80
OLAP\Process\ DatabaseConnectionPoolMax	64	50
ThreadPool\Process\MaxThreads	128	64
PreAllocate	20	

- Note: these optimized parameters are currently being evaluated and are also undergoing regression tests at the Microsoft test labs
- The number of partitions being processed was gradually increased till a certain CPU utilization number and then we increase the number of CPUs/cores
- We created a cube partition processing matrix to keep track of the processing rate

Processed Data – Sizes

SQL fact tables: 436.7 million rows / 28.6 GB each

18 aggregates

File sizes on disk:

- 120 files per partition
- 628 MB per partition

Compare this with the cube data sizes in Figure 10 to see the compression at work.

Table 2: Optimized SSAS parameters: msmdsrv.ini changes

Name	Size	Type	Date Modified
163.fact.data	401.456 KB	DATA File	
163.agg.rigid.data	58.655 KB	DATA File	
163.Bc...Dim Bc...Number.fact.map	55.404 KB	MAP File	
163.agg.flex.data	54.972 KB	DATA File	
163.Product.Dim Product.fact.map	16.780 KB	MAP File	
163.Product.Description.fact.map	15.195 KB	MAP File	
163.Product.Product Code.fact.map	15.016 KB	MAP File	
163.Organisation.Dim Org.fact.map	6.911 KB	MAP File	
163.Organisation.Cashier No.fact.map	6.911 KB	MAP File	
163.Organisation.Store No.fact.map	3.504 KB	MAP File	
163.Product.SCG Desc DE.fact.map	2.517 KB	MAP File	
163.Organisation.Div No.fact.map	797 KB	MAP File	
163.Product.CG Desc DE.fact.map	591 KB	MAP File	
163.Time Of Day.Time Until.fact.map	473 KB	MAP File	
163.Time Of Day.Time From.fact.map	473 KB	MAP File	
163.Time Of Day.Quarter Text Short.fact.map	473 KB	MAP File	
163.Time Of Day.Quarter Text Long.fact.map	473 KB	MAP File	
163.Time Of Day.Dim Time Of Day.fact.map	473 KB	MAP File	
163.Time Of Day.Hour Text Short.fact.map	88 KB	MAP File	
163.Time Of Day.Hour Text Long.fact.map	88 KB	MAP File	
163.fact.data.hdr	85 KB	HDR File	
163.Time.Wo...Fact.map.hdr	71 KB	HDR File	
163.Time.W Tag.fact.map.hdr	71 KB	HDR File	
163.Time.Quartal Ja...fact.map.hdr	71 KB	HDR File	
163.Time.Monat Ja...fact.map.hdr	71 KB	HDR File	
163.Time.Ja...fact.map.hdr	71 KB	HDR File	
163.Time.ID.fact.map.hdr	71 KB	HDR File	
163.Time Of Day.Time Until.fact.map.hdr	71 KB	HDR File	
163.Time Of Day.Time From.fact.map.hdr	71 KB	HDR File	
163.Time Of Day.Quarter Text Short.fact.map.hdr	71 KB	HDR File	

Figure 10: Cube data sizes

32-Core System Cube Processing Results

The cube was processed on a 32-core system by gradually increasing the number of partitions processed in parallel. The best throughput measured on this system was for a cube with

Total # Rows: 1,323,338,880 (~1.3 billion)
Partitions: 32
Parallel tasks: 32

The total processing time was 21 minutes 04 seconds with a throughput of 1,046,945 rows per second. The snapshot system resource utilization during this processing test is shown in Figure 12.

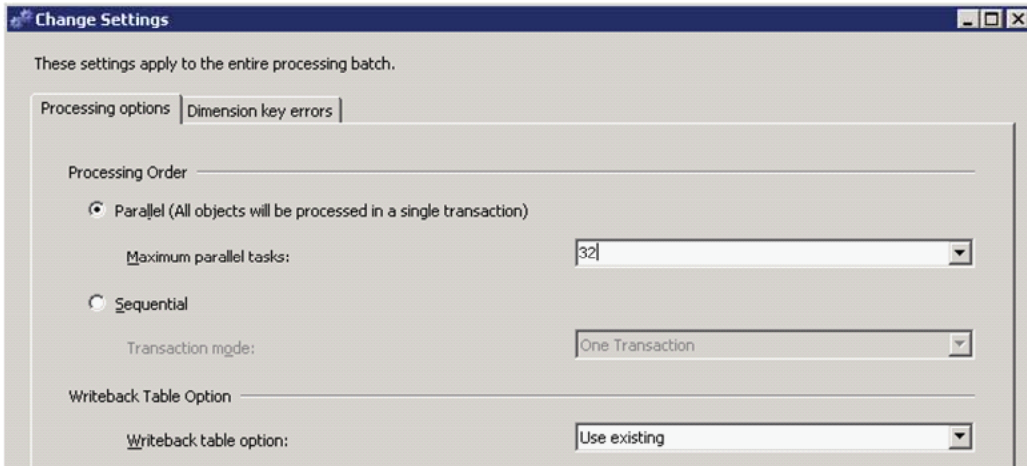


Figure 11: Cube processing 32 parallel tasks

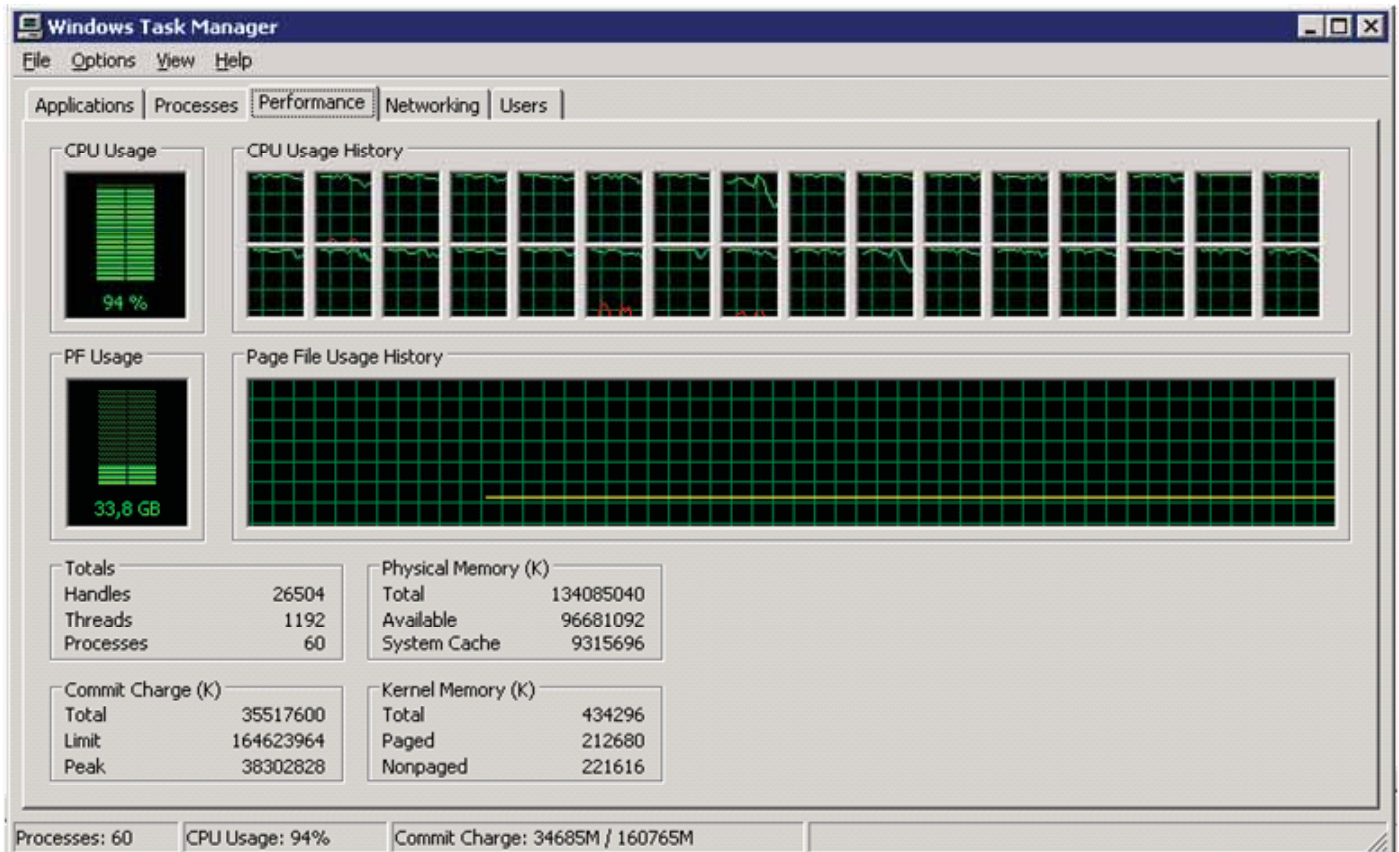


Figure 12: 32 parallel tasks cube processing CPU utilization

The 32-core system cube processing with various parallel processing numbers are shown in Figure 13. As shown the full cube processing throughput reaches a plateau between 28 to 32 parallel partition processing.

64-Core System Cube Processing Result

The final phase of the testing was done with full cube processing with the full processor capacity of the dual-core 32 sockets server. The second node of the cluster was shutdown and all the resources were allocated to the one partition. The best measured performance was for a cube with the following settings:

- SQL TCP/IP connections on loopback adapter
- 51 partitions
- 2,109,071,340 rows (~2.1 billion)

The total processing time was 46 minutes 12 seconds with an average throughput of 760,848 rows per second. Figure 14 shows the CPU and network utilization during this test and clearly the system was not very well utilized, which was a point for investigation.

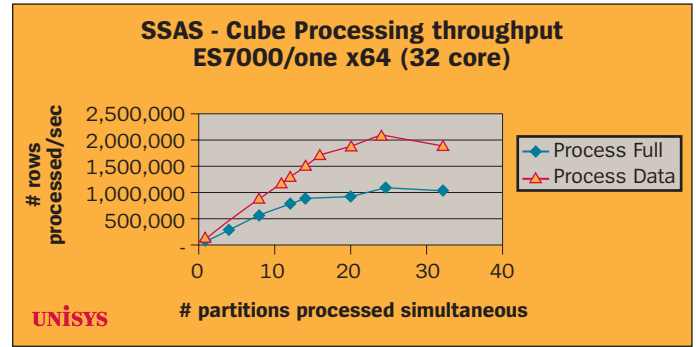


Figure 13: Varying number of parallel partitions processing throughput

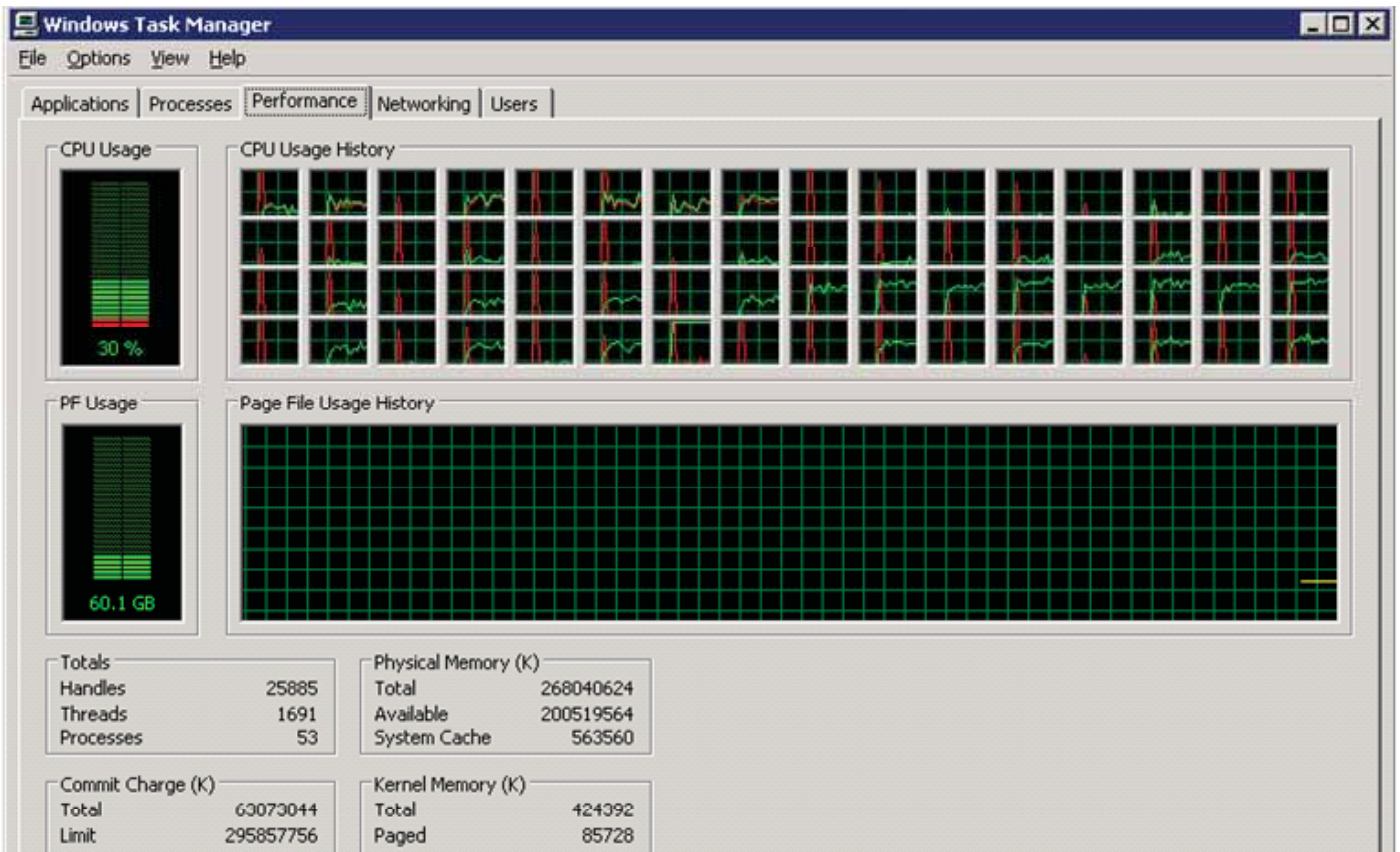


Figure 14: CPU utilization of 64 cores with TCP/IP connection (limited)

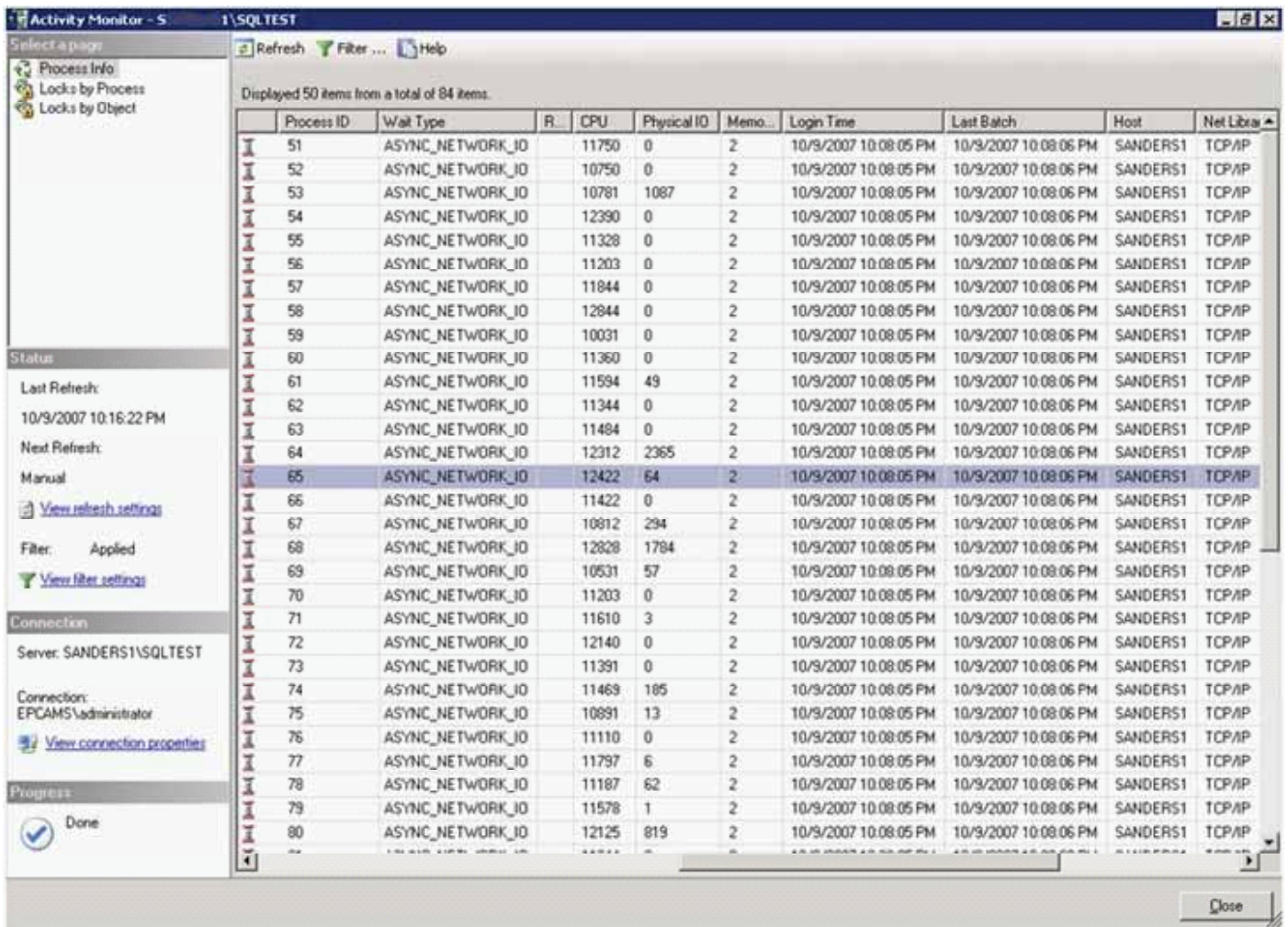


Figure 15: Activity Monitor showing hourglasses, during the 64 core processing with TCP/IP connections

Notice the tasks are suspended and generally waiting for ASYNC_NETWORK_IO.

The second test on 64 cores was done on a local SQL instance instead of the clustered instance, which allowed usage of SQL in shared memory (LPC) connections. With the same SQL configuration defined in this local instance, the parallel processing capacity of SQL shows a amazing increase of throughput.

- shared memory LPC connections
- 51 partitions
- 2,109,071,340 rows (2.1 billion)

The processing time was reduced to 19 minutes 8 seconds with an average throughput of 1,821,305 rows per second. Figures 16 and 17 show the CPU and network utilization during this test and clearly the system is doing much more productive work.

The results on the following page show that SSAS on a 64-bit Unisys ES7000 server scales very well, and takes full advantage of additional CPUs as the number of processes being executed in parallel increases.

These results conclusively demonstrate the highly parallel performance of data load for data warehouse and the OLAP cube processing running on the Unisys ES7000 server. A key contributing factor is the large memory support and ability to perform caching of large dimension tables in the 64-bit environment.

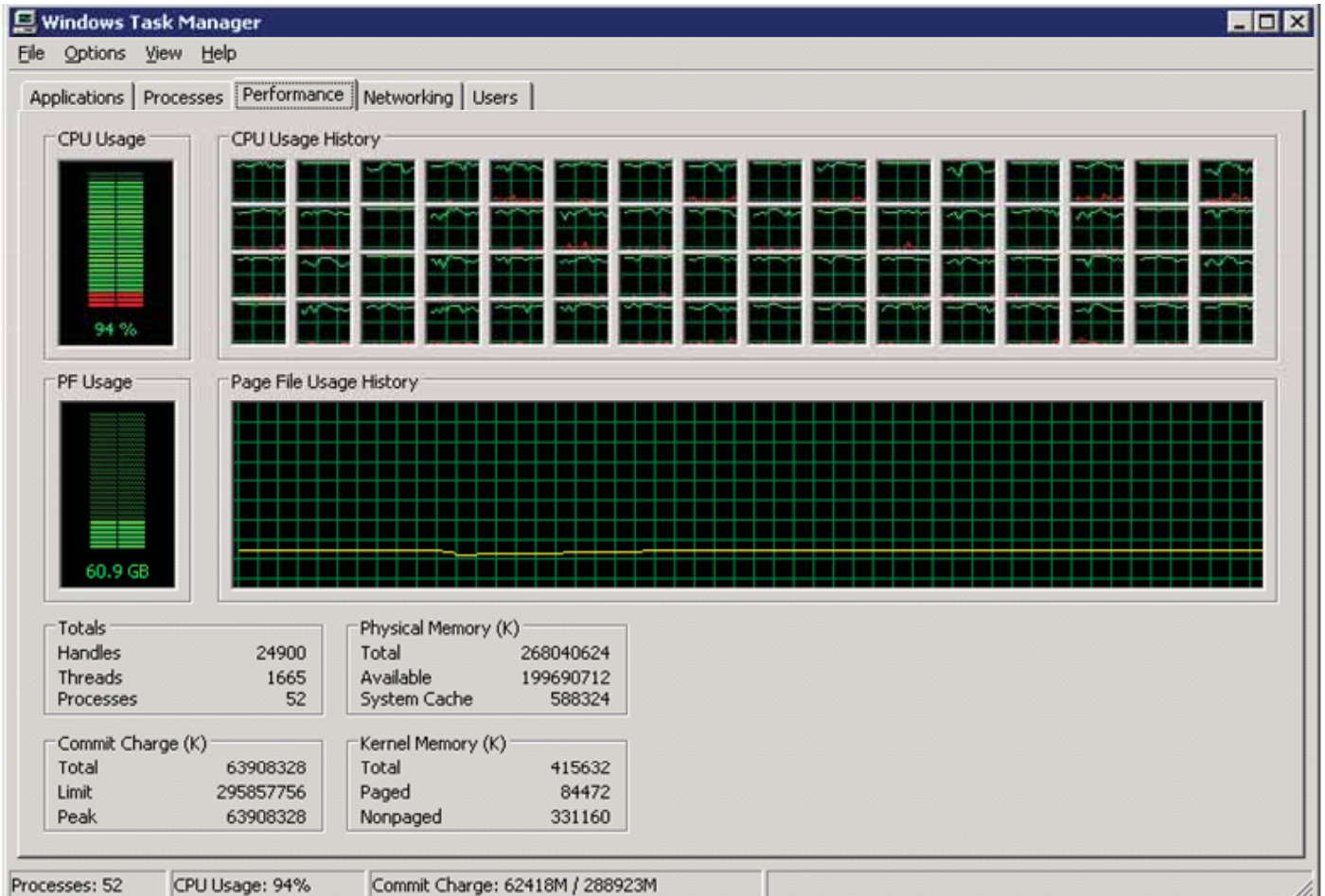


Figure 16: Optimized CPU utilization for 64 cores processing using the fast LPC connections

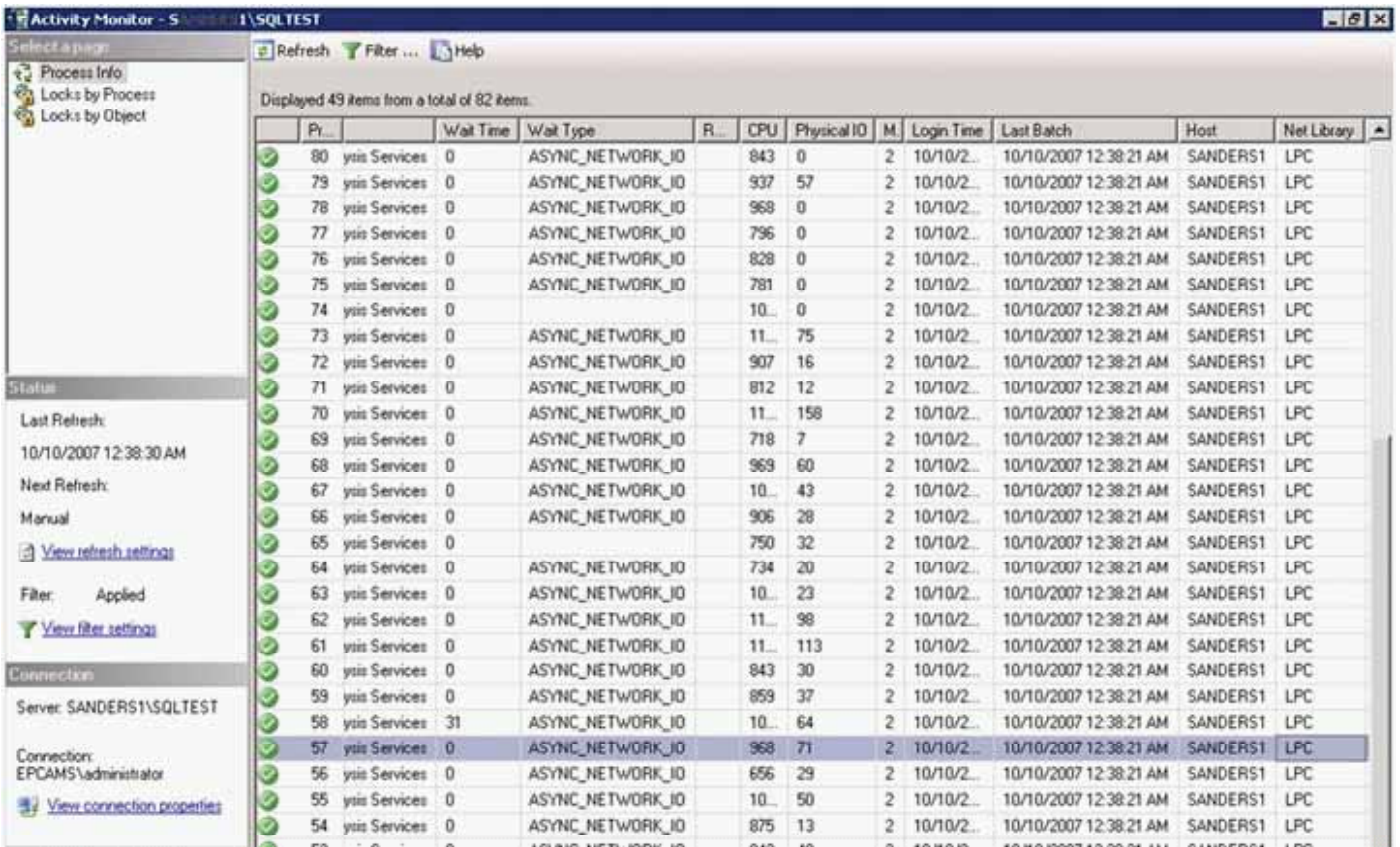


Figure 17: Optimized Activity Monitor for 64 cores processing using the fast LPC connections

Conclusion

These test results prove that the Microsoft SQL Server 2005 Enterprise Edition suite of products running on a Unisys ES7000 server can handle complex, large, enterprise-scale data warehouse operations. We provide a viable alternative to competitive pure play ETL options or specialized business intelligence options. Our solution takes advantage of parallelism and multiple processors to deliver extremely high levels of transaction throughput. The scalable 64-bit platform with the economy of better compatibility of x64 processor can execute complex operations quickly by using the immense memory capacity of the 64-bit system.

ES7000 servers are ideal for hosting demanding or high-growth data warehousing environments. Taking advantage of the efficient, large-scale I/O throughput inherent to Intel EM64T-based ES7000 servers, our solution can

- manipulate large data sets
- scale efficiently with increased number of processors
- process diverse ETL, data management, and business intelligence processing in parallel and manage them all effectively on a single server
- provide the ability to manage and allocate resources to accommodate diverse processes and user demands

With the SQL Server 2005 product suite, Microsoft and Unisys deliver an optimized data warehouse solution that will support even the most demanding enterprise business intelligence initiatives. Through collaborative benchmark testing and engineering, we mitigate implementation risk by validating optimal performance as well as creating best practices in system sizing and configuration to meet high performance.

About the Unisys Performance Centre for Business Intelligence

The Unisys Performance Centre for Business Intelligence is equipped with state-of-the-art ES7000 server technology and an expert staff and global support that collaborate with Microsoft Research and Development, technical architects and pre-sales support organizations to reduce

client risk of enterprise-wide Microsoft deployments. The Unisys COE for Business Intelligence specializes in performance and scalability testing for data warehousing, analytics and business intelligence applications and databases. The Centre provides the skills and resources to help plan, deploy, and optimize business intelligence solutions using Unisys ES7000 servers and leading best-of-breed partner technologies in a no-risk environment, using your own data.

About Unisys

Unisys is a worldwide information technology services and solutions company. Unisys combines expertise in consulting, systems integration, outsourcing, infrastructure and server technology with precision thinking and relentless execution to help clients, in more than 100 countries, quickly and efficiently achieve competitive advantage. For more information, visit <http://www.unisys.com>.

About Microsoft

Microsoft (Nasdaq "MSFT") is the worldwide leader in software, services and solutions that help people and businesses realize their full potential. For more information, please visit <http://www.microsoft.com>.

About Platon

Platon A/S was founded in 1999. With more than 150 employees and twice as many clients, and offices in Denmark, Norway, Sweden, Finland, and Iceland, Platon is today the largest independent consulting firm in the Nordic Region within our business field. Platon is currently expanding its field of operation internationally through partners in Europe, Asia and North America. They focus exclusively on Information Management and related fields. Our skills and expertise covers Business Intelligence, Business Integration, Data Warehousing, Master Data Management, Financial Management, Customer Relationship Management and Knowledge Management. They cover both the business side as well as the IT side of these businesses in all leading technologies. For more information, please visit <http://www.platon.net>.

© 2008 Unisys Corporation.

All rights reserved.

Unisys is a registered trademark of Unisys Corporation. Intel, Itanium, and Intel are registered trademarks of Intel Corporation. Microsoft, PerformancePoint, ProClarity, and Windows Server are registered trademarks of Microsoft Corporation. All other brands and products referenced herein are acknowledged to be trademarks or registered trademarks of their respective holders.

February 2008



4137 0958-000