Savonia
University of Applied Sciences

Application Workflow Analysis and Simulation

Bachelor's Thesis

Tuomo Ruuskanen

Degree Programme in Information Technology

Accepted ___.___.____

SAVONIA UNIVERSITY OF APPLIED SCIENCES Degree Programme Information Technology		
Author Tuomo Ruuskanen		
Title of Project Application Workflow Analysis and Simulation		
Type of project Final Project	Date 25 May, 2010	Pages 36
Academic Supervisor Mr. Keijo Kuosmanen	Company Supervisor Mr. Henry Sulkers	
Company GE Healthcare Finland Oy		

Abstract

Computer systems are growing in performance very rapidly. Modern processors can handle massive amounts of data in a very small period of time. It has become apparent that other subsystems in a computer are becoming an increasing bottleneck. The hard disk subsystem is the slowest part of the computer system.

GE Healthcare initiated this project to provide better information about minimum system requirements to their clients. The purpose of this project was to obtain comprehensive knowledge about disk subsystem usage and performance in certain scenarios.

The first part of this project for GE Healthcare was to analyze data flow and corresponding disk subsystem input and output of two clinical information systems. The second part of this project was to create synthetic simulation tooling to stress the disk subsystem in a way similar to real application environment.

The results of this project showed that there is a clear similarity between theorized disk usage and real disk usage. Knowledge received from this project showed that synthetic performance measurement can be used to test the disk subsystem before a real application is deployed.

Keywords

Performance, database, storage subsystems, workflow analysis, hard disk.

Confidentiality

Publi

SAVONIA-AMMATTIKORKEAKOULU Koulutusohjelma Tietotekniikka		
Tekijä Tuomo Ruuskanen		
Työn nimi Ohjelmiston Tiedonkulun analysointi ja simulointi		
Työn laji Insinöörityö	Päiväys 25.5.2010	Sivumäärä 36
Työn valvoja lehtori Keijo Kuosmanen	Yrityksen yhdyshenkilö Henry Sulkers	
Yritys GE Healthcare Finland Oy		

Tiivistelmä

Tietokonejärjestelmien suorituskyky on kasvanut nopeasti. Modernit suorittimet laskevat massiivisia määriä tietoa pienessä ajassa. On ilmennyt, että muut alijärjestelmät tietokoneessa ovat kehittymässä pullonkaulaksi. Yleisesti ottaen, jos siirrymme kauemmaksi suorittimesta kaistanleveys pienenee eksponentiaalisesti ja keskimääräinen hakuaika kasvaa eksponentiaalisesti. Kovalevyjärjestelmä on tietokoneen hitain alijärjestelmä.

GE Healthcare käynnisti tämän projektin saadakseen lisää tietoa järjestelmän minimivaatimuksista. Tämän lopputyön tarkoitus oli saada kattavaa tietoa levyjärjestelmän käytöstä ja suorituskyvystä.

Tässä lopputyössä analysoitiin kliinisten tietojärjestelmien tallennusjärjestelmään kohdistuvaa liikennettä. Toinen osa lopputyötä oli kehittää synteettinen työkalu, joka rasittaa levyjärjestelmää kuten oikea ohjelmistoympäristö.

Tämän lopputyön tulokset näyttävät, että teoreettisen levynkäytön ja oikean levynkäytön ero on pieni. Tässä lopputyössä saavutettu tieto osoittaa että, synteettistä suorituskyvyn mittausta voidaan käyttää levyjärjestelmän testauksessa .

Avainsanat
suorituskyky, tietokannat, levyjärjestelmät

Julkisuus
Julkinen

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

During the course of this project I have learned more how to work in a IT company. One of the most important things I have learned is how to work in a project. It is important to define goals in a project and to divide the work in smaller pieces to better track successes and problems.

I want to thank GE Healthcare for providing this opportunity for me. Additionally I want to thank my family, friends, teachers and colleagues for providing much needed support.

ABBREVIATIONS

abbreviation	**meaning**
RAM	Random Access Memory
CPU	Central Processing Unit
DB	Database
OS	Operating System
HDD	Hard Disk Drive
RAID	Redundant Array of Independent Drives
NAS	Network Attached Storage
SAN	Storage Area Network
MB	Megabyte
GB	Gigabyte
ns	nanosecond
ms	millisecond
I/O	Input / output
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer System Interface
SAS	Serial Attached SCSI

1. INTRODUCTION

1.1 Project Introduction

GE Healthcare Finland started this project to analyze disk subsystem traffic in two different clinical information systems. These two application environments will be referred as application 1 and application 2. These applications have multiple software components and function in a computer network environment. A typical environment consists of a number of servers and multiple client workstations. Due to time limitations, only the ingoing data from medical devices to the disk subsystem will be examined in this thesis. This thesis is divided into five parts.

In the course of this project it is important to study what components are involved in a general computer system. The study of application work-flow and disk subsystem traffic requires knowledge of different subsystems: disk systems, caching between components, random access memory and the processor. Without the knowledge about these components it is impossible to determine bottlenecks in application workflow or to set up measurement tools correctly.

The goal of this project is to analyze disk subsystem traffic in two application environments. This thesis provides information on what software and hardware components are involved in application 1 and application 2 environments. Both applications use the same database software. The internal logic and internal monitoring tools of the database are examined in this thesis. General names of the software components are used, no real component names are given.

The major part of this project is to measure disk subsystem traffic. This thesis provides information on what tools are used in the measurements. Operating system tools are used to measure disk behavior and database internal tools are used to measure database activity. In this project these tools are used extensively, therefore automation and scripting of these tools will be discussed.

The most interesting part of this thesis will be the results gathered from measurements.

The resulting data gives information about disk subsystem utilization under load by different sets of medical devices in both application environments. Only the difference between these applications will be provided, no real measurement data will be given. The correlation between theoretical disk input and actual measured input will be calculated and analyzed.

The final part of this project is to measure real world performance of certain disk subsystems. This thesis provides information on what tools were used to measure performance and how they are configured. The goal of the performance measurements is to simulate how well the disk subsystem would behave under load by application 1 or application 2 before the actual applications are installed.

1.2 Centricity

In this project the workflow of data is analyzed in two Centricity products: Centricity Critical Care and Centricity Anesthesia. It is reasonable to provide some information on which kind of uses these applications have and what environments they are used in.

Centricity is a clinical documentation and information system that provides doctors, nurses and other care personnel access to all patient information and documentation at the point-of-care or in another convenient location remotely whenever and wherever needed. The application provides care personnel reference information and document templates in clinical setting throughout the whole care process. /1/

1.3 Centricity Critical Care

Centricity Critical Care includes a wide selection of different tools. This Clinical information system provides ability to easily and reliably standardize, plan, and document the whole care process. It makes available all the information needed for effective patient care and resource planning. The system can be easily modified according to local need of the care unit. Care procedures can be configured according to preferred standards without any programming. /1/

Centricity Critical Care enables doctors and nurses to concentrate more on patient care by reducing their workload at the point-of-care. Automatic and standardized documentation can reduce the probability of medical errors. Data is automatically gathered from bedside medical devices such as infusion pumps, monitors and ventilators. This system enables automatic system integration to other hospital systems such as patient administration and radiology department reducing time needed for documentation. /1/

Care documentation can be done at the point-of-care, which enables early and on site data collection. The remote station provides the possibility to access all patient data and a way to plan treatments from a remote location such as a physician's office. The multibed station provides tools for device interfacing and care documentation from one workstation to multiple beds in intermediate care units like high dependency and step down units. The user-interface of Centricity Critical Care is depicted in Figure 1. /1/

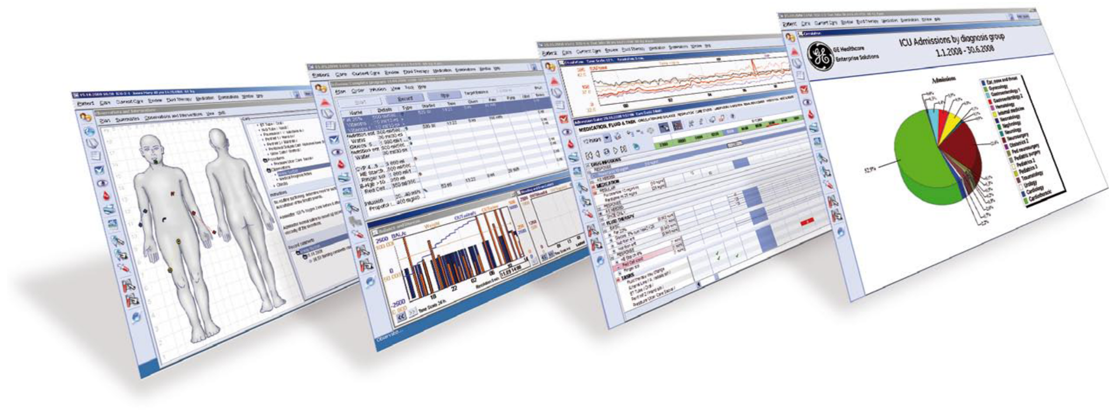


Figure 1. Centricity Critical Care. /1/

1.4 Centricity Anaesthesia

The fundamental priorities in anesthesia include providing high quality patient care with efficient planning and care process management. Achieving these goals means having the crucial information available at the point-of-care. Centricity Anesthesia is a clinical information system. It is an easy to use solution for high performance perioperative environment. Centricity Anesthesia provides electronic anesthesia records of the entire anesthesia process from pre-operative assessment to post anesthesia care. /1/

This solution helps medical personnel to see patient data more clearly and accurately by automatically collecting data from medical devices or hospital information systems. The user interface is optimized to save time and increase data viewing by touch screen optimized user interface, preconfigured cases and pick lists. The needs of care personnel are usually unique and different, therefore this system has been designed to be flexible for a wide variety of configurations. /1/

Centricity Anesthesia is scalable from small stand-alone units to fully networked large installations. Centricity anesthesia provides an option to install a basic documentation version or a full perioperative solution to cover the entire process from anesthesia planning to post anesthesia care. The user-interface of Centricity Anaesthesia is depicted in Figure 1. /1/

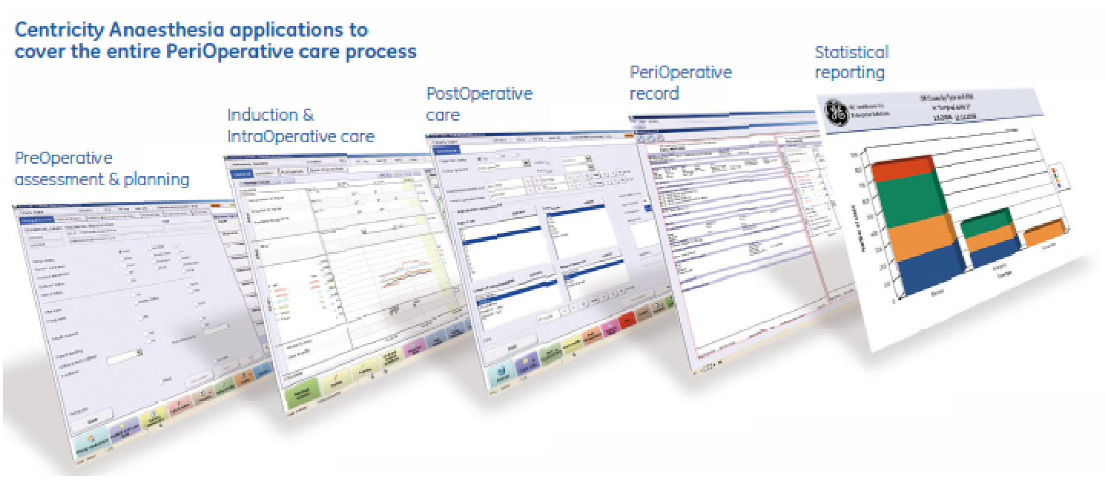

Centricity Anaesthesia applications to cover the entire PeriOperative care process
PreOperative assessment & planning
Induction & IntraOperative care
PostOperative care
PeriOperative record
Statistical reporting

Figure 2. Centricity Anaesthesia. /1/

1.5 General Computer System Overview

A typical computer system consists of the following parts: the central processing unit or the CPU, chipset, random access memory, non-volatile storage i.e. hard disk. These are the main components that are involved in storing, fetching and processing data. T computer system contains other components as well, such as power supply, several peripherals, numerous I/O devices and so on. In the context of this thesis it is only important to examine those parts, which are involved in moving, storing and processing the actual data. Figure 3 shows a typical computer system. The application fetches data from the hard disk, processes it in the CPU, while using memory as a “working table” to store processed data.

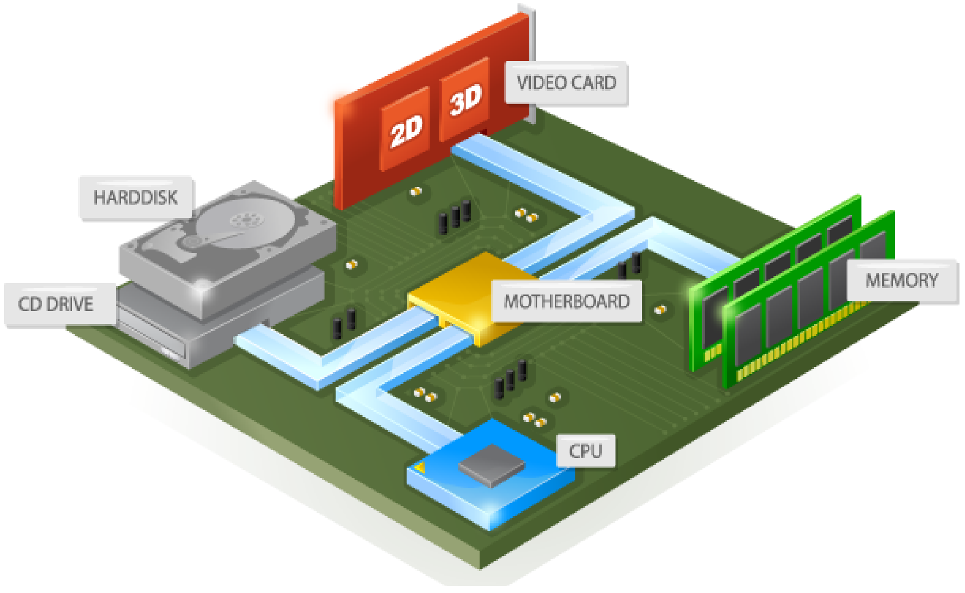


Figure 3. Computer system. /2/

1.6 Bottlenecks in General Computer Systems

Modern computer systems have grown in performance very rapidly since the first computers were built in the mid 20th-century. This trend of growing performance has been steady and there is no end in sight. The Moore's law which states that the number of transistors in a processor doubles every two years is still true and will continue to be true for years to come. This increase in transistors has made possible the increase of processor operating frequencies.

In a computer system it has become apparent that other subsystems have become a bottleneck for the processor, moving farther away from the processor the working memory of a computer system called the random access memory, RAM is reached. Increase in frequencies of the RAM has clearly fallen behind from increasing CPU frequencies this has lead to the invention of multi-core CPUs and multi-channel memory systems. It is more efficient to use the CPU transistors to create multiple cores and to access data via multiple channels simultaneously from the memory. Still the memory subsystem is significantly slower than the internal caches of the CPU, which is the so-called Von Neumann Bottleneck.

The non-volatile storage of a computer system or the hard disk subsystem is clearly the slowest component of the overall system. The mechanical nature of hard disk storage makes it the slowest part of the computer system. Total throughput of a single HDD is typically 100 times smaller than the throughput of the RAM system and the random access time is typically almost 1 million times longer.

To combat increasing bottlenecks in a computer system it is reasonable to add caches between slower parts of the system. For example the CPU has usually multiple levels of internal cache memory to access frequently used data faster. The cache closest to the CPU is the fastest, but also the smallest and the cache farthest from the CPU is the slowest, but also the largest. Random access memory can in some applications be thought as a cache between the CPU and the hard disk storage system. Usually hard disks have their own internal caches and bigger storage systems such as RAID systems have a storage-controller, which can have multiple gigabytes of cache for faster access of frequently used data. This helps to queue write operations more efficiently. Caching of write operations does not come without a risk, a loss of power could lead to data loss or corruption of data.

The computer system can be thought as a person who is working at a table. The person is the CPU, memory is the table and hard disk is the bookshelf that is located farther away from the person. Items that are on the table can be accessed faster than the items in the bookshelf. Approximations of typical bandwidths and access times gathered in Table 1.

Table 1. Subsystem performance

Subsystem	**Bandwidth**	**Access latency**
CPU	~100-300GB/second	1-10ns
RAM	~5-30GB/second	10-20ns
Mechanical hard drive	10-100MB/second	8-15ms
Solid state drive	50-250MB/second	0,1-10us

The problem of slower external systems and caching as a solutions has been known for a long time. The designers of the first computer systems in 1946 already realized this problem and this development has continued throughout the ages of computing.

“Ideally one would desire an indefinitely large memory capacity such that any particular word would be immediately available. We are forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible.” /3/

2. APPLICATION WORKFLOW

In this project for GE Healthcare the disk subsystem traffic was analyzed in two different application environments. These environments contain several servers, client workstations and software components to transform data between different formats. Application 1 and application 2 share some similar components. Different components or devices of the overall systems are described in Figures 4 and 5.

2.1 Application 1

Application 1 environments consists of the following components:
- Medical devices
- Concentrator boxes
- Device interfacing software
- Interface engine
- External systems
- Application server
- Workstations
- Database server
- storage subsystem

Medical devices record clinical data from patients such as heart rate or temperature. Medical devices transfer data via a serial connection to concentrator boxes or directly to a device interfacing workstation. A concentrator box is a device that converts serial data from medical devices to the TCP/IP stream. This device has multiple serial ports, which are mapped to corresponding TCP/IP ports. Concentrator boxes are connected to device interfacing workstations.

Device interfacing software handles medical device connectivity. This software component contains device drivers, error logging and logic to send data from medical devices to the interfacing engine. Interface engine is a third party server software component. The Interface engine is responsible for transforming data between different formats. Different data routes and communication points can be specified according to production environment requirements. The interface engine can also be used to parse useless or redundant data out.

External systems send or request data to or from the database, such as the hospital information system or laboratory system. The interface engine handles communications between external systems and application 1. The interface engine can handle external system communications in both applications 1 and 2.

The application server serves client workstation requests and is connected to the database in order to save and request data. In this example it is located in the same physical computer as the database server. Workstations request or add data from or to application server. This includes medical device data from medical devices and documentation added from client workstations.

The database server is a relational model database back-end, that stores all data from the whole system and supplies data to the application server. Storage subsystem stores data in disk drives or group of drives. There are many different storage systems available including storage area network, SAN, network attached storage, NAS or directly attached storage. NAS systems are not used in applications studied in this thesis.

A SAN consolidates large groups of hard drives together. A SAN provides “Virtual hard drives” (logical unit numbers, LUNs) to computers that need to access storage. Computers that use these LUNs see them as though they were local to themselves. SANs use block-level separation of data. SANs are usually used in large production environments, where multiple computers require high performance and storage consolidation. /4/

A NAS uses file-based protocols to access data, such as NFS (Network File System) or SMB. In a NAS it is clear to the storage-using computer that the storage is remote and the computers access data at file-level, not in block level as in a SAN. Directly attached storage is the case when hard drives are directly attached to a computer via a SATA-, SCSI-, or a SAS-bus, like in a normal workstation computer. /4/

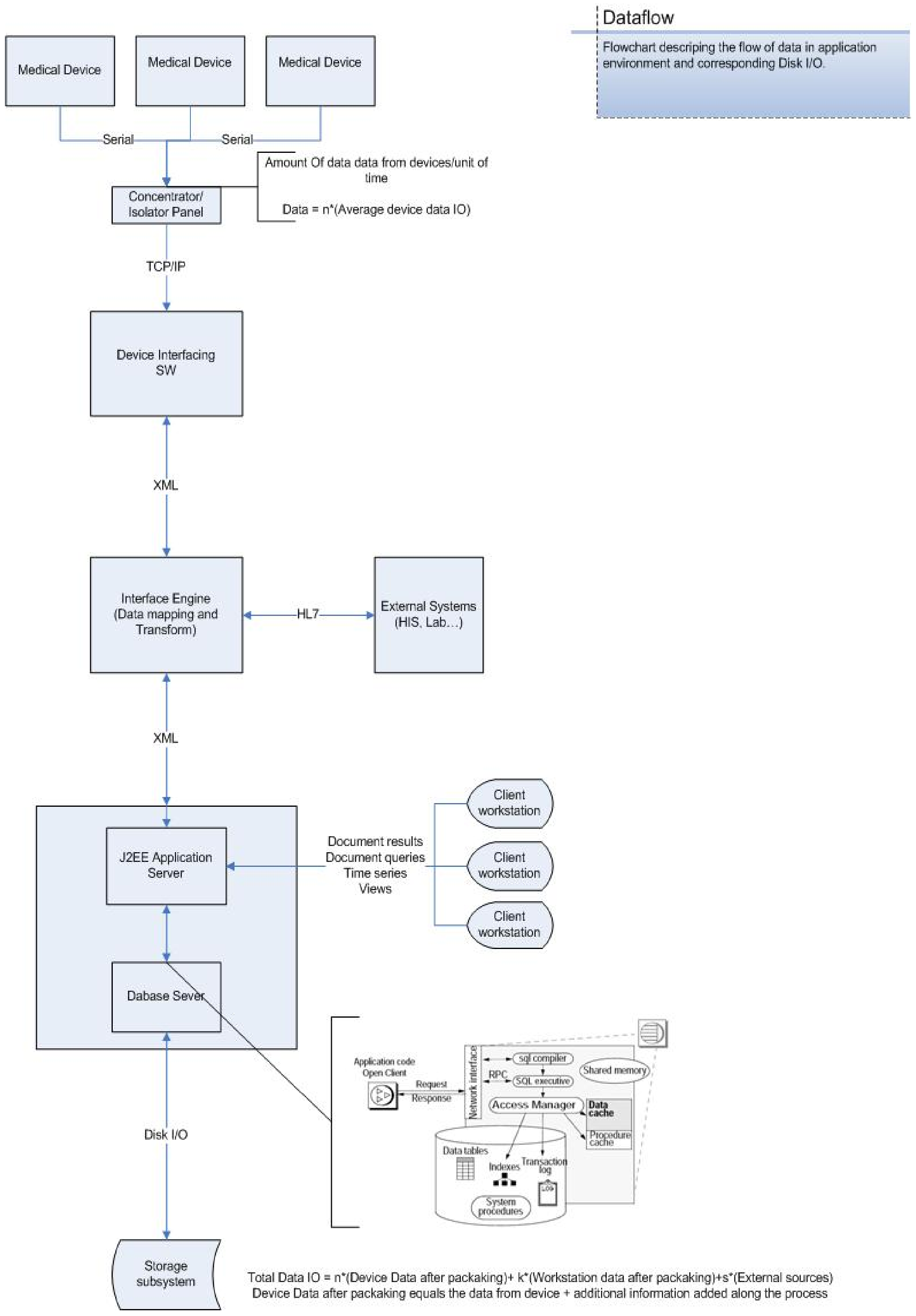

Dataflow
Flowchart descriping the flow of data in application environment and corresponding Disk I/O.
Medical Device
Medical Device
Medical Device
Serial
Serial
Amount Of data data from devices/unit of time
Data = n*(Average device data IO)
Concentrator/ Isolator Panel
TCP/IP
Device Interfacing SW
XML
Interface Engine (Data mapping and Transform)
HL7
External Systems (HIS, Lab…)
XML
J2EE Application Server
Document results
Document queries
Time series
Views
Client workstation
Client workstation
Client workstation
Dabase Sever
Disk I/O
Application code
Open Client
Request
Response
Network interface
sql compiler
RPC
SQL executive
Shared memory
Access Manager
Data cache
Procedure cache
Data tables
Indexes
Transaction log
System procedures
Storage subsystem
Total Data IO = n*(Device Data after packaking)+ k*(Workstation data after packaking)+s*(External sources)
Device Data after packaking equals the data from device + additional information added along the process

Figure 4. Application 1 workflow

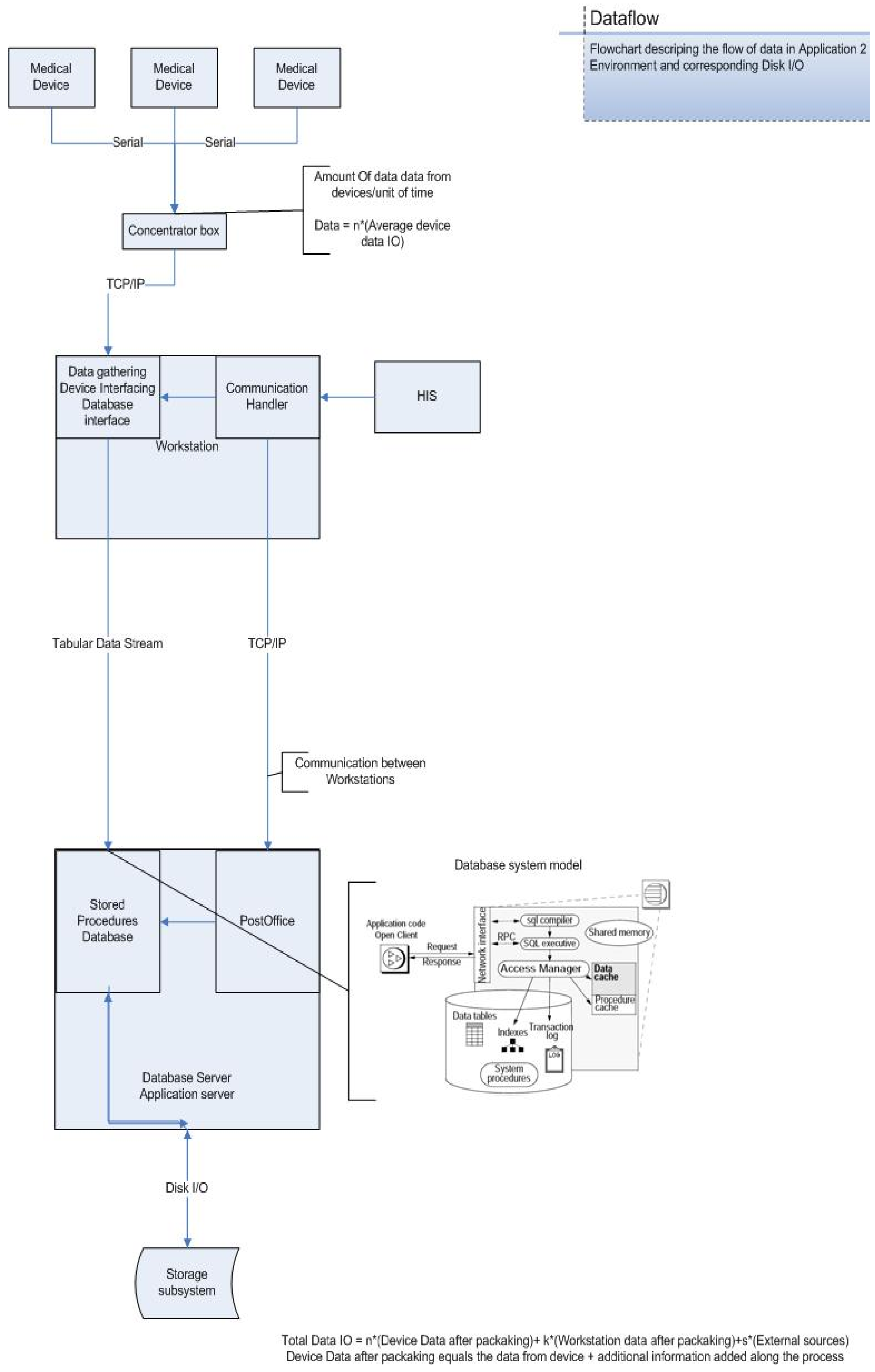
2.2 Application 2

Application 2 environment contains many of the same components that application 1. Some of the components are absent in application 2 and some components have different roles. Application 2 does most of the work in the workstation computer, this is very different from application 1, where most of the work is done in the application server. Application 2 environment consists of the following components:

-Medical devices
-Concentrator boxes
-Interface engine
-External systems
-Application server
-Workstations
-Database server
-Storage subsystem

In application 2 environment device interfacing, data gathering, compression and database interfacing are done in workstation software. Application server handles distribution of workstation software and manages internal post office for inter-workstation communications. The database server uses the same software, but it is configured in a different way to suit application requirements.

Figure 5. Application 2 workflow


Dataflow
Flowchart descriping the flow of data in Application 2 Environment and corresponding Disk I/O
Medical Device
Medical Device
Medical Device
Serial
Serial
Amount Of data data from devices/unit of time
Data = n*(Average device data IO)
Concentrator box
TCP/IP
Data gathering Device Interfacing Database interface
Communication Handler
HIS
Workstation
Tabular Data Stream
TCP/IP
Communication between Workstations
Stored Procedures Database
PostOffice
Database Server Application server
Database system model
Application code Open Client
Request
Response
Network interface
RPC
sql compiler
SQL executive
Shared memory
Access Manager
Data cache
Procedure cache
Data tables
Indexes
Transaction log
LOG
System procedures
Disk I/O
Storage subsystem
Total Data IO = n*(Device Data after packaking)+ k*(Workstation data after packaking)+s*(External sources)
Device Data after packaking equals the data from device + additional information added along the process

3. MEASUREMENT TOOLS

3.1 Application Workflow Monitoring

Application workflow monitoring of application 1 and application 2 will be carried out with windows performance counters and database internal procedure. These monitoring tools provide a wide variety of parameters to measure, such as CPU usage, memory usage, network usage, physical disk usage. In this thesis the most interesting data concerns disk usage.

The reason, why two different tools are used is that the database internal tooling does not provide information about physical disk usage and windows performance counters do not provide information about database internal performance. This chapter provides information on how to configure these tools and helps understand the input and output of these tools. Figure 6 depicts, what tools are used for which parts of the system.

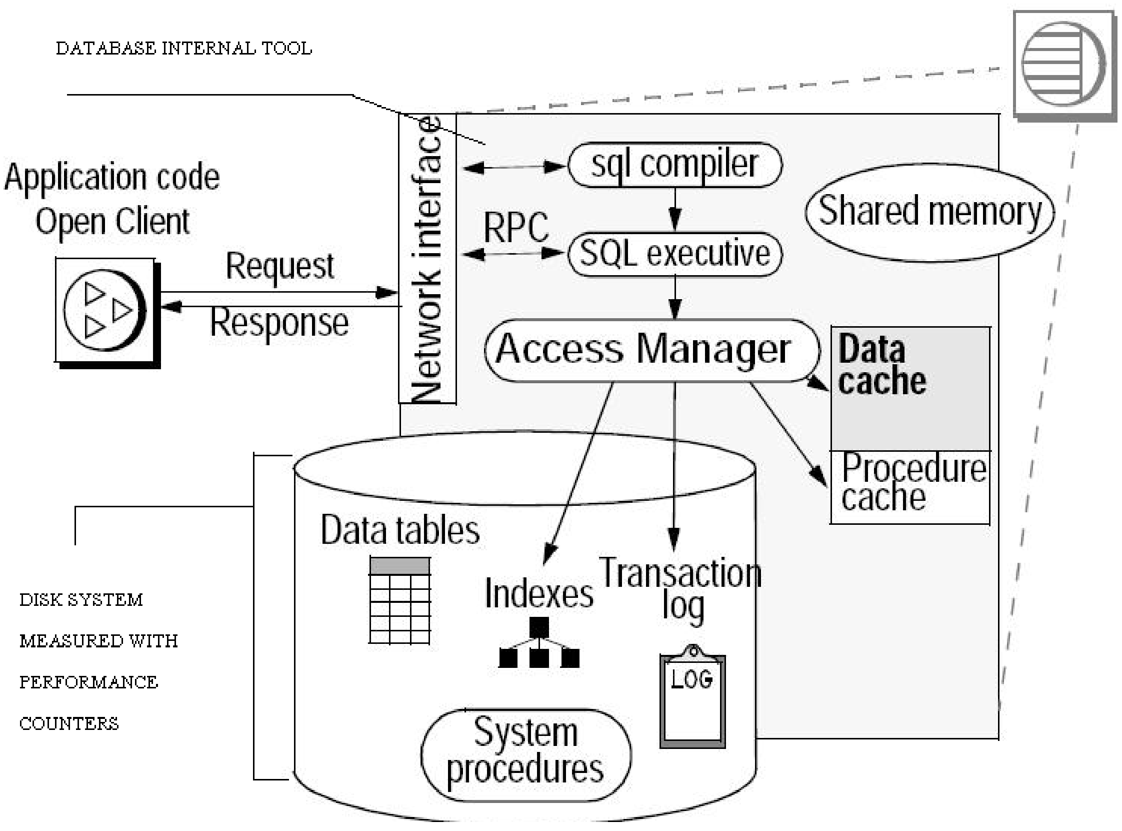
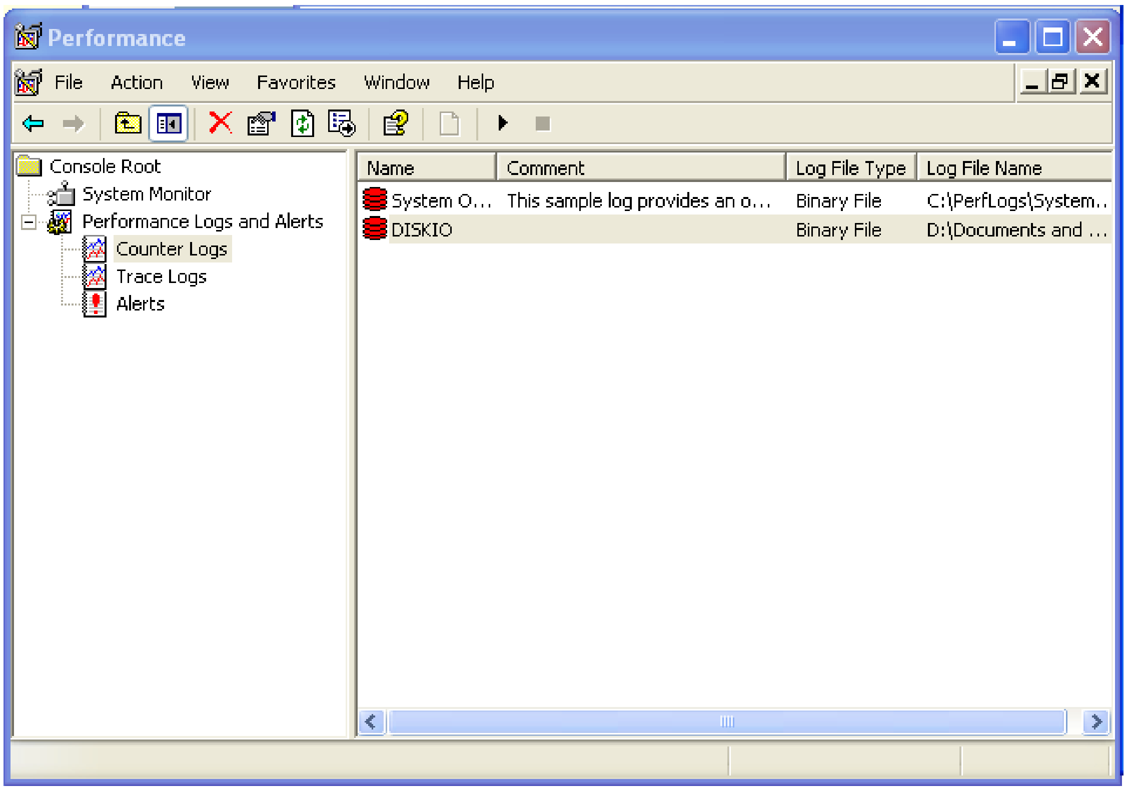


Figure 6. Database internal components. /5/

3.2 Windows Performance Counters

Windows performance counters provide the possibility to monitor the utilization of different hardware and operating system components, for example the CPU usage, memory usage, network activity and storage system utilization. Different counters can be added from the properties menu of a certain counter. Settings parameters, such as sample interval, log file-type, log file location, and monitoring schedule can be given.

In this project the main interest is in the physical disk usage, so in this thesis it is reasonable to concentrate only in that section and provide configuration details about physical disk monitoring.



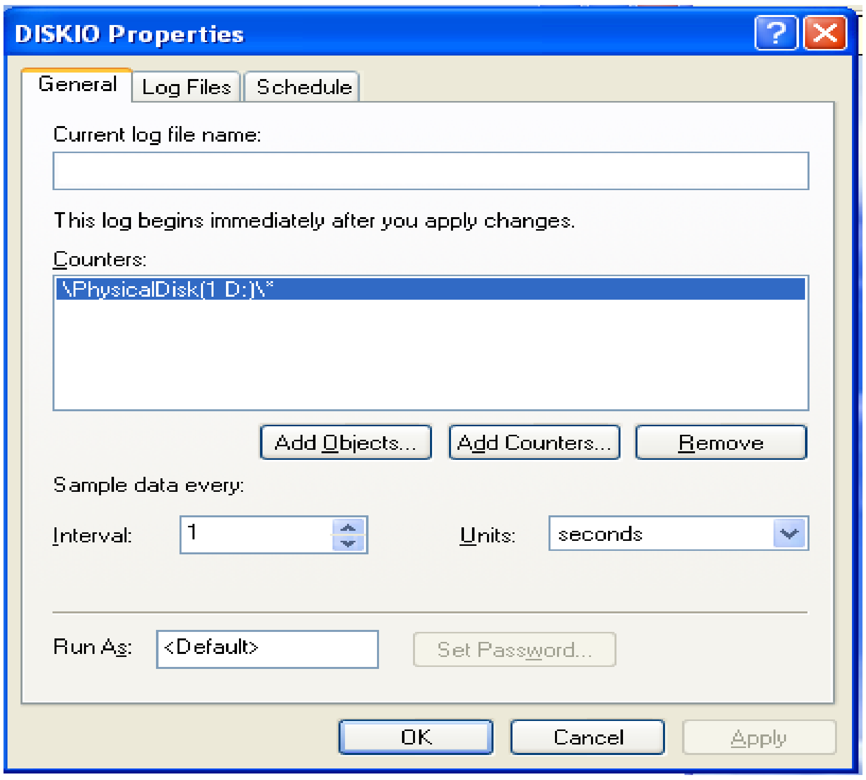
Name	Comment	Log File Type	Log File Name
System O...	This sample log provides an o...	Binary File	C:\PerfLogs\System..
DISKIO		Binary File	D:\Documents and ...

Figure 7. Performance counters

3.3 Commands to Automate Counter Usage

The performance counters can be configured via windows GUI as seen in the Figure 7. However, if performance counters are used repeatedly it is reasonable to create some script tooling to configure the counters automatically. This saves time and ensures that the counters have the same settings always. These commands can be put in a windows batch file, or can be run from a command line. Here are a few examples on how to script counter creation, updating and starting. Figure 8 shows the properties for the created counter.

```
logman create counter DISKIO  -si 1
//Create counter named DISKIO and set sample interval to 1
//second
logman update DISKIO -c "\PhysicalDisk(1 D:)\* -rf  01:00:00
// update DISKIO counter to monitor all available counters
// of physical disk usage of drive
// D: and set test duration of 1 hour
logman query DISKIO
//Shows the properties of counter DISKIO
logman start DISKIO
//Start counter DISKIO
logman stop DISKIO
// Stop counter DISKIO; output will be generated to the
//specified file.
```


DISKIO Properties

Current log file name:

This log begins immediately after you apply changes.

Counters:

\PhysicalDisk(1 D:)*

Sample data every:

Interval: 1

Units: seconds

Run As: <Default>

Figure 8. Performance counter setting

3.4 Database Internal Performance Monitoring

The database internal monitoring procedure provides a comprehensive overview of database performance from the database point of view. The monitoring procedure is initialized with sql-utility from windows commandline. The monitoring procedure asks for the sample time interval as input, it is also possible to specify which parameters are measured. After the sample interval time has passed the results will be outputted to a text file. In this thesis it is reasonable to only concentrate on the subjects that give information about disk subsystem utilization. The following subjects are useful when monitoring disk subsystem utilization: Kernel Utilization, Task Management, Transaction Profile, Transaction Management, Data Cache Management, Disk I/O Management and Network I/O Management. /5/

There is a possibility to monitor data and behavior of the subjects; Application Management, Cache Wizard, Data Cache Management, Disk I/O Management ESP management, Housekeeper Task Activity, Index Management, Kernel Utilization, Lock Management, Memory Management, Metadata Cache Management, Monitor Access to Executing SQL, Network I/O Management, Parallel Query Management parallel, Procedure Cache Management, Recovery Management, Task Management, Transaction Management, Transaction Profile and Worker Process Management. /5/

Kernel utilization shows output for database engine utilization. It outputs data for engine busy time, engine idle time, network I/O checks and disk I/O checks. Checks occur when database engine checks, whether a certain operation is complete, if completed next operation can start. /5/

Task management outputs data about opened connections to database, number of context switches of database engines and task context switching reason. Context switches occur when database engine switches from one user task to another, when it has to wait for some operation to complete for example disk I/O. Context switching reason shows count of context switches for particular reason for example Disk I/O or Network I/O. /5/

Transaction profile outputs the number of transactions occurred during the sample interval. Separate counters exist for different types of transactions. Output shows counts for inserted rows, deleted rows, updated rows and total rows affected. /5/

Transaction management shows data for transaction log activity. It contains data of user log cache flushes to transaction log. The transaction log is a database device that holds data about committed transactions, such as row inserts, deletions and updates. The user log cache is a small cache, which gets flushed to disk usually when transaction commits, but other reasons exist, such as full ULC or change of database. This section outputs counts and reasons for ULC flushing. /5/

Data Cache Management gives data about cache usage, large I/O usage and asynchronous prefetch statistics. Cache usage shows counts of cache misses and cache hits. Cache hits occur when a certain page is found in cache and it does not have to be read from disk. Cache misses occur when a page is not found from cache and has to be read from disk. Large I/O is a read operation from disk that reads several pages at once to improve performance; it is initiated when something is needed from disk. Asynchronous prefetch tries to estimate what data is needed from disk and places that data to cache for possible later use. /5/

Disk I/O management shows per device data about database disk usage. Database devices are a way to separate data over different database files that can be distributed amongst several physical disks or partitions. The output data includes; Number of write operations, number of read operations and number of APF read operations. /5/

Network I/O Management shows the network I/O operations. Items included in the output are; sent packets, sent bytes, received packets, received bytes. Information about network management is useful when examining correlations between network traffic and disk utilization. /5/

3.5 Script Tooling

Part of this project was to generate measurement tools. Without proper script tools it is impossible to start these tools simultaneously, hard to ensure that certain parameters do not change, cumbersome to configure tools separately and to keep track of output files. Below is an overview of a script created to automate measurements. Due to intellectual property protection the complete code is not given, only the steps of the script.

1. Check if the database server is installed, if not, inform the user and exit
2. Give information about the database and the computer
3. Query the registry for available drives to monitor
4. Ask how many hours to run the test, ensures correct time format automatically
5. Ask how many minutes to run the test, ensures correct time format automatically
6. Ask how many seconds to run the test, ensures correct time format automatically
7. Ensure non zero time, if input ask time again
8. Convert measurement time to seconds
9. Ask for the first drive to be monitored
10. Go to 9, if nothing is given here
11. Ask for the second drive to be monitored
12. Ask for test environment description, will be added to filename
13. Test whether performance counter already exists if not create it
14. Communicate testing parameters to the database server
15. Communicate testing parameters to windows performance counters
16. Start database internal performance monitoring and windows performance counters simultaneously.
17. Wait for test run to complete, output files will be put in the same folder and are given descriptive names according to the test environment.

4. TEST RESULTS

The purpose of these tests was to measure the database disk I/O under load by medical devices. The main goals of these tests were to measure how much disk input medical devices create in the database. No exact numbers are given in the results, only comparison between application 1 and application 2.

4.1 The Results

The medical devices send certain parameters to be saved to the database. Some of these are settings parameters, which tell the client user what settings are selected in a certain device and some data parameters that are considered the real payload. These parameters include for example patient heart rate and body temperature. These parameters and their respective sample rates are listed in tables 2 and 3. Sample rates are different in application 1 and 2.

Table 2. Medical Device parameters

Device	Device 1	Device 2	Device 3	Device 4
parameters	12	10	2	2
Data parameters	80,00%	100,00%	100,00%	50,00%
Settings parameters	20,00%	0,00%	0,00%	50,00%

Table 3. Sample rates

Environment	Device 1	Device 2	Device 3	Device 4
Application 1				
Data parameters	36/min	30/min	3/min	6/min
Settings parameters	3/15min			
Application 2				
Data parameters	9,6/2min	10/2min	1/15min	1/15min
Settings parameters	2,4/15min			

The number of disk I/Os shows Number of disk input /output operations by the database as reported by the database performance procedure. This is a mix of 2kB, 4kB and 16kB I/O operations. Writes show the number of write operations by the database as reported by performance procedure. This is a mix of 2k and 4k writes. Reads show the number of read operations by the database as reported by database performance procedure.

Cache hits show the number of times the database was looking for a page in the cache and found it. Cache misses show the number of times the database was looking for a page in the cache but did not find it. This means that the database has to access the disk. A page is the smallest unit of data the database is able to handle. Total write transfer rate is calculated from windows performance counter output report. Network bytes show database engine sent and received bytes.

Table 4. Device 1 results

Environment	# of Disk I/Os	Cache	Network Bytes	Bytes/s total write transfer rate
Application 1	writes 1x reads 1x	hits 1x misses 1x	Received: 1x sent: 1x	1x
Application 2	writes 5,4x reads 1x	hits 0.04 misses 1x	Received: 0,53x sent: 0,12	5,85x

Table 5. Device 2 results

Environment	# of Disk I/Os	Cache	Network Bytes	Bytes/s total write transfer rate
Application 1	writes 1x reads 1x	hits 1x misses 1x	Received: 1x sent: 1x	1x
Application 2	writes 5,89x reads 10x	hits 9,4x misses 1x	Received: 0,86x sent: 0,27x	6,51x

Table 6. Device 3 and 4 results

Environment	# of Disk I/Os	Cache	Network Bytes	Bytes/s total write transfer rate
Application 1	writes 1x reads 1x	hits 1x misses 1x	Received: 1x sent: 1x	1x
Application 2	writes 20,3x reads 10x	hits 3,04x misses 1x	Received: 1,96x sent: 0,005	30,4x

Table 7. Results of all devices combined

Environment	# of Disk I/Os	Cache	Network Bytes	Bytes/s total write transfer rate
Application 1	writes 1x reads 1x	hits 1x misses 1x	Received: 1x sent: 1x	1x
Application 2	writes 5,56x reads 1x	hits 17,5x misses 1x	Received: 1,7x sent: 0,43x	7,26x

4.2 Analyzing the Results

Tables 2 and 3 show the theoretical amount of parameters that application 1 and application 2 save to the database. Below is a calculation, of how many parameters are saved to the database per minute. From this theory it is possible to calculate the data input ratio between these applications.

```
Application 1(total) = (36+30+3+6+0,2)/min = 75,2/minute
Application 2(total) = 4,8+5+(1/15)+(1/15)+(2,4/15) =
10,1/minute
Ratio = Application 1(total)/Application 2(total)
Ratio = (75,2/minute)/(10,1/minute) = 7,4
```

The amount of database activity and disk write transfer rate can be seen in tables 4 to 7. The most interesting data when concentrating on the disk input is on the "# of disk I/Os" and "Bytes/s total write transfer rate". These readings report the bulk data transfer to the database and finally to the disk subsystem. Cache and network statistics data is more useful when analyzing database read operations.

Some comparison of theory and practice can be calculated from these results. Below is a calculation of how close the theoretical database input is to the measured input. Comparison is taken from above calculations and from Table 7. Results of all devices combined.

Theoretical and measured ratios between application 1 and application 2:

```
Theoretical database input ratio: 7,4
Measured database input ratio: 7,26

Difference:  (7,4-7,26)/7,4*100% = 1,9%
```

From the calculations it is apparent that theory and actual measurements correspond well. This is logical when only the device sample rates change between application 1 and 2.

5. PERFORMANCE MEASUREMENT BY SQLIO

In earlier stages of the project, the storage subsystem usage under certain test scenarios was analyzed. In order to put the storage subsystem usage into context it is important to measure the actual performance of a certain system. The performance measurement should simulate the I/O patterns of application 1 and application 2. This is done to find out what performance can be achieved from a certain disk subsystem in these two application environments.

5.1 Microsoft Sqlio

Sqlio is a free tool by Microsoft that is designed to simulate storage subsystem I/O workload of a Microsoft SQL Server. Although it is designed for Microsoft SQL Server it has been proven a useful simulation tool for other databases as well. Before using this tool it is recommended to have at least some understanding of the storage subsystem. Many variations of different block sizes, queue depths, thread amounts, I/O directions and I/O types should be tested to get comprehensive data about possible storage bottlenecks. Figure 9 shows a system to be tested with Sqlio, arrows in the figure represent different I/O patterns. /6/

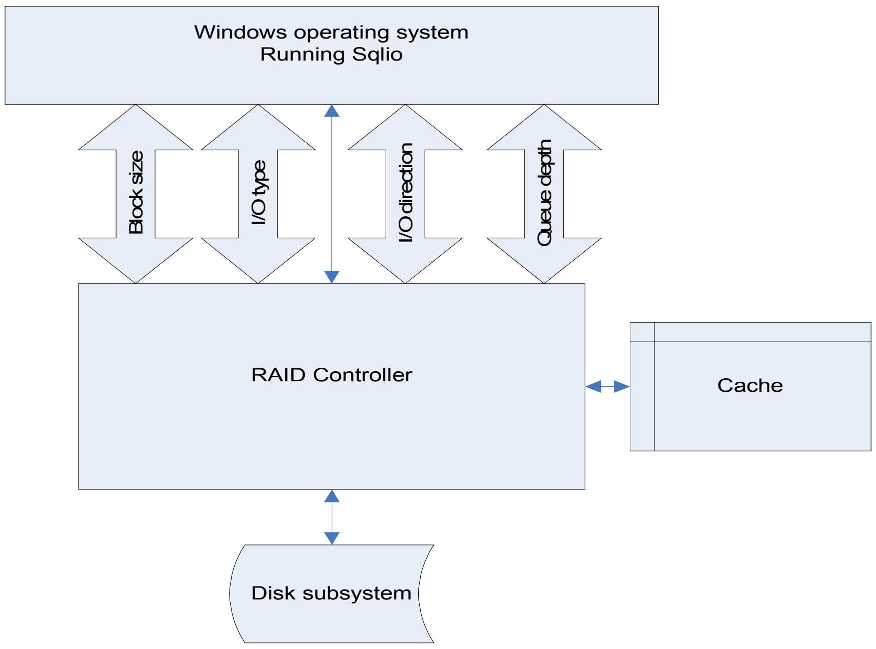


Figure 9. System analyzed by Sqlio

5.2 Sqlio Parameters

These are some of the useful parameters to be included in Sqlio tests. These parameters determine the behavior, output and test duration of Sqlio. After analyzing the disk I/O generated by medical devices as shown before, it is reasonable to tune Sqlio to generate similar disk I/O patterns. Tuning Sqlio parameters can generate different mixes of I/O patterns. The most important things to tune are; type of I/O (sequential or Random), block size, read/write, queue depth. Different parameters are shown in table 8. These are used in the sample script. /6/

Table 8. Sqlio parameters

Parameter	Description
-o	The number of outstanding I/O requests. Increasing the queue depth may increase total throughput. Common values are 8, 32 and 64.
-t	Number of simultaneous threads that issue I/O to the testfile specified in param.txt. The –o parameter specifies per thread queue.
-LS	Record latency information of the storage subsystem. It is recommended to capture this in the testing.
-k	Select either R or W(Read or Write). It is recommended to simulate a mix of reads and writes in the testing.
-s	Duration of test in seconds. It is recommended to run 5-10 minutes per I/O size to get a good idea of I/O performance.
-b	Size of the I/O requests in bytes(block size).
-f	Type of I/O. either random or sequential It is recommended to simulate a mix of random and sequential I/O in the testing.
-F	param.txt file which contains location and size of the file, where the I/O will be issued to.
Testfile.dat	File, where the I/O will be issued. Consider the right size of this file, should be at least 2 times bigger than the storage subsystem cache (SAN cache, NAS cache or hard drive cache). This file will be created to a location specified in param.txt
Timeout /T	Time duration to wait between tests. Lets the storage subsystem “rest” before running another I/O pattern. This esures more reliable results

5.3 Sample Script

This is a sample script of Sqlio to be copied to the run.bat file. This script contains tests for both write and read operations in random and sequential pattern when different block sizes are used (2, 4, 8 and 16kB). Every line represents a certain I/O pattern, all lines output performance metrics to the output file. The time parameter –s should be sufficiently large in order to get more accurate results from the testing, this is because some disk subsystems need time to tune for different I/O patterns.

```
sqlio -kW -s360 -frandom -o2 -b2 -LS -Fparam.txt
timeout /T 10
sqlio -kW -s360 -frandom -o2 -b4 -LS -Fparam.txt
timeout /T 10
sqlio -kW -s360 -frandom -o2 -b8 -LS -Fparam.txt
timeout /T 10
sqlio -kW -s360 -frandom -o2 -b16 -LS -Fparam.txt
timeout /T 10

sqlio -kW -s360 -fsequential -o2 -b2 -LS -Fparam.txt
timeout /T 10
sqlio -kW -s360 -fsequential -o2 -b4 -LS -Fparam.txt
timeout /T 10
sqlio -kW -s360 -fsequential -o2 -b8 -LS -Fparam.txt
timeout /T 10
sqlio -kW -s360 -fsequential -o2 -b16 -LS -Fparam.txt
timeout /T 10

sqlio -kR -s360 -frandom -o2 -b2 -LS -Fparam.txt
timeout /T 10
sqlio -kR -s360 -frandom -o2 -b4 -LS -Fparam.txt
timeout /T 10
sqlio -kR -s360 -frandom -o2 -b8 -LS -Fparam.txt
timeout /T 10
sqlio -kR -s360 -frandom -o2 -b16 -LS -Fparam.txt
timeout /T 10
```

```
sqlio -kR -s360 -fsequential -o2 -b2 -LS -Fparam.txt
timeout /T 10
sqlio -kR -s360 -fsequential -o2 -b4 -LS -Fparam.txt
timeout /T 10
sqlio -kR -s360 -fsequential -o2 -b8 -LS -Fparam.txt
timeout /T 10
sqlio -kR -s360 -fsequential -o2 -b16 -LS -Fparam.txt
```

5.4 Sample Output of Sqlio

Below is a sample sqlio output. In the first line the used parameters can be seen. After "`CUMULATIVE DATA:`" is the most important performance data. This data shows the number of I/O operations, total throughput and latencies that can be achieved with given parameters.

When examining sqlio output some trends in performance can be noted. Usage of small block sizes (-b parameter) typically results in lower overall throughput, when mechanical hard drive systems are tested. This is also true if the testfile fits in the RAID controller cache (Figure 9). This scenario is called a "cached run", where the actual data is not read or written to the disk but accessed from the faster cache-RAM. This is an unwanted situation, which will result in too high performance data.

```
C:\PROGRA~1\SQLIO>sqlio -kW -s360 -frandom -o8 -b1 -LS -Fparam.txt
sqlio v1.5.SG
using system counter for latency timings, -1102787296 counts per
second
parameter file used: param.txt
 file c:\testfile.dat with 2 threads (0-1) using mask 0x0 (0)
2 threads writing for 360 secs to file c:\testfile.dat
 using 1KB random IOs
 enabling multiple I/Os per thread with 8 outstanding
size of file c:\testfile.dat needs to be: 104857600 bytes
current file size: 0 bytes
need to expand by: 104857600 bytes
expanding c:\testfile.dat ... done.
using specified size: 100 MB for file: c:\testfile.dat
initialization done
```

```
CUMULATIVE DATA:
throughput metrics:
IOs/sec:   311.18
MBs/sec:     0.30
latency metrics:
Min_Latency(ms): 0
Avg_Latency(ms): 50
Max_Latency(ms): 341
```

5.5 Data Extraction and Analysis

If extensively long test runs are used it is useful to create some kind of scripting for data mining. The most interesting data is after "`CUMULATIVE DATA:`" below is a small batch script to extract this data and place it in a spreadsheet file for later processing. This script finds all lines containing for example "IO/sec:", and places the whole line to a new row to an xls-file.

```
@echo off > results.xls
for %%T in (output.txt) do find "IOs/sec:" <%%T>> results.xls
for %%T in (output.txt) do find "MBs/sec:" <%%T>> results.xls
 for %%T in (output.txt) do find "Min_Latency(ms):" <%%T>> results.xls
for %%T in (output.txt) do find "Avg_Latency(ms):" <%%T>> results.xls
for %%T in (output.txt) do find "Max_Latency(ms):" <%%T>> results.xls
REM find something and place it to a xls-file
```

Understanding the results of Sqlio is important. In the output of Sqlio it can be useful to look for a few cases, where the total throughput (MBs/sec) is the highest and the average latency (Avg_Latency) is reasonably low. After finding these cases, record the parameters that were used to achieve these results. In the future testing can be done with these parameters only, shortening the time needed for test runs. /7/

CONCLUSION

In this project it has been noted that understanding application workflow is important, when analyzing storage subsystem usage under different scenarios. Without studying the application environment it is impossible to create different use-cases to simulate database and disk activity. Studying the theory of database activity in different application environments is important, when comparing theoretical workflow to actual measured workflow.

Measurement tools and scripts are a crucial asset to this kind of project. Understanding the output of different tools is as important as the results themselves. Scripting is useful when test are run in different computers, they ensure the synchronous start of different monitoring tools, provide way to set parameters easily, label output files and gather data to a common location.

Finally it is important to compare theoretical data to the measured data and draw some conclusion about it. After analyzing measurement data it can be useful to measure and analyze the performance of some storage subsystems to put the whole data into context.

REFERENCES

[1] GE Healthcare website (Online) (10 April, 2010)
http://www.gehealthcare.com

[2] Passmark Software Performance Test Suite (Free Software) (10 April, 2010)
http://www.passmark.com/

[3] A. W. Burks, H. Goldstine, J. V. Neumann Preliminary Discussion of the Logical Design of an Electronic Computing Instrument, 1946. (15 May, 2010)

[4] Free Internet encyclopedia (Online) (6 April, 2010)
http://en.wikipedia.org/wiki/Storage_area_network

[5] Sybase Books (Online PDF) (15 April, 2010)
http://infocenter.sybase.com

[6] Microsoft SQLIO Disk Subsystem Benchmark Tool (Online) (5 April, 2010)
http://www.microsoft.com/downloads/details.aspx?familyid=9a8b005b-84e4-4f24-8d65-cb53442d9e19&displaylang=en

[7] Website about Microsoft Sqlio (Online) (6 April, 2010)
http://www.brentozar.com/archive/2008/09/finding-your-san-bottlenecks-with-sqlio