The South African Institute for Computer Scientists and Information Technologists

ANNUAL RESEARCH AND DEVELOPMENT SYMPOSIUM

23-24 NOVEMBER 1998
CAPE TOWN
Van Riebeeck Hotel in Gordons Bay

Hosted by the University of Cape Town in association with the CSSA,
Forchefstreem University for CHE and
The University of Natal

PROCEEDINGS

EDITED BY
D. FEIKOV AND L. VENTER

SPONSORED BY

ABSA Group
The South African Institute for Computer Scientists and Information Technologists

ANNUAL RESEARCH AND DEVELOPMENT SYMPOSIUM

23-24 NOVEMBER 1998
CAPE TOWN
Van Riebeeck hotel in Gordons Bay

Hosted by the University of Cape Town in association with the CSSA, Potchefstroom University for CHE and The University of Natal

GENERAL CHAIR: PROF G. HATTINGH, PU CHE

PROGRAMME CO-CHAIRS:
PROF. L VENTER, PU CHE (Vaal Triangle), PROF. D. PETKOV, UN-PMB

LOCAL ORGANISING CHAIR: PROF. P. LICKER, UCT - IS

PROCEEDINGS

EDITED BY
D. PETKOV AND L. VENTER

SYMPOSIUM THEME:
Development of a quality academic CS/IS infrastructure in South Africa

SPONSORED BY
FOREWORD

The South African Institute for Computer Scientists and Information Technologists (SAICSIT) promotes the cooperation of academics and industry in the area of research and development in Computer Science, Information Systems and Technology and Software Engineering. The culmination of its activities throughout the year is the annual research symposium. This book is a collection of papers presented at the 1998 such event taking place on the 23\textsuperscript{rd} and 24\textsuperscript{st} of November in Gordons Bay, Cape Town. The Conference is hosted by the Department of Information Systems, University of Cape Town in cooperation with the Department of Computer Science, Potchefstroom University for CHE and and Department of Computer Science and Information Systems of the University of Natal, Pietermaritzburg.

There are a total of 46 papers. The speakers represent practitioners and academics from all the major Universities and Technikons in the country. The number of industry based authors has increased compared to previous years.

We would like to express our gratitude to the referees and the paper contributors for their hard work on the papers included in this volume. The Organising and Programme Committees would like to thank the keynote speaker, Prof M.C Jackson, Dean, University of Lincolnshire and Humberside, United Kingdom, President of the International Federation for Systems Research as well as the Computer Society of South Africa and The University of Cape Town for the cooperation as well as the management and staff of the Potchefstroom University for CHE and the University of Natal for their support and for making this event a success.

Giel Hattingh, Paul Licker, Lucas Venter and Don Petkov
Table of Contents

Lynette Drevin: Activities of IFIP wg 11.8 (computer security education) & IT related ethics education in Southern Africa

Reinhardt A. Botha and Jan H.P. ElofT: exA Security Interpretation of the Workflow Reference Model

Willem Krige and Rossouw von Solms: Effective information security monitoring using data logs


Carl Papenfus and Reinhardt A. Botha: A shell-based approach to information security

Walter Smuts: A 6-Dimensional Security Classification for Information

Philip Machanick and Pierre Salverda: Implications of emerging DRAM technologies for the RAM page Memory hierarchy

Susan Brown: Practical Experience in Running a Virtual Class to Facilitate On-Campus Under Graduate Teaching

H.D. Masethe, T.A Dandadzi: Quality Academic Development of CS/IS Infrastructure in South Africa

Philip Machanick: The Skills Hierarchy and Curriculum

Theda Thomas: Handling diversity in Information Systems and Computer Science Students: A social Constructivist Perspective

Udo Averweg and G J Erwin: Critical success factors for implementation of Decision support systems

Magda Huisman: A conceptual model for the adoption and use of case technology

Paul S. Licker: A Framework for Information Systems and National Development Research

K. Niki Kunene and Don Petkov: On problem structuring in an Electronic Brainstorming (EBS) environment
Derek Smith: Characteristics of high-performing Information Systems Project Managers and Project Teams

Lucas Venêr: INSTAP: Experiences in building a multimedia application

Scott Hazelhurst, Anton Fatti, and Andrew Henwood: Binary Decision Diagram Representations of Firewall and Router Access Lists

Andre Joubert and Annelie Jordaan: Hardware System interfacing with Delphi 3 to achieve quality academic integration between the fields of Computer Systems and Software Engineering

Borislav Roussev: Experience with Java in an Advanced Operating Systems Module

Conrad Mueller: A Static Programming Paradigm

Sipho Langa: Management Aspects of Client/Server Computing

T Nepal and T Andrew: An Integrated Research Programme in AI applied to Telecommunications at ML Sultan Technikon

Yuri Velinov: Electronic lectures for the mathematical subjects in Computer Science

Philip Machanick: Disk delay lines

D Petkov and O Petkova: One way to make better decisions related to IT Outsourcing

Jay van Zyl: Quality Learning, Learning Quality

Matthew O Adigun: A Case for Reuse Technology as a CS/IS Training Infrastructure

Andy Bytheway and Grant Hearn: Academic CS/IS Infrastructure in South Africa: An exploratory stakeholder perspective

Chantel van Niekerk: The Academic Institution and Software Vendor Partnership

Christopher Chalmers: Quality aspects of the development of a rule-based architecture

Rudi Harmse: Managing large programming classes using computer mediated communication and cognitive modelling techniques
Michael Muller: How to gain Quality when developing a Repository Driven User Interface

Elsabe Cloete and Lucas Venter: Reducing Fractal Encoding Complexities

Jean Bilbrough and Ian Sanders: Partial Edge Visibility in Linear Time

Philip Machanick: Design of a scalable Video on Demand architecture

Freddie Janssen: Quality considerations of Real Time access to Multidimensional Matrices

Machiel Kruger and Giel Hattingh: A Partitioning Scheme for Solving the Exact k-item 0-1 Knapsack Problem

Ian Sanders: Non-orthogonal Ray Guarding

Fanie Terblanche and Giel Hattingh: Response surface analysis as a technique for the visualization of linear models and data

Olga Petkova and Dewald Roode: A pluralist systemic framework for the evaluation of factors affecting software development productivity

Peter Warren and Marcel Viljoen: Design patterns for user interfaces

Andre de Waal and Giel Hattingh: Refuting conjectures in first order theories

Edna Randiki: Error analysis in Selected Medical Devices and Information Systems
MANAGEMENT ASPECTS OF CLIENT/SERVER COMPUTING

S. Langa

Department of Computer Studies, Mangosuthu Technikon, P.O. Box 12363, Jacobs, 4026
Tel. (031) 9077158 Fax (031) 9077145 Email slanga@julian.mantec.ac.za

Abstract

The purpose of the research is to investigate the management aspects of client/server computing. The latter is seen as a prerequisite for better management of all aspects of client/server computing. Noting the fact that the client/server environment often consists of heterogeneous hardware and software products imposing demands on management, an analysis of different types of client/server architectures according to network and system management is included as one of the subgoals of the research. A literature survey was used to carry out this analysis.

The different client/server architectures are surveyed in order to determine difference in maintenance cost and investigating the need for network and system management tools as a means of minimising labour costs which are brought about by labour-intensive maintenance approach. This survey is accomplished through the use of questionnaires various organisations. Given the fact that the client/server environment is complex in that it contains equipment and software from different vendors and such complexity makes it difficult to properly account for availability, the issue of client/server system availability was also investigated. It provided insights into the information requirements towards better management of such installations. Due to the fact that many organisations do not keep outage data which is important in measuring client/server system availability, an automated database system that assist in recording outage data was designed.

Introduction

In simple terms, client/server computing can be defined as a computerised system that consists of two basic components: a client and a server. The client requests a service to be performed. This service may be to provide data, print, or fax a document. The server provides the requested services for clients. Both clients and servers may reside in the same computer or in two or more independent computers connected by a network. This allows tasks to be distributed over several systems, and to be carried out by those machines which are most suitable for task execution.

The client/server concept is based on a conversation between the client and the server. A client passes a request on to the server and the server performs the task. The server takes action only after having received a request from the client. The server tasks do not have to run on different computers which may include printer, file, database, application, communication, transaction, multimedia, or batch servers. Servers are, in turn, able to activate other servers, given the fact that servers can send requests to other servers (Zantinge, 1996:1-13).

The client/server system’s components must conform to some basic principles if they are to interact properly. These principles are applicable to clients, servers, and network. The principles which form the foundation on which client/server systems are built include: hardware and software independence, open access to services, process distribution, and standards.

- The client includes hardware and software components. Desirable client hardware and software features include: PC with powerful CPU, an operating system that is capable of multitasking, a graphical user interface, and capability of connecting and accessing multiple network operating systems.

- The server also has hardware and software components. Desirable server hardware features include: a fast CPU, possess fault tolerant capabilities, expandability of CPU, memory, disk and peripherals, bus support
for multiple add-on boards, and multiple communications options. The software component includes the server application which runs on top of the operating system and interacts with network operating system to "listen" for the client’s requests for services.

The network, through the communications middleware, provides the means through which clients and servers communicate to perform specific actions. The middleware is made up of several layers of software that aid the transmission of data and control information between client and servers (Coronel, 1996: 614-670).

The client/server systems are implemented using different architectures depending on how an organisation want to work with it and what the best solution is for the organisation. Since the different client/server architectures have different impact on network and system management, the next section aims at investigating the different architectures in the light of network and system management.

**Client/Server Architectures**

**Two-Tier Architectures**

Two-tier architecture consists of three components distributed in two layers; client (requester of service) and server (provider of services). The three components are:-

1. User system interface (such as sessions, text input, dialogue, and display management services).
2. Processing management (such as process development, process enactment, process monitoring, and process resource services).
3. Database management (such as data and file services).

The two-tier design allocates the user system interface exclusively to the client. It places database management on the server and splits the processing management between client and server, creating two layers.

In general, the user interface client invokes services from the database management server. In many two-tier designs, most of the application portion of processing is in the client environment. The database management server usually provides the portion of the processing related to accessing data. Clients commonly communicate with the server through SQL statements or a call-level interface.

The two-tier architecture will scale up to service 100 users on a network. It appears that beyond this number of users, the performance capacity is exceeded. This is because the client/server exchange “keep alive” messages continuously, even when no work is being done, thereby saturating the network (Schussel, 1996: 2-6). Network traffic must be measurable and adjustable so that performance remains good (Zantige, 1996: 614-670).

Implementing business logic in stored procedures can limit scalability because as more application logic is moved to the data management server, the need for processing power grows. Each client uses the server to execute part of its application code, and this ultimately reduces the number of users that can be accommodated.

The two-tier architecture limits interoperability by using stored procedures to implement complex processing logic (such as managing distributed database integrity) because stored procedures are normally implemented using a commercial database management system (DBMS) proprietary language. This means that to change or interoperate with more than one type of DBMS, application may need to be rewritten. Moreover, DBMS’s proprietary languages are generally not as capable as standard programming languages in that they do not provide a robust programming environment with testing and debugging, version control, and library management capabilities.

Two-tier architectures can be difficult to administer and maintain because when applications reside on the client, every upgrade must be delivered, installed, and tested on each client. The lack of uniformity in the client configurations and lack of control over subsequent configurations changes increase administrative workload.
Three-Tier Architecture

The three-tier architecture emerged in the 1990s to overcome the limitations of the two-tier architecture. The third-tier, or the middle tier server, is between the user interface (client) and the data management where business logic and rules are executed and can accommodate hundreds of users by providing functions such as queuing, process monitoring, application execution, and database staging. The three-tier architecture is used when an effective distributed client/server design is needed that provides increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed process from the user (Schussel, 1996: 2-6).

A three-tier distributed client/server architecture includes a user interface tier where user services (such as session, text, input, dialogue, and display management) reside. The middle tier provides process management services (such as process development, process enactment, process monitoring, and process resourcing) that are shared by multiple applications. The third tier provides database management functionality. There are variety of ways of implementing the middle tier, such as transaction processing (TP) monitors, message servers, application servers, Internet.

The third tier, database management, is dedicated to data and file services that can be optimised without using any proprietary DBMS language. It also ensures that the data is consistent throughout the distributed environment through the use of features such as data locking, consistency and locking. The middle tier server (or the application server) improves performance, flexibility, maintainability, reusability, and scalability by centralising process logic. Centralised process logic makes administration and change management easier by localising system functionality so that changes must only be written once and placed on the middle tier server to be available throughout the system (Eckerson, 1995: 3).

In addition, the middle process management tier controls transactions and asynchronous queuing to ensure reliable completion of transactions (Schussel, 1996: 2-6). The middle tier manages distributed database integrity by the two phase commit process. It provides access to resources based on names instead of locations, and thereby improves scalability and flexibility as system components are added or moved (Edelstein, 1995: 34).

Building three-tier architecture is complex work. Programming tools that support the design and deployment of three-tier architectures do not yet provide all of the desired services needed to support a distributed computing environment. A potential problem in designing three-tier architecture is that the separation of user interface logic, process management logic, data logic is not always obvious. Some process management logic may appear on all three tiers. The placement of a particular function on a tier should be based on criteria such as ease of development and testing, ease of administration, scalability of servers, and performance (including both process and network load) (Edelstein, 1995: 34).

Maintenance is an important aspect of client/server system management. In the next section, the use of management tools as a means of minimising maintenance costs is investigated. A analysis of the survey that was conducted through questionnaires in several organisations done to complement the investigation.

Maintenance Costs

Maintenance is an important aspect of the cost evaluation of a client/server environment. A number of fourth generation languages 4GL suppliers, including Oracle, Informix, Sybase, Ingres, and many others, have begun to tune their development environments to meet the requirements of client/server environments. Separate servers for the execution of most functions. Changes in operating systems or user interfaces have already been taken care of by these suppliers, which leads to a decrease in maintenance costs.

Client/server will increase your maintenance costs due to the large number of hardware and software systems dependencies. Maintenance costs of client/server can be controlled only if you are using a good architecture, standards, and management tools. Network and system management tools are a must in a client/server environment (Zantige, 1996: 167-198). Noting these "golden rules", a closer look at network and system management tools is taken in the following section.
Network Management Tools

A good network management system can display the network on screen and systems can be managed remotely. All system data can be stored in a database on a network which works with Simple Network Management Protocol (SNMP). The SNMP protocol was built for current network management tasks and is easy to work with. All parts of the network can be made visible on screen, and they can be accessed remotely. With the help of network management tools, the whole environment can be managed from one or more central points. If an error occurs on the network for instance, (using a network tool) the manager can execute a number of checks, such as: Are the hubs functioning correctly? In which segment is the breakdown? In order to get this information, the manager must resort to remote system management tools.

System Management Tools

In a client/server environment one will find many dependencies between hardware and software products, and to manage this is difficult. To overcome this problem, one can use management tools. System management controls and manages the client/server hardware and software environment. All the rules and actions related to hardware and software products and the applications within the organisation must be established and have to be automated. A good quality control in configuration management and the support of software distribution tools and version control tools, will result in reduced labour costs.

Another important issue is the use of right tools. Some tools are related to network management tools while others can be connected to these tools. Although client/server environments do need such tools, but not every tool is suitable for each client/server system. Any action taken by one tool may result in action from another tool. If these tools cannot work together one will need to write ones own connection and maintain that for each new version. Some other tools are used in PC LAN environment, but they are unable to control client/server environment with intelligent hubs, routers and other equipment, and operating systems. They are more networking tools than system management tools.

The problem that exists now is that all the different system management products are unable to work together or need a specific vendor-dependent connection. This problem can be solved if one uses international standards like CORBA and if the system suppliers have committed themselves to this standard (Zantige, 1996: 167-198).

Survey Analysis:-

- The number of organisations operating a three-tier architecture is 67%.
- An average number of PC workstations connected to the system is 1067.
- All organisations surveyed do use network and system management tools.
- 33% of the organisations use more than one such tools and do not work together.
- None of the tools cannot manage the system remotely.
- None of the is able to display all parts of the system on screen.
- Average number of employees responsible for network and system management is 14.
- 67% have no computerised or manual system for the collection of data reflecting the performance of the network and its components.
- 67% considered bandwidth utilisation necessary in order to better monitor facilities and improve their management.

One of the primary aims of network and system management tools is to keep the client/server system up and running at all times. This brings to the fore the most crucial issue of client/server availability which is investigated in the following section.

Client/Server Availability

With all the complexity that client/server systems have, one might expect system availability to be very poor. Client/server availability is a complex issue due to the many possible configurations of client/server environments and failure modes of client, server, and network devices. Such complexity makes it difficult to properly account for availability in client/server architectural design.
Measuring Client/Server Availability

It is hard to define a system failure. For instance, if one PC fails, its user cannot access the system, but all other users can continue operating (unless the PC failed in a mode that causes an outage for other users). If one out of 2000 users cannot access the system, is the system down? What about 10 out of 2000, or 200 out of 2000? All outages do not have the same impact from a business point of view, if they are not of equal duration. In order to accurately reflect an outage’s impact, one must measure it from the point of view of the user rather than the system. Therefore, an appropriate client/server availability measurement needs to include both the outage duration and the number of users affected (Wood, 1995: 41-48).

Client/Server Outage Data

In his article, Alan Wood presented a methodology to help design engineers evaluate client/server availability. One difficulty that was met when trying to measure client/server availability was the fact that very few companies kept useful outage data especially the single-client outages. It is for this reason that an automated database system that assists in recording outage data and measuring client/server availability has been designed based on Alan Woods methodology.

Outage data is classified into five categories:
- Physical faults of physical component, such as a CPU or power supply.
- Design errors in hardware and software including software failures.
- Operations errors caused by operations personnel intentionally or unintentionally.
- Environmental problems such as power or cooling-system failures, natural disaster, or accidents.
- Reconfiguration (planned) outages, including maintenance and configuration changes.

From the outage data, a list of outage causes for a typical client/server environment is developed. Most of these causes apply to any type of computing environment, but some are propagating (a failure affects more than the users or equipment directly involved) type outages peculiar to the client/server computing environment. The client/server outage data is used to determine the number of outage minutes that should be ascribed to each outage cause. Table 1 below contains five outage categories for each type of client/server equipment that may be connected in a typical client/server configuration.

The system allows a user to enter the duration (in minutes) of the outage, the number of units or users in a system, and the number of users affected. Then the total outage duration for each outage cause is calculated and for each category. Finally the system produces percentage user availability, a graphical representation of user outage minutes by equipment type and by outage category.

References


