

Scalable Data Storage For Public Information System Middleware MidArc

Ivan Benc

Ericsson Nikola Tesla d.d., Krapinska 45
Zagreb, 10000, Croatia
ivan.benc@etk.ericsson.se

and

Franjo Plavec

Electrical and Computer Engineering Department, University of Toronto,
10 King's College Road, Toronto, Ontario, M5S3G4, Canada
<http://www.eecg.toronto.edu/~plavec/>

and

Sinisa Srblijic

School of Electrical Engineering and Computing, University of Zagreb
Unska 3, Zagreb, 1000, Croatia
<http://www.zemris.fer.hr/~sinisa/>

ABSTRACT

Recent development in Internet shopping and e-commerce leads to large number of services available on the Internet. Many of these services require a set of basic functionalities in order to work. Basic functionalities include user identification and authentication, access control, billing, and user profile management. Service programmers need to implement these functionalities repeatedly for each service implementation.

MidArc application middleware implements basic functionalities and enables services to use them through interfaces, which simplifies the service development process. Implementation of basic functionalities requires a robust storage system. Such storage system needs to offer satisfactory response times and a scalable architecture.

In this paper, we propose three designs of the storage system. The first storage system solution is based on a relational database, the second uses combination of relational database and a directory, while the third upgrades the second with caches and buffers. All storage system solutions implement the same interface and can be used interchangeably. Measurements are performed that compare the behavior of proposed storage system solutions in diverse environments.

Keywords: application middleware, storage system, relational database, directory, and scalability

1. INTRODUCTION

Nowadays one of the most valuable assets is information. Having right information at the right

moment is crucial for many businesses. In the last decade, Internet became the greatest repository of information. Some of the information available is useless, or can even be considered harmful. Obtaining the correct, useful information isn't always easy. There is a growing business related to the process of obtaining, processing, and allowing access to the information. Access to the information and its processing is implemented through services.

Services vary in type of information they provide. However, many services require a set of common basic functionalities. For example, services need to register and authenticate users, control access to information through authorization, and store user, authorization, and usage tracking data. Each time a new service is built, service programmers need to implement all of these functionalities, which results in a long and tedious development process.

Application middleware [1] simplifies and eases the development of services. MidArc is an application middleware that implements basic functionalities common to many services, and enables access to these functionalities through interfaces exposed on the public network. Services use basic functionalities through defined interfaces and do not implement them by themselves. Basic functionalities implemented by the MidArc platform can potentially be used by thousands of services, and possibly millions of users. Hence, they have to be designed with scalability and robustness in mind.

MidArc storage system stores data required in operation of basic functionalities. Since implemented basic functionalities have to be fast, scalable, and reliable, the same characteristics are expected of the storage

system. In this paper, we propose three designs of the storage system, and compare their performances in environments with different number of services and users.

The rest of the paper is organized as follows. Section 2 briefly elaborates database technologies considered in design of three storage systems. Section 3 gives an overview of MidArc application middleware, while section 4 describes three MidArc storage system solutions. Performance measurement results are analyzed in Section 5. Final comments and conclusion are given in Section 6.

2. BACKGROUND

In design of the MidArc, two database technologies were considered as a basis for the storage system: directories and relational databases. In this section, each of these technologies is briefly described and compared to the other.

The directory service, or simply directory, is a collection of open systems that cooperate in holding a logical database of information about a set of real world objects [2]. Database is optimized for read-mostly operations and based on a hierarchical database model. Hierarchical database model organizes data in a hierarchical structure called tree. Nodes in the tree represent data, while vertices represent relations between data. Two nodes connected by a vertex are said to be in a parent-child relationship. In a hierarchical database model, each node can have multiple children, but only one parent. Relationship between nodes reflects relation between objects represented by nodes in the real world. Hierarchical database model is efficient for storing hierarchically organized data. Storing non-hierarchically organized data in hierarchically organized database is hard, and often leads to data redundancy. Hierarchical database model is efficient for storing read-mostly data. Read operations are performed fast, since all the information about the entity can be found in entity's node, or its subtree. Write and modify operations are more time consuming, because they often cause significant changes in tree structure. Directories are defined in ITU-T X.500 recommendation [2]. Client access to directory is usually performed using Lightweight Directory Access Protocol (LDAP) [3, 4, 5]. The advantages of using directory as a base for storage system are fast read operations, scalability, and easy physical distribution of data to multiple computers. The drawbacks are data redundancy and extremely slow write operations that render directory practically unusable in situations where data modifications are often.

Relational databases enable storing data in a way that is easy to understand and use by end users. In relational databases logical and physical data organization is defined separately. Data is logically organized into tables, each containing data of the same type. Each row in the table represents an entry, while columns represent

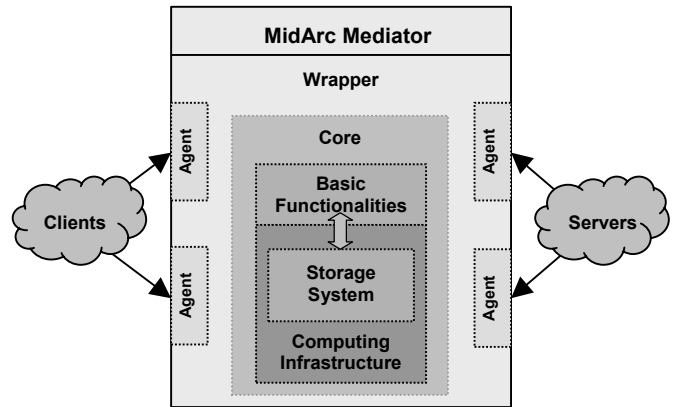


Figure 1: Architecture of MidArc Mediator

properties of the entry. Relation between data is represented logically, through equal property values in different tables. All operations on data in database can be made through universal Structured Query Language (SQL). Single query usually retrieves data from multiple tables, thus making relational database model less efficient when performing read operations than hierarchical model. Most of the databases available today are based on relational model, because of its ease of use. The advantages of relational database are equally fast read and write operations as well as low data redundancy. The drawbacks are slower performance for read operations than directory and complex physical distribution of data.

3. MIDARC

The development of new Internet services is a complex, expensive and error prone process. The complexity of the service development is a result of many functionalities that service provider has to implement. Almost all services share a set of basic functionalities that are implemented in each service. In order to facilitate service development, MidArc mediator is created. MidArc mediator is an application level middleware that implements basic functionalities and enables service providers to use them through well-defined interfaces.

The purpose of the MidArc mediator is to provide the support for service development and the run-time support for service execution. As shown in the Figure 1, the mediator consists of the core and the wrapper. The core contains implementations of basic functionalities common to many services and the computing infrastructure required for execution of these functionalities. Storage system is a part of the computing infrastructure. It stores data required in operation of basic functionalities. The computing infrastructure enables mediator's scalability.

The wrapper provides the access to basic functionalities for users and services. It also enables the cooperation of technologically, technically and organizationally disparate systems. The wrapper consists of the container application and the agents. The container

application provides the run-time support for agents, as well as their communication, synchronization and cooperation mechanisms. Agents are independent software components that are plugged into the wrapper and can perform various tasks. Some agents implement interfaces through which users and services access basic functionalities in the mediator core. Others work as proxies that interconnect clients and servers. Agents can also perform tasks like protocol transcoding, communication ciphering and unification of diverse services into logically single system.

In the current revision of the MidArc mediator six agents are implemented. Five agents implement Web Service [6] interfaces for accessing basic functionalities, and one works as a HTTP proxy that interconnects Web servers and clients.

4. MIDARC STORAGE SYSTEM

MidArc storage system is a part of the core of the MidArc mediator. Storage system stores various data about users, services, service usage and state of the mediator. Basic functionalities and other subsystems of the mediator use storage system to store and retrieve data required during their operation.

The access to the storage system is made through strictly defined storage interface (API). Storage interface contains functions for storing, retrieving, modifying and deleting data in the storage system. Since all access to the storage system is made through defined interface, it is easy to exchange various solutions of the storage system without compromising correct execution of the mediator.

The conceptual architecture of all three storage system solutions is equal and presented in Figure 2. The architecture consists of two layers: a software library and a database. The database stores mediator's data, while software library implements storage interface and translates interface calls into database calls. Storage system solutions described in the following sections differ in database technologies used in database layer and implementation of software library.

MidArc Data Types

Temporal properties of MidArc data have an important role in choosing database technology suitable for storing data. MidArc data is classified into three categories based on data's temporal properties: persistent, non-persistent, and log-type data.

Persistent data type includes infrequently changed data. After initial entry, persistent data is modified occasionally, while it is accessed frequently. Typical example of persistent data type is user data entered into data storage during user registration and hardly ever changed.

Non-persistent data type includes the data that is changed frequently, such as session data. Each time a user authenticates to the MidArc, a new session is established and session data is saved into database. When

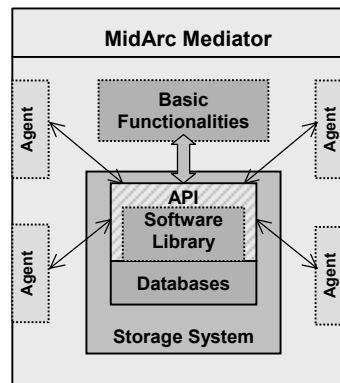


Figure 2: Conceptual architecture of MidArc Storage System

the user logs off, the session ends and the session data is removed from the database.

Usage records are of log-type. Log-type data includes data that is stored once and never modified. Every time a user calls a service, usage record is emitted into database. Unlike persistent data type, log-type data is read rarely, usually on a monthly basis.

MidArc Data

MidArc data is divided into six sets: user, service, group, authorization, session, and usage records. User data contains user information, such as user name, password, contact information and others. User data is stored so other users and services can locate and contact the user. User data is protected from misuse. Only authorized users and services can access user data. User data is of persistent data type.

When registering a service, basic information about the service is stored. Basic information includes: service name, password, network location, access mechanism and service description. Service information is required so users and other services can locate and use the service properly. Service data is of persistent data type.

Groups are used to ease the administration of user and service rights. A group can contain any user, service, or another group. Group information is used by access control basic functionality to check authorization rights. Although group data changes more frequently than user or service data, it is still read much more often than modified, thus it is categorized as persistent data type.

Authorization data defines rights to use services and rights to access MidArc data storage. Users can use only services they are authorized to use, and access only the data they are authorized to access. Since authorization data changes infrequently, it is classified as persistent data type.

Session data is used to establish and maintain sessions between services, users and the MidArc. Users and services provide the session key as a proof of their identity, instead of authenticating themselves each time they use the system. Session data is changed frequently and is therefore categorized as non-persistent data type.

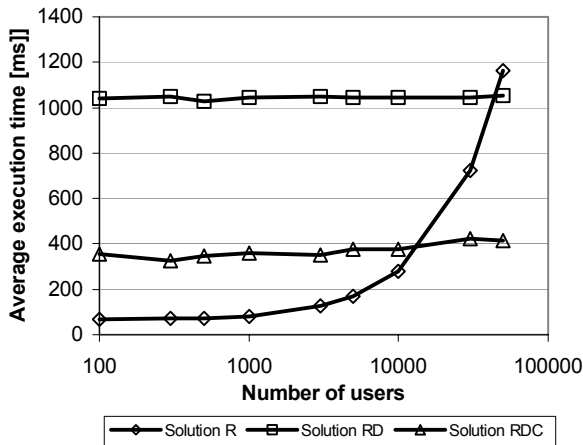


Figure 3: Average execution time of six critical storage system functions as a function of number of registered users

Usage record is saved every time a user uses service, or service uses another service. Collected usage records can later be used for billing. Usage records are log-type data.

Solution R – Relational database

The first storage system solution *R* uses relational database to store all MidArc data [7]. In the *R* solution, database layer is made out of a single relational database that stores all six sets of MidArc’s data. Software library of the *R* solution transforms storage interface calls into SQL commands and forwards the commands to the relational database. It also, receives the responses from the database and forwards them to the caller of the interface function.

Solution RD – Relational database & Directory

The database layer of the second storage system solution *RD* uses both the relational database and the directory. Part of the MidArc’s data sets are stored to the directory [8] and the rest to the relational database. Data’s temporal properties determine which data sets are put into the relational database and which to the directory. Since directory offers performance advantage when used for read-mostly data, all data sets of persistent type are stored into it. Since non-persistent and log-type data are often written, data sets of these types are stored into relational database. Again, software library is built that implements functions defined by the storage interface. Software library of the *RD* solution transforms storage interface calls into calls to the directory (using LDAP) or calls to the relation database (using SQL).

Solution RDC – Relational database, Directory & Cache

In order to improve storage access time, the third solution is developed. The third solution *RDC* is equivalent to the second one, but uses caching and buffering techniques to avoid frequent access to the



Figure 4: Average execution time of six critical storage system functions as a function of number of registered services

relational database and directory. Caches are used when reading MidArc’s data sets and buffer is used while storing usage records into the database. Caches and buffers are implemented in the software library of the *RDC* solution.

5. PERFORMANCE MEASUREMENT

In order to assess the scalability of the designed storage systems, measurements are performed. Since all solutions have software libraries that implement common interface, same data can be accessed with the same function in all solutions. Testing all functions defined by the storage interface is long and tedious, because interface defines more than a hundred different functions. Therefore, six most frequently used functions are selected for testing. These functions are used during each service call handled by the MidArc, thus have significant impact on overall mediator performance.

Tests measure average execution times of ten thousand calls of six frequently used functions. Measurements are made on mediator system with three computers: one runs the directory, the other relational database, and the third software libraries of different storage system solutions. All computers are based on PIII 800 MHz processors, with 128 MB RAM and connected with 100 Mbit/s Ethernet network. Measurement results are shown in Figures 3 and 4.

In the first test, the effect of the number of registered users on the average execution time of the different storage solutions is measured. As Figure 3 shows, average execution times of *R* storage system solution show exponential growth as the number of registered users grows. On the other hand, *RD* storage system solution’s average execution times remain nearly the same from a hundred to a hundred thousand registered users. However, *R* solution shows shorter response times for small number of registered users. This is a consequence of the database technologies used in

database layer of these two solutions. While *R* solution uses relational database, *DR* solution stores most of the data into the directory. Directory is optimized for read-mostly operations and design for environments with large number of users. Since out of six critical functions, five are reading data from the storage, this kind of optimization greatly improves average execution time. Caches and buffers added in the *RDC* solution reduce the average response time, but retain the scalability of the *RD* solution. Measurement results also show that there is no need to use the directory if expected number of registered users does not exceed certain number (in shown case 20000 users).

In the second test, the influence of the number of registered services on the average execution time of the different storage system solutions is measured. As Figure 4 shows the number of registered services has little impact on system performance for given number of services, no matter what storage solution used. While number of users of the MidArc system is expected to be measured in millions, the number of services offered through MidArc is measured in thousands. This difference in scale results in practically constant average execution time for all storage system solutions. However, the *RDC* solutions still has the shortest average access time.

6. CONCLUSION

Application middleware system MidArc implements basic functionalities commonly used by many Internet services. In addition, MidArc enables services to use these functionalities through precisely defined interfaces. During operation, basic functionalities need to manage various data about users, services, and state of the MidArc system. Data management in the MidArc is done by the storage system. Since basic functionalities implemented by the MidArc are potentially used by thousands of services, and even millions of users, the storage system that they use has to be scalable and robust.

Data stored in the storage system are managed exclusively with the functions defined by the storage interface. Since all data management is done through defined interface, there can be multiple storage system solutions implementing the same interface that can be used interchangeably.

In this paper, three storage system solutions are presented. Solution *R* uses single relational database to store and manage all MidArc's data. Solution *RD* uses directory and relational database. All MidArc data that is of persistent (read-mostly) type is stored into the directory, since directory offers performance advantage when used for data of this type. All other (non-persistent and log-like) MidArc's data is stored into the relational database. Solution *RDC* is an upgrade of the *RD* solution that uses caches and buffers to improve data access time.

Measurement results show that the simplest solution *R* is suitable for small organizations with small number of

registered users. Larger organizations should consider using *RD* or *RDC* storage system solution, because their performance is not affected by number of users registered in the system. Since all storage systems implement the same storage interface, it is possible for an organization to start MidArc with *R* storage system and to upgrade to more scalable storage system solution as the number of registered users and services grow.

Acknowledgement. This work was sponsored by Ericsson Nikola Tesla, Croatia. Their support is gratefully acknowledged. We also thank Goran Radic for his support in design of relational database storage system solution.

7. REFERENCES

- [1] Lerner, M., Vanecek, G., Vidovic, N. and Vrsalovic, D., **Middleware Networks: Concept, Design, and Deployment of Internet Infrastructure**, Kluwer Academic Publishers, 2000.
- [2] ITU-T Recommendation X.500, Information technology – Open Systems Interconnection – **The Directory: Overview of concepts, models and services**, 2001.
- [3] D. Thompson, **Understanding LDAP**, White paper, Windows 2000 Server Documentation, 2000.
- [4] T. Howes, **LDAP: Use as Directed, Netscape**, 1999. <http://www.networkmagazine.com/article/DCM20000502S0039/1>, September 2001.
- [5] T. Howes, G. Good, M. Smith, **Understanding and Deploying LDAP Directory Services**, Macmillan Technical Publishing, 1999.
- [6] Microsoft Corporation, **Web Services Specifications**, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsspecover.asp>, August 2002
- [7] G. Radić, **Relational Database of a Middleware System**, diploma thesis in Croatian, original title **Relacijska Baza Podataka Middleware Sustava**, School of electrical engineering and computing, Zagreb, 2002.
- [8] F. Plavec: **Directory of Middleware System**, diploma thesis in Croatian, original title **Direktorij Middleware Sustava**, School of electrical engineering and computing, Zagreb, 2002.