Media Informatika, Vol. 2, No. 2, Desember 2004, 23-31
ISSN: 0854-4743

COMPARISON ON EFFICIENCY AND SPEED OF 2-TIER AND 3-TIER OLTP SYSTEMS

Teduh Dirgahayu, Muhammad Indra Utama
Department of Informatics, Faculty of Industrial Engineering,
Islamic University of Indonesia
Jl. Kaliurang Km. 14 Yogyakarta 55501
Telp. (0274) 895287 ext. 122, Faks. (0274) 895007ext. 148
E-mail: teduh.dirgahayu@fti.uii.ac.id, mhs98523177@yahoo.com

ABSTRACT

Online Transaction Processing (OLTP) systems are characterized by a large number of users accessing online data simultaneously. Available architectures for distributed OLTP systems are 2-tier and 3-tier client/server architecture. Two major factors that have to be considered in deciding architecture of an OLTP system are resource efficiency and speed. This research aimed to compare those two different architectures on efficiency and speed. As a test bed, we developed 2-tier and 3-tier web-based application for online banking using Microsoft COM+ and ASP. We then tested them using Microsoft Web Application Stress Tool. Our components were designed into two layers: Business Access Layer (BAL) and Data Access Layer (DAL). The results show us that there was a trade-off on resource efficiency and system's speed. The 2-tier OLTP system gave us better speed performance but lower resource efficiency. On the other hand, the 3-tier system offered more efficient resource utilization i.e. it saved 62% - 64% of the connection needed for 2-tier system, but its speed reduced at about 24% than the speed of 2-tier system. Further research is needed to examine more number of concurrent users in longer test duration so as to get realistic behaviors of large OLTP systems.

Keywords: *2-tier and 3-tier architecture, TP monitor, efficiency and speed*

1. INTRODUCTION

Online Transaction Processing (OLTP) systems are characterized by a large number of users accessing online data simultaneously. Data accessed by OLTP system resides in one or more databases. Currently the number of users accessing OLTP system is on the increase up to thousands or billions.

Most of nowadays OLTP systems use 2-tier client/server architecture. The idea of 2-tier client/server architecture is to split processing load in two: front-end application and back-end data storage. The majority of business logic runs on the front-end application, which typically sends Structured Query Language (SQL) requests to the back-end data storage. An alternative approach to improve this architecture is to use stored procedures in order to off-load some of business logic processing to back-end side that is a database server. This approach is called 2.5-tier architecture (Edwards and DeVoe, 1997).

Another available architecture for OLTP systems is 3-tier client/server architecture. This architecture comes in three parts: user interface on client terminal, business rules run on application server, and data storage. One of the architecture's advantages is that it allows the application server to manage all client connections to the database server instead of letting each client make its own connection, since too many connections will waste resources on the database server. This concept is called connection pooling, which means that client requests are put into a pool or queue to wait for an available connection. As a connection is available, it can be used for the request in queue. The connection pooling software is known as Transaction Processing (TP) Monitor.

Two major factors that have to be considered in deciding whether we shall build an OLTP system as 2-tier or 3-tier system are resource efficiency and speed. Our research aimed to compare those two different architectures on efficiency and speed.

2. TP MONITOR

TP Monitor is a middleware that focuses on coordination of processes and highly reliable transactions. TP Monitor is not used for program-to-program communication. It provides an environment for transaction applications that are accessing databases. It is used on mission-critical applications that require rapid response and tight controls over security and integrity of database. TP Monitor shall be considered when transactions need to be coordinated and synchronized over multiple databases. TP Monitor provides robust run-time environment which is able to support large-scale OLTP system that require immediate responses.

TP Monitor can be thought as an operating system for transaction processing applications. TP Monitor does three following functions (Edwards and DeVoe, 1997; Orfali et al., 1996).

a. Process management, which includes starting server processes, funneling work to them, monitoring their execution, and balancing their workloads.
b. Transaction management, which means that TP Monitor guarantees the Atomicity, Consistency, Isolation, and Durability (ACID) properties to all applications that run under its protection.
c. Client/server communication management, which allows client and services to invoke application components in a variety of ways including request, response, conversations, queuing, publish and subscribe, and broadcast.

A transaction is a unit of consistent and reliable computation. A database state consists of a set of values of data items in the database. The state is consistent if the database obeys all integrity constraints or if the database was consistent before the transaction was executed, regardless of the facts that are:

d. Transaction was executed concurrently with other transaction.
e. Failures may have occurred during execution of transactions.

The job of a TP Monitor is to guarantee the ACID properties while maintaining high transaction throughput. So that it must manage the execution, distribution, and synchronization of transaction workload.

As component-based middleware widely implemented, transaction coordination becomes even more important. Clients mix and match several components that may make updates to the database. Consequently their updates need to be coordinated. TP Monitor plays this role by providing ACID properties to the application (Browne, 2000). TP monitor ensures that all updates associated with aborted transaction are removed or rollback. It can even perform when components are on different servers and are updating different databases from different vendors. When resource managers are across networks, TP Monitor synchronizes all the transaction's updates using two-phase commit protocol.

3. MICROSOFT COM+

Microsoft COM+ (Component Object Model *plus*) is a middleware for distributed computing that enabling applications to be full scalable, flexible, and maintainable. It allows applications to use a TP monitor in the middle tier of a distributed system (Browne, 2000). Figure 1 illustrates COM+ providing full scalable N-tier systems.

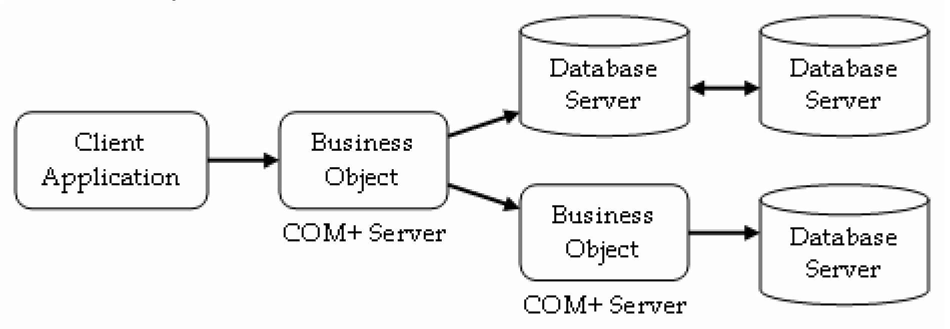

Figure 1. COM+ provides N-tier distributed system

COM+ handles not only distributed transaction processing, but also application messaging and the full complement of services necessary to build and run enterprise-wide applications. It enables developers to create applications that span multiple hardware platforms, databases, and operating systems with full freedom to mix-and-match those platforms to best fit the application environment.

COM+ is targeted as a platform of OLTP systems. Business objects access data in various data sources such as Database Management System (DBMS) across the network. A TP Monitor built into COM+ ease developers in defining transaction across multiple data sources. COM+ uses declarative transactions, which hide proprietary transaction APIs from its developers.

4. METHODS

In order to conduct the research, we built 2-tier and 3-tier web-based applications for online banking using Microsoft COM+ and ASP (Active Server Pages) scripting technology. We then tested them using Microsoft Web

Application Stress Tool (WAST) to simulate a large number of concurrent users. In order to provide similar load for and from clients, we built same user interface for both applications. During the test, we collected the numbers of connections made to the database server and the numbers of hits provided by the application server. We exploited our applications for 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000 concurrent users between clients and application server. Each of them ran for 5 and 10 minutes. Data collected were analyzed statistically by means of linear regression and correlation analysis.

To carry out our research, we used several hardware and software. The specifications are shown in Table 1.

Table 1. Hardware and software specifications

Machine	*Hardware*	*Software*	*Number*
Database Server	Celeron 800 MHz RAM 128 MB	Windows 2000 Advanced Server SQL Server 2000	1
Web Server	Celeron 800 MHz RAM 128 MB	Windows 2000 Advanced Server Internet Information Server (IIS)	1
Client	Celeron 800 MHz RAM 128 MB	Windows 2000 Advanced Server Web Application Stress Tool (WAST)	7
Monitor	Celeron 800 MHz RAM 128 MB	Windows 2000 Advanced Server Performance (a Windows 2000 feature)	1

5. COM+ COMPONENTS DESIGN

We designed our COM+ components by considering the following:

a. All components were made in dual interface to support late binding and vTable binding technique.
b. The COM+ object were divided into two layers: Business Access Layer (BAL) and Data Access Layer (DAL).

The design of COM+ components are shown in Figure 2.

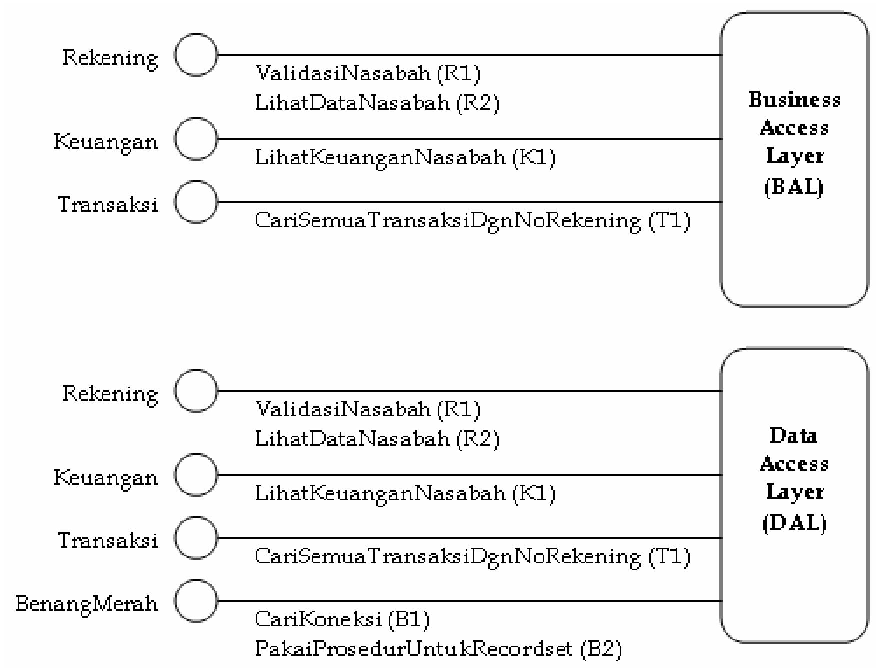

Figure 2. Design of components as BAL and DAL

Business Access Layer (BAL)

BAL component consisted of three interfaces: **Rekening**, **Keuangan**, and **Transaksi**. All business methods were then built into these interfaces. These methods were available to be invoked by client applications.

Data Access Layer (DAL)

DAL component consisted of four interfaces: **Rekening**, **Keuangan**, **Transaksi**, and **BenangMerah**. The first three interfaces directly supported the interfaces in Business Access Layer respectively. We used same names for BAL and DAL interfaces and methods only to indicate the relationships between them. Methods in the last interface, **BenangMerah**, were used by other interfaces in DAL components in making database connections.

The relationships between methods of interfaces are shown in Figure 3.

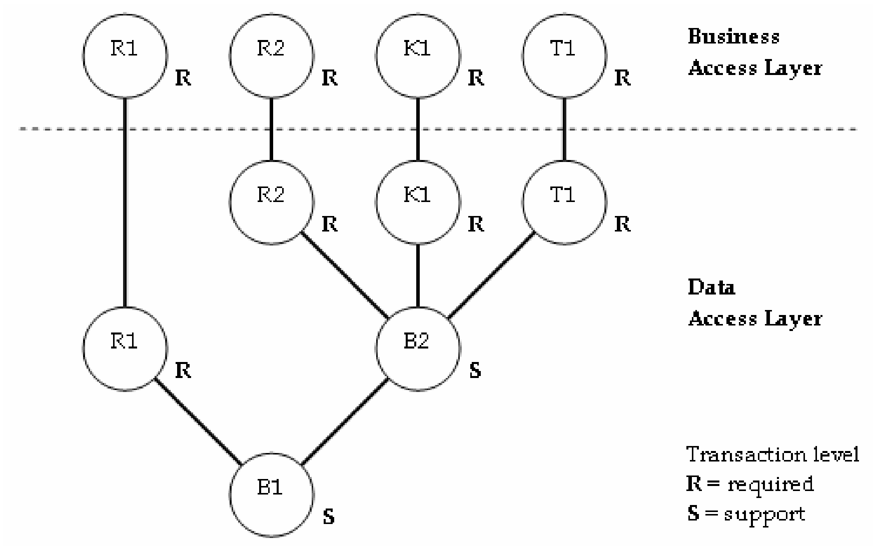

Figure 3. Relationship between methods of interfaces

6. RESULTS AND DISCUSSION

The numbers of connections obtained from the test are shown in Table 2. It shows us that, in average, 3-tier OLTP system could save number of connections about 62% - 64% than 2-tier systems. The 3-tier system only needed 10.47 to 14.25 connections, while the 2-tier system consumed 27.52 to 34.39 connections. Hence implementing distributed application as 3-tier system was more resources efficient than the 2-tier one.

Table 2. Numbers of connections

Concurrent Users	*5 minutes*			*10 minutes*		
	2-tier	*3-tier*	*Saving*	*2-tier*	*3-tier*	*Saving*
100	27.52	13.24	52%	30.51	12.96	58%
200	32.00	12.17	62%	32.15	13.29	59%
300	32.13	12.05	62%	35.26	12.86	64%
400	33.59	14.25	58%	35.07	10.96	69%
500	31.57	13.41	58%	34.39	11.80	66%
600	33.38	11.40	66%	33.63	12.96	61%
700	31.72	10.47	67%	32.86	11.68	64%
800	31.82	11.81	63%	34.07	11.16	67%
900	32.43	10.57	67%	32.13	11.14	65%
1000	31.80	11.52	64%	31.87	11.39	64%
		Average	62%		Average	64%

The results of linear regression and correlation analysis of the data are shown in Table 3. The table denotes that the numbers of connections had some correlations with the number of concurrent users connected, although it was not tight correlation between those two variables for 2-tier systems (indicated by r = 0.150 and r = 0.420). However the influence of the number of concurrent users (NCU) was considered as very small portion of the number of connections (NC).

Table 3. Analysis results from the numbers of connections

System	*5 minutes*		*10 minutes*	
	Linear regression	*R*	*Linear regression*	*R*
2-tier	NC = 30.589 + 0.002 NCU	0.402	NC = 33.236 – 0.0001 NCU	0.150
3-tier	NC = 13.463 – 0.002 NCU	0.617	NC = 13.165 – 0.002 NCU	0.701

Notes:NC: number of connections; NCU:number of concurrent users; R:correlation coefficients

Beside the numbers of connections, also we collected the numbers of page hits in order to measure their speeds. The numbers of page hits obtained from the test are shown in Table 4. It shows us that 3-tier system lose about 11% to 35% page hits compared to 2-tier systems. Since we intended to analyze the systems' speeds, we converted Table 4 into Table 5, which shows us the comparison of average speeds in unit of page hit per second.

Table 4. Numbers of page hits

Concurrent users	*5 minutes*			*10 minutes*		
	2-tier	*3-tier*	*Loss*	*2-tier*	*3-tier*	*Loss*
100	60,795	39,899	34%	121,254	79,577	34%
200	60,758	39,740	35%	121,256	79,176	35%
300	60,889	44,239	27%	115,449	82,776	28%
400	60,564	44,169	27%	114,915	79,811	31%
500	59,993	46,902	22%	118,017	91,582	22%
600	59,402	50,477	15%	118,857	89,005	25%
700	61,322	46,321	24%	119,211	91,015	24%
800	58,011	49,252	15%	103,757	89,456	14%
900	58,341	51,994	11%	111,489	84,198	24%
1000	58,685	51,994	11%	123,008	94,760	23%

Table 5. Average speeds in page hits per second

Concurrent users	*2-tier*	*3-tier*	*Slowdown*
100	202.37	132.81	34%
200	202.31	132.21	35%
300	197.69	142.71	28%
400	196.70	140.12	29%
500	198.34	154.49	22%
600	198.05	158.30	20%
700	201.55	153.05	24%
800	183.15	156.63	14%
900	190.14	156.82	18%
1000	200.32	165.62	17%
Average	197.06	149.28	24%

From Table 5 we can see that the processing speed of 3-tier system was lower about 24% in average than the 2-tier system. It only provided maximum 165.62 page hits per second whereas the 2-tier system gave us 202.37 page hits per second. Therefore the performance of 2-tier OLTP systems was faster than the 3-tiers system's performance. However there was an interesting observable fact that the speed of 3-tier system tended to rise from 132.81 to 165.62 page hits per second whilst the 2-tier about stayed constant.

7. CONCLUDING REMARKS

There was a trade-off on resource efficiency and system's performance between 2-tier and 3-tier OLTP systems. The 2-tier OLTP system gave us better speed performance by sacrificing system's resource efficiency. On the other hand, the 3-tier system offered more efficient resource utilization, but it reduced system's speed. By recognizing these facts, 2-tier architecture is a good choice for small OLTP system and 3-tier architecture is recommended for medium to large OLTP system. Using 2-tier architecture for medium or large OLTP systems have a critical risk i.e. the system will die since there will be no more resource available to meet the connections' needs.

Putting requests on a pool to queue them for available connections in 3-tier architecture makes the system's speed slow down. From Figure 3, it is shown that potential bottlenecks occurred in **B1** and **B2** methods of **BenangMerah** interface on Data Access Layer. As connection request increases, COM+ server creates more connections to database in order to shorten the queue in pools. This mechanism is transparent from developer's view.

Further research is needed to examine more number of concurrent users in longer test duration so as to get realistic behaviors of large OLTP systems.

REFERENCES

Armstrong, T. (1999). *COM + MTS = COM+: The Next Step in Microsoft Component Strategy*. Retrieved from http://archive.devx.com/premier/mgznarch/vcdj/1999/ febmag99/commts1.asp

Browne, C. B. (2000). *Transaction Processing Monitors.* Retrieved from http:// cbbrowne. com/info/tpmonitor.html

Edwards, J., and DeVoe, D. (1997). *3-Tier Client/Server at Work*. New York: John Wiley & Sons.

Orfali, R., Harkey, D., and Edwards, J. (1996). *The Essential Distributed Objects Survival Guide*. New York: John Wiley & Sons.