Journal of Korea Multimedia Society Vol. 18, No. 11, November 2015(pp. 1367-1374)
http://dx.doi.org/10.9717/kmms.2015.18.11.1367

A Risk-based System Analysis Model for Improving the Performance of Financial Solutions

Jong Yun Lee[†], Jong Soo Kim[††], Tai Suk Kim[†††]

ABSTRACT

In this paper, we propose a model which can prioritize the performance improvement work by analyzing the major risks and their influence, which can cause performance degradation in the system and show an example of a performance improvement using this model. In presentation-tier, as a result of log data analysis before and after the performance improvement of key processes which handle financial transactions, this model brought the CPU utilization and memory enhancement in the performance improvement work of the financial system which was carried out by applying the proposed model. It has been confirmed that the entire end-user can be accommodated. In the web-tier, the available memory increased by 200MB and we were able to improve the server restart(Recycling) that was sustained in the existing system. In the business logic-tier, we have been able to see better figures after performance improvements through the graph which analyzes the log collected with the key performance counters such as CPU%, Batch Requests/sec. In the data-tier, it has been confirmed that CPU usage and standby operation were reduced and the throughput was found to increase.

Key words: Risk-based, Multi-tier, Business Logic, Performance Counter, Windows

1. INTRODUCTION

If the problem causing the deterioration in the multi-tiered system of complex structures occurs, it is difficult to find the evaluation model that can diagnose and improve it. In many cases, some systems with assessment models are specialized in Unix or Windows-based specific applications. There are few systematically documented methods for maintaining computer systems with complex structures without performance degradations and enhancing their performance when necessary. Most methods in use take advantage of the opinions of experts in the field, previous experience, and computer knowledge.

Thus, in the event of a problem, it is often difficult to resolve the problem by a general approach. Key studies relating to performance enhancements for computer systems include studies to reduce time by automating load testing [2], studies predicting performance through estimation techniques using statistical regression analysis for load tests [1], and studies on load testing methods for large scale websites accomodating numerous requirements [3].

In this paper, we propose a model that can efficiently carry out the improvement work by analyzing the cause of performance degradation which occurs during the operation of the large capacity financial system consisting of a multi-tiered sys-

※ Corresponding Author : Tai Suk Kim, Address: Dept. of Computer Software Engineering, Dong-eui University 176 Eomgwangno Busan_jin_gu, Busan 614-714, Korea, TEL : +82-51-890-1707, FAX : +82-51-890-1724, E-mail : tskim@deu.ac.kr
Receipt date : June 30, 2015, Revision date : Aug. 26, 2015
Approval date : Sep. 11, 2015

[†] Dept. of Computer Software Engineering, Dong-Eui University (E-mail : joylee@microsoft.com)
[††] Dept. of Lift Engineering, Korea Lift College (E-mail : seatree@klc.ac.kr)
[†††] Dept. of Computer Software Engineering, Dong-Eui University
※ This work was supported by Dong-eui University Grant(2015AA040).

tem on the Windows server base. Among the financial systems, BPS(Branch Process System) is a critical system, which should handle the customer request quickly and reliably.

In order to improve the performance of the BPS, we created a system model to select a performance improvement priority by analyzing risks and their impact which can occur from such as the business process, the major hardware, and the software resources. After that, we have completed the work improvement by applying it to the corresponding system. After the work improvement applying the proposed model, it is shown that the performance for each tier that makes up the system in the performance evaluation has improved.

2. RELATED WORKS

WMI(Windows Management Instrumentation) by Microsoft is created by implementing WBEM (Web-Based Enterprise Management Initiative) which is industry's initiative to create a standard to access and share management information in an enterprise network. WMI provides an integrated support for CIM(Common Information Model) which is a data model describing the object existing in the managed environment[4-5].

PerfMon for collecting performance analysis logs in the Windows systems can be monitored locally or remotely and can dynamically generate the menu and user interface for setting the time and circumstances. Tools to diagnose the system for each product have also been frequently deployed for the performance analysis of the main platform software which can hardly be analyzed only by performance logs in the system. These tools help you to quickly find performance bottlenecks and other problems. Typically, there are SQL Profiler and PSSDIag of SQL Server and CLR Profiler and Allocation Profiler of .NET and so on[6-9]. Even if performance tools can collect various logs, there are logs that are collected preferentially for the efficiency of the work.

3. RISK-BASED SYSTEM ANALYSIS MODEL

In this chapter, we will discuss about the risk-based system analysis model to efficiently process performance improvements of the system built in "A" financial company and show an example of implementing the system performance improvement by applying the analysis model.

3.1 Problem Domain

The structure of the Windows server-based financial solutions is shown in Fig. 1.

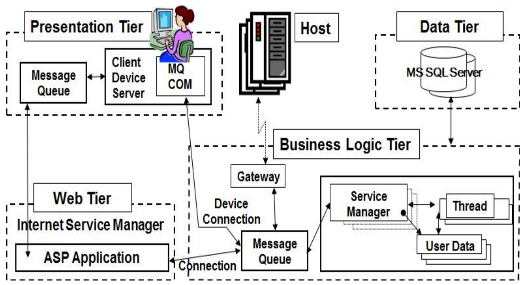


Fig. 1. The architecture of financial solutions.

For Web-tier, a web server based on a Windows server is installed while SQL Server was built as a distributed database system for data-tier. The realization of BP server that is responsible for the implementation of business logic uses the MSMQ (Microsoft Message Queuing) technology to efficiently process messages that are sent to and received by each tier and defines the business rules of the central host(HOST) and each tier. The target systems for analysis were web servers connected to by thousands of clients, and dozens of BP servers handling its requests. Testing was performed for the web server, BP servers, and database server, and the key issues returned from the analysis were as follow.

1) Whereas the threshold for performance counter available bytes in memory is 20% or higher,

and 20 or less pages/sec, the results from testing showed 15% and 60 pages/sec. That is, a memory shortage phenomenon appeared.

2) Whereas in memory, 50% or lower is the threshold value for % processor time, the result was 90%, and for processor QLength, where the threshold value was 5 or lower, the result was 30. That is, a CPU bottleneck appeared.

3.2 Performance Improvement Model

Financial solution is characterized in that the requested processing must be performed unlike other computer system. In this respect, the determination of the major risk factors that interfere with the request processing is important. The risk-based system analysis model which is proposed in this paper in order to improve the performance of a specialized computer system such as the financial sector is shown in Fig. 2.

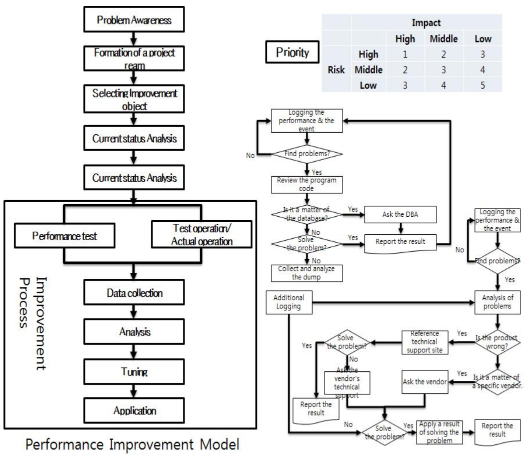


Fig. 2. Risk-based system analysis model.

The general model can be applied as a guide for the performance improvement of the system. However, since most of the large computer system structures in financial institutions in recent years have consisted of multi-tier, it is necessary to have a performance improvement plan which fully considers the section of bottlenecks in the main tier. In view of these characteristics, we divide them into several processes such as the performance testing, data collection, analysis, tuning, improvement processes for the efficient operation of the application stage, process improvement for the tuning of specific product issues, business logic improvement process, and user application improvement process. In addition, using items such as user feedback, the application log and performance log, you can identify the sections which require performance improvements.

Prioritizing the work is one of the big issues. Since all works look as if they were all the high priority tasks, it is not easy to determine the importance of the work. Table 1 shows a one-dimensional priority according to the risk.

Table 1. The risk of one-dimensional priorities

Risks	Description
1	State of not being able to proceed with the work at all.
2	Tasks are not discontinued but cause some problems from time to time.
3	No problem to proceed with the work as a fallback.

The degree of influence on the risk is determined based on the potential that can cause problems and the influence caused by problems and it was expressed as a quantitative method from 1 to 5 to determine the intensity of the priority for risks which are specifically subject to improvement objects. The issues for a flexible schedule can be categorized by, for example, an impact analysis of the issue, the part of the system where the issue occurred, and the current status of the issue.

We will prioritize by using various items such as business professionals, software engineers, and interviews with hardware engineers, independent assessment, workshops, brainstorming, checklists and so on. As described above, once you one-dimensionally prioritize the various tasks, it is difficult to prioritize due to the presence of a lot of the same level of priority tasks. In determining the priority of the problem, the following methods can be used to classify risks. The priority based on the risk is selected in consideration of the influence of a business or system for the possibility of failure. Business section can be selected with the help of business professionals while the system section can prioritize through interviews of each system-specific engineer.

3.3 Applying the Proposed Model

In regard to the process where a number of end-user login at the same time for the start of the work in whole branches of banks running the conventional computer systems, we have obtained test results of the development department and get the user feedback as well as periodic monitoring results of the operation department. Using these data, we have identified several issues as follows: In some clients, the response time during the initial logon service stays very slow and finally stop responding. For a large number of clients, the response time to General Trading(director approval, deposit and withdrawal transactions, etc.) becomes very slow as well. To address the problem appearing at this stage, we set the performance index by identifying the factors affecting the performance of each tier and perform the analysis accordingly.

The severity and impact on risk in the web server sector represent the results from interviewing the technical support engineers at company M. The key items and priorities were selected based on these results. The priorities for the key items requiring improvement in the Windows server system, IIS web server, and ASP.NET comprising the web server.

The analysis targets selected based on the priorities of key items in the web server sector, as shown in Table 2, were server PAE options, ASP.NET Recycling, ASP.NET Hang, and Private Queue code improvement.

Table 2. Selection of analysis targets in the web server

Web Server	Main Item	Server PAE Opiton	ASP.NET Recycling/ Hang	Private Queue
Windows Server	CPU	▨	▨	▨
	Memory	▨	▨	▨
	Network Interface		▨	▨
	Process	▨	▨	▨
	System	▨	▨	▨
IIS Server	IIS		▨	
	Web Service		▨	
	Network Interface			▨
ASP	ASP		▨	
	Web Service		▨	
	.NET		▨	

It was analyzed that there is need to improve the phenomenon where, under excessive load, ASP.NET is unresponsive, and the web server restarts automatically, and the phenomenon where hang occurrs in ASP.NET, causing the status of the system to become inoperable.

Both Identifying problem resolution time according to the importance of the work and rescheduling of business processes for performing the whole job efficiently should be additionally considered.

As a result of performing the test by applying a risk-based system model, it is proven that the preferred improvement is required in the following items.

- Staff approval process analysis
- The system goes down as the trading transaction increases
- MSMQ(Microsoft Message Queuing) Code Optimization
- Degradation of some databases

As a result of analyzing the staff approval process which is one of important business, used in the presentation tier, issues that increase the network traffic due to large amounts of data and communication frequency that send and receive data with BP in order to handle the request of staffs and approval requestors as shown on the left picture in Fig. 3 were found in the conventional methods. To solve this problem, we minimized the amount of data and communication frequency between clients and servers by changing staff approval process into communication format as shown on the right picture in Fig. 3.

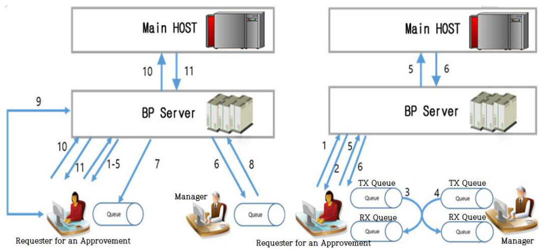


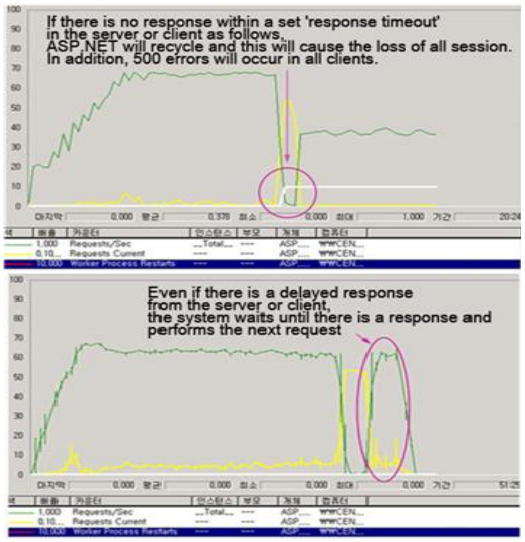
Fig. 3. Staff Approval Process before(left) and after (right).

For the improved staff approval process, we process the requested operation in the following order.

① Sending transactions from the terminal(http communication)
② If it's the approval at all times in the business logic-tier, zoom down the staff list
③ Data is sent to the staff and the operator(MQ communications)
④ Transmission of approval and rejection(MQ communications)
⑤ Transmission to professional deal and staff approval information HOST(via BP)
⑥ Reception from the host(via BP)

If ASP.NET is not promptly processing the response to the requested action in the web-tier during the occurrence of over-load, the issue is found that the system will determine that there is no response and restart recycling automatically. Since the cause of problem is that the response timeout is set to 3 minutes, it is analyzed that if you do not respond within the specified time, the system becomes unstable.

If there is no response to a requested operation within the time(3 minutes) in the left graph analyzing logs collected by Requests/Sec performance counter of Fig. 4, you can see the ASP.NET restarts.


If there is no response within a set 'response timeout' in the server or client as follows,
ASP.NET will recycle and this will cause the loss of all session.
In addition, 500 errors will occur in all clients.

Even if there is a delayed response from the server or client,
the system waits until there is a response and performs the next request

Fig. 4. ASP.NET response timeout setting before(top), after(bottom) the change

As measures to improve these problems, the recycling function is set to 'disable' so that ASP.NET does not use the recycling function even under the excessive load and the set value of 'reponse DeadlockInterval' which is the response timeout property was changed to infinity. After changing the value of the property, the test was performed. Test results show that even if there is a delayed response from the server or client, it can be seen that the system waits until there is a response and performs the next request as you can see in the figure on the right.

In business logic-tier, when communicating with MSMQ(Microsoft Message Queuing) that supports the communication with system which can be temporarily off-line and that supports the applications running on different time in a different type of communication network, the user should remove the generated cue to stop the waste of memory resources after using the MSMQ object that users created. However, we could not find information about deleting the queue in the current code. To solve this problem, we performed the code improvement as Table 3.

Table 3. a code to delete user-generated MSMQ

```
#include <Mq.h>
…

CWinApp theApp; //declare unique object

namespace declare;

…… omitted

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
   int nRetCode = 0;

   if (fail of MFC init) {
      Message print(“Fatal Error : fail of MFC init\n”);
      nRetCode = 1;
   }
   else  //sucess on the creation of the MFC object
   {
      HRESULT hr = 0L;
          QUEUEHANDLE hQueue; //Queue handler
      WCHAR queueName[] =
                     L“.\\PRIVATE$\\myQ”;
      WCHAR formatName[256];
      DWORD lenOfFormatName =
          sizeof(formatName)/sizeof(WCHAR);

      hr = MQPathNameToFormatName(
      queueName, formatName, &lenOfFormatName);
      if(MQ_OK==hr){
          hr=MQDeleteQueue(formatName);
          if(MQ_OK==hr)
              MsgBox("sucess : MQ delete\n");
          else MsgBox(“fail : MQ delete HR=%x\n”, hr);
      }else MsgBox(“fail : MQPathNameToFormatName
HR=%x\n”, hr);

      if(hr==CreateMSMQQueue())   //MQ create
          MsgBox(“sucess : CreateMQ HR=%x\n”, hr);
      else MsgBox(“fail : CreateMQ HR=%x\n”, hr);

      …… omitted

   }
…… omitted
}//main() method end.
```

In the DB-specific Lock Timeout analysis of the database-tier, Lock Timeout occurs most frequently in Tempdb and Lock Timeout in Tempdb occurred entirely for NULL object. In the analysis of application-specific Lock Timeout, we can see that the specific applications that use Tempdb a lot tend to generate a lot of Lock Timeout. This means that if users are driven excessively, it becomes a cause for a bottleneck. Thus, the improvement work was carried out to separate the Tempdb to a different physical disk.

4. PERFORMANCE EVALUATION

In Fig. 5, you can see the graph analyzed by the Available Memory(Available MByte) Counter for the evaluation before and after the performance improvement of the web server.

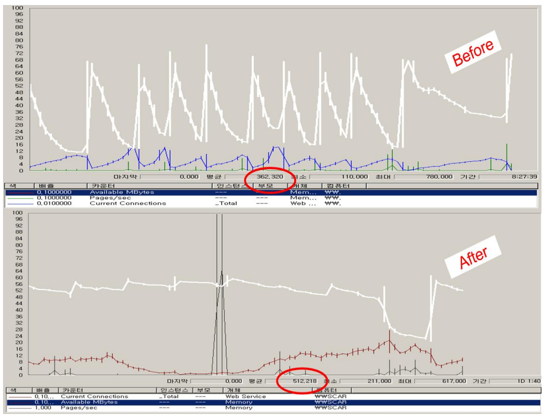


Fig. 5. Web-tier performance analysis.

After the improvement, it has been confirmed that Available Memory increased more than 200 MB. In servers, generally speaking, 200MB cannot be said to be a large value. Despite a reduction in the value of the % processor time performance counter in the server with a memory capacity of 4GB resulting from a resolution of the relevant issues, the 5% increase in the value for Available Memory can be said to indicate that CPU and memory usage of the server is relatively stable.

For a number of servers that perform the business logic, the analysis result on the average of these servers before and after improvements has confirmed that CPU usage was reduced and the processing speed per second increased. The resource status comparison table of key performance counters before and after the system improvement is summarized in Table 4.

Table 4. Business logic-tier resource status comparison

	CPU %	Batch Req/sec	Tx/Sec (SH)	DB Con	Recompile
Before	97.8	267.9	43.1	697.2	7.34
After	30.9	337	59.8	728.4	9

In regard to data-tier, as a result of analyzing logs before and after the performance improvement using Avg CPU Time(MS) and Batch requests/sec performance counter, we have found that the data improved like in Fig. 6.

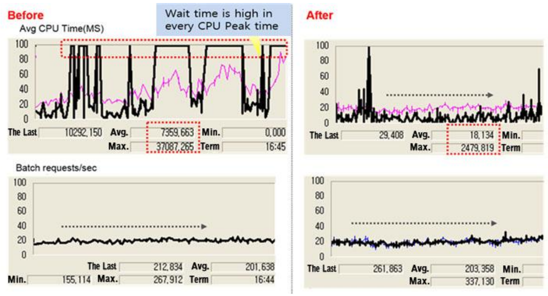


Fig. 6. Data-tier evaluation analysis.

In the graph at the upper left corner before the database improvement, you can see that, the average CPU usage time curve appears unstable. This is because the waiting time to use resources gets longer as higher CPU utilization is required due to the blocking phenomenon. The average CPU usage time curve after improvement which can be seen at the top right corner appears stable and the peak condition has temporarily occurred but been immediately recovered. Also, in the waiting time(Batch requests/sec) graph at the bottom right corner, it can be seen that the plain curve at an average of 20 degree is drawn. In particular, even if the query number(261,863) requested from the Peak Time is similar, the maximum number of requests(337,130) is shown to increase than before the improvement.

5. CONCLUSIONS

In this paper, we proposed a risk-based system analysis model to carry out the performance improvement of the system that was built on a multi-tier based on Windows Server. The improvement work was carried out by applying the proposed model to ‘A’ company's financial solutions.

With respect to major improvement work, the staff approval process on the client implementation was simplified. As trading transactions increased in the web-tier, we changed the properties to disable the recycling function in order to improve the down event and set a new response timeout property. We optimized the code to delete the MSMQ(Microsoft Message Queuing) object that users created for message communications that were generated in each tier in the business logic-tier. The TempDB that caused the greatest performance degradation in the data-tier was separated on separate physical disks.

After carrying out a performance improvement work by applying a risk-based analysis model, we compared the performance before and after improvements, using logs collected by using the key performance counters. And it has been confirmed there were overall performance improvements.

In future research, we need more in-depth system analysis model considering the rapidly changing architecture and a variety of multi-tier system. In addition, it is required to expand into a model that is applicable regardless of the base system.

REFERENCE

[1] T.H.D. Nguyen, B Adams, Z. Jiang, and A.E. Hassan, “Automatic Load Test Verification

Using Control Charts," *Proceedings of the 18th Asia-Pacific Software Engineering Conference* APSEC, pp. 282-289, 2008.

[2] S. Duttagupta and M Nambiar, "Performance Extrapolation using Load Testing Results," *International Journal of Simulation, Systems, Science and Technology*, pp. 66-74, 2008. ISSN: 1 67 473-804x online, 1473-8031 print.

[3] S.W. Lee, J.S. Kim, and T.S. Kim "Optimization of DB Server and Web Server to Enhance the Performance of ECM," *Journal of Korea Multimedia Society*, Vol. 16, No. 12, pp. 1446-1453, 2013.

[4] WMI Provider-Monitoring in .NET Distributed Application Design, http://msdn.microsoft.com/asp.net/using/understanding/perf/default.aspx?pull=/library/en-us/dnbda/html/monitordotnet.asp 2014.11.05

[5] M. Russinovich, D. Solomon, and A. Ionescu, *Windows Internals Vol. 1-2 set*, Microsoft Press, 2012. Redmond, Washington.

[6] Windows Sysinternals, http://www.microsoft.com/technet/sysinternals, 2014.11.05

[7] PSSDIAG Data Collection Utility, http://support.microsoft.com/default.aspx?scid=kb;en-us;830232, 2014.12.02

[8] Debugging Tools for Windows(WinDbg, KD, CDB, NTSD), http://msdn.microsoft.com/ko-kr/library/windows/hardware/ff551063(v=vs.85).aspx, 2014.12.02

[9] How To: Use SQL Profiler, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenethowto15.asp, 2014.12.02.

LEE JONG YUN

He received his B.S. degree from University of MyongJi University in 1994, his major is Industrial Engineering. M.S. degree from the department of Industrial Engineering, Korea University in 1996. Ph.D. degrees from the department of Software Engineering, Dong-eui University in 2015. He has worked at Microsoft Korea as a Account Executive. His current interests are database and software design.

KIM JONG SOO

He received his B.S. degree from Pukyong National University in 1992, his M.S. degree from the department of Computer Engineering, Busan University of Foreign Studies in 2003, and his Ph.D. degree from the department of Software Engineering, Dong-eui University in 2006. His current research interests are software design and web applications.

KIM TAI SUK

He received his B.S. degree from the department of electrical engineering, Kyungpook National University in 1981 and his M.S. and Ph.D. degrees from the department of computer science, Keio University in 1989 and 1993, respectively. Since 1994, he has been a faculty member of the Dong-eui University, where he is now the professor in the department of software engineering. His current research interests are information system and Internet business.