July 15, 2009

RT0865
Computer Science 6 pages

Research Report

A New Approach for Estimating Per-Transaction-Type Resource Consumption

Kiyokuni Kawachiya, Michiaki Tatsubori, and Kazunori Ogata

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Limited Distribution Notice
This report has been submitted for publication outside of IBM and will be probably copyrighted if accepted. It has been issued as a Research Report for early dissemination of its contents. In view of the expected transfer of copyright to an outside publisher, its distribution outside IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or copies of the article legally obtained (for example, by payment of royalities).

A New Approach for Estimating Per-Transaction-Type Resource Consumption

Kiyokuni Kawachiya Michiaki Tatsubori Kazunori Ogata

IBM Research, Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato, Kanagawa 242-8502, Japan
<kawatiya@jp.ibm.com>

Abstract

In a middleware environment, various types of transactions are processed simultaneously. In such an environment, it is very important to know the amount of resources necessary for processing each type of transaction. Such data can be used both for resource planning and bottleneck identification. This report proposes a new approach to estimate the per-transaction-type resource consumption in a real production environment. Two types of execution logs are collected, and resource consumption is estimated by solving simultaneous equations based on these logs.

1. Introduction

In a middleware environment such as WebSphere Application Server (WAS) [1], various types of “transactions” are processed simultaneously. In such an environment, it is very important to estimate the amount of resources (such as CPU time or memory) necessary for each type of transaction. Such data can be used both for resource planning and bottleneck identification.

Two conventional approaches to estimating resource consumption are using trace code in the middleware or using a profiler [2]. However, such tracing or profiling code slows down the execution of the application. Adding trace code depends on the internal structure of target middleware and is not so simple. In addition, if a transaction is processed with the cooperation of multiple threads or if multiple types of transactions are processed by single thread, it can be difficult to track which transaction should be credited with the consumption of various traced resources.

Another approach is to run each type of transaction *separately* in a special test environment. However, preparing such an environment is also difficult, especially for a production system.

In this report, we discuss a new approach to estimate the per-transaction-type of resource consumption in a real production environment, without adding extra trace code.

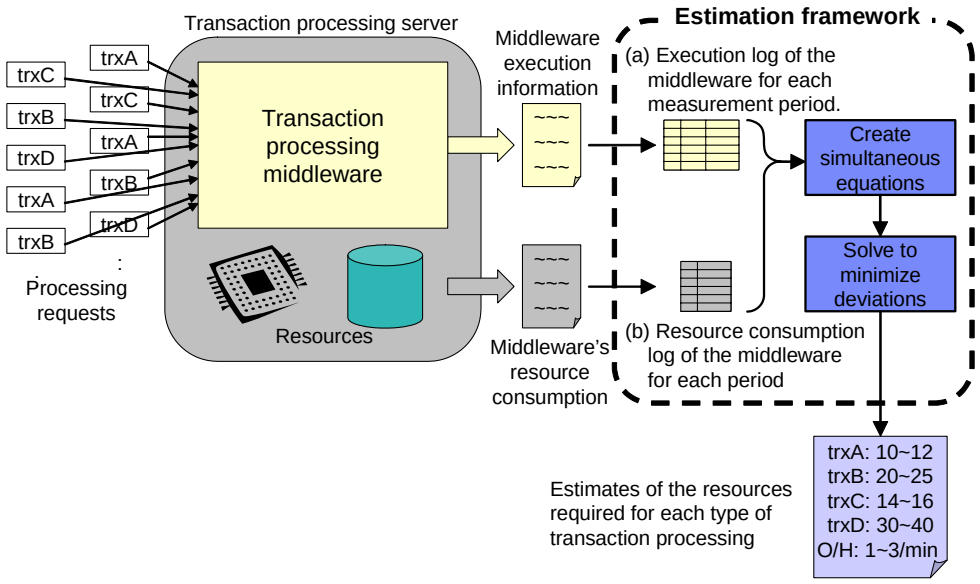


Figure 1: Overview of the estimation framework

2. Using Execution Logs for Estimation

Key idea of our approach is to estimate the resource consumption for each transaction based on external observations. More specifically, *execution logs* of the middleware as it executes various types of transactions in production mode, are used for the estimation. Figure 1 is a high-level view of the proposed estimation framework.

For the estimation, two types of logs are collected for multiple periods.

(a) The execution log of the middleware, including the numbers of each type of transactions executed during each period.

(b) A resource consumption log for the middleware for each period.

In general, such logs are already being recorded during the normal operation of the middleware for maintenance and monitoring. Therefore, it is not necessary to add new measurement code.

Next, by considering the resource consumption of each type of transaction as a variable, an equation is created for each of the measured periods. By continuing the measurements for multiple periods, we can get multiple equations. By solving these simultaneous equations while minimizing the deviations, we can estimate the resource consumption for each type of transaction.

Since the number of equations increases the longer the application executes, the resource consumption of each type can be estimated even when there are many types of transactions. In addition, since the estimation is done from normal execution logs, it can be done for the actual operating conditions, without slowing down or otherwise affecting the processing.

(a) Logs of the execution count for each transaction type

	Measured Period	trxA	trxB	trxC	trxD
Period 1	8:00~03	3 times	2 times	1 times	0 times
Period 2	8:03~08	2 times	5 times	3 times	2 times
Period 3	8:08~12	3 times	6 times	2 times	0 times
Period 4	8:12~17	5 times	1 times	5 times	3 times
Period 5	8:17~22	4 times	3 times	3 times	1 times
Period 6	8:22~25	1 times	1 times	2 times	3 times
:	:	:	:	:	:

(b) Logs of the CPU time consumption by the application during the same period

	Measured Period	CPU
Period 1	8:00~03	97 sec
Period 2	8:03~08	259 sec
Period 3	8:08~12	210 sec
Period 4	8:12~17	272 sec
Period 5	8:17~22	209 sec
Period 6	8:22~25	180 sec
:	:	:

Figure 2: Examples of the two execution logs

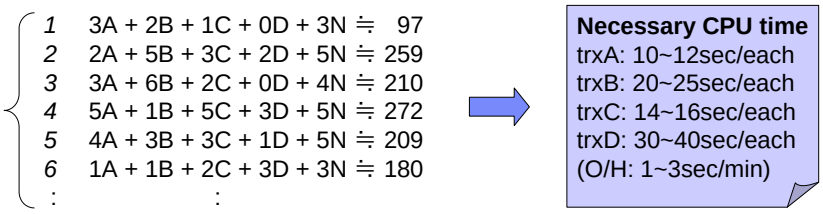


Figure 3: Simultaneous equations formulated from the logs of Figure 2, and estimated results

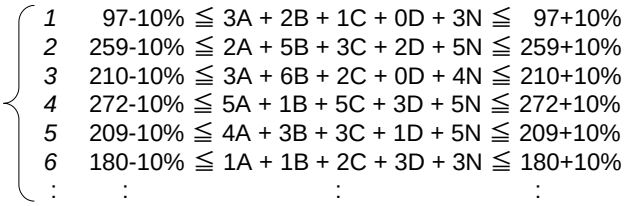


Figure 4: Simultaneous inequalities with explicit deviations

2.1 Example Estimation Flow

Figure 2 shows an example of execution logs divided into six measurement periods. These are:

(a) Logs of the execution counts for each transaction type (trxA, B, C, D).

(b) Logs of the CPU time consumption for the application during the same period.

From these logs, simultaneous equations are formulated as shown on the left side of Figure 3. Here, variables A, B, ... denote the amount of CPU time required for the processing trxA, B, ..., and variable N denotes the amount of CPU time that is consumed by the application itself in the background (per minute).

```
# ps 7684 <- Process ID of WAS
  PID TTY   STAT  TIME COMMAND
 7684 pts/0 S    13:24 /opt/IBM/WebSphere/AppServer/java/bin/java -Declipse.security
```

Figure 5: Measurement of CPU consumption by a ps command

Example of the ps command results
At 8:00, TIME=13:24. At 8:03, TIME=15:01. At 8:08, TIME=19:20.
At 8:12, TIME=22:50. At 8:17, TIME=27:22. At 8:22, TIME=30:51. ...

	Measured Period	CPU	
1	8:00~03	97 sec	<= 15:01 - 13:24
2	8:03~08	259 sec	<= 19:20 - 15:01
3	8:08~12	210 sec	<= 22:50 - 19:20
4	8:12~17	272 sec	<= 27:22 - 22:50

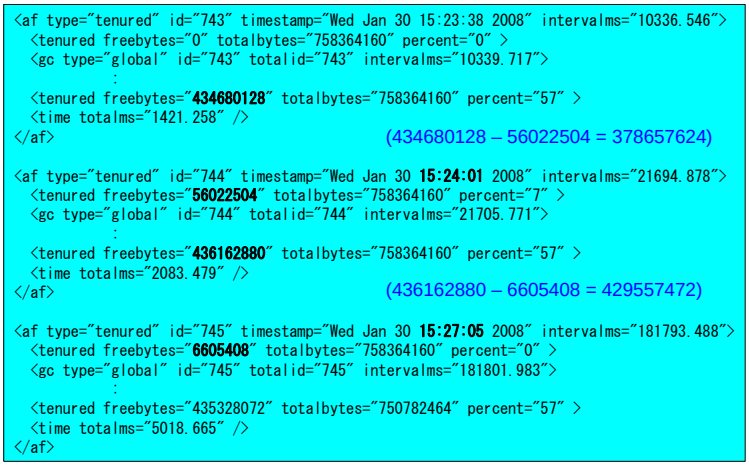
Figure 6: Examples of ps command results and calculated "CPU consumption of each measurement period" log

However, no exact solution for such simultaneous equations normally exists since there are some variations in the actual resource consumption. Therefore, we calculate the values of A, B, ... and N, that minimize the deviations. These results become the estimates of the resources required for each type of transaction processing (CPU time, in this example). For example, they can be calculated as shown on the right side of Figure 3.

To solve simultaneous equations with deviations, general multiple linear regression analysis [3] can be used. In addition, these equations can also be solved by setting up inequalities in which the deviations are explicitly taken into account, as shown in Figure 4, and then obtaining the ranges for A, B, ... and N using linear programming [4]. It is also possible to correct the estimation incrementally by using more measurement periods.

3. Actual Examples of Log Collection

There are various possible methods for collecting the resource consumption log and dividing it into multiple periods. This section describes two actual examples, for CPU consumption and object generation. In each of the following examples, only one resource consumption is estimated. This is because of making the explanation simpler, and it is also possible (and expected) to estimate multiple resource consumptions at the same time, by collecting multiple resource consumption logs simultaneously.



	Measured period	Object generation
1	15:23:39~15:24:01	378,657,624 bytes
2	15:24:03~15:27:05	429,557,472 bytes

Figure 7: Examples of “-verbose:gc” output and the calculated “object generation in each measurement period” log

3.1 Estimation of CPU Resource Consumption

The cumulative CPU consumption of a process can be obtained using a “ps” command (for UNIX-like operating systems) or any similar utility provided by the operating system. The example in Figure 5 shows that 13 minutes and 24 seconds of CPU time was consumed since the WAS process (process ID: 7684) was started. Note that this value is not the actual elapsed time, but the amount of CPU time consumed by the process.

In this example, the measurement periods were divided at specific points in time at which the fewest transactions were being processed. At each of these times, a ps command was used to record the CPU time consumed by the WAS process. The CPU consumption of each measurement period can be obtained by subtracting these values, as shown in Figure 6. By combining this with the transaction processing log for each period as generated by the WAS, we can get the necessary two types of logs separated into multiple measurement periods, then can estimate per-transaction-type CPU consumption.

3.2 Estimation of Object Generation

Another resource example is the estimation of the number of Java objects generated for each type of transaction. In Java processes such as WAS, the Java heap status at each garbage collection (GC)

point can be obtained by specifying a "-verbose:gc" startup option [5]. Figure 7 shows an example output with this option for three GC points. By looking at the reduction in the "freebytes" for these log entries, the object generation between the GC points can be calculated. For instance, 378,657,624 bytes of objects were generated between 15:23:39 and 15:24:01 in the example in Figure 7.

By considering each GC-to-GC period as the "measurement period" and separating the transaction processing log of WAS into these periods, we can get the necessary two types of logs separated into multiple measurement periods, then can estimate per-transaction-type object generation.

4. Conclusion

In this report, we showed an approach for estimating resource consumption of each type of transaction without decreasing performance. Two types of execution logs are continuously collected for multiple measurement periods, and resource consumption is estimated by solving simultaneous equations while minimizing the deviations. This approach provides methods to estimate the per-transaction-type resource consumption in a real production environment.

References

[1] IBM Corporation. WebSphere Application Server.
http://www.ibm.com/software/webservers/appserv/was/

[2] Frank Levine. JPROF - Java Profiler.
http://perfinsp.sourceforge.net/jprof.html

[3] George A. F. Seber and Alan J. Lee. *Linear Regression Analysis*, Wiley Series in Probability and Statistics, 2003.

[4] Alexander Schrijver. *Theory of Linear and Integer Programming*, John Wiley & Sons Ltd, 1986.

[5] Mattias Persson and Holly Cummins.
Java technology, IBM style: Garbage collection policies, Part 2, 2006.
http://www.ibm.com/developerworks/java/library/j-ibmjava3/