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

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

<table>
<thead>
<tr>
<th>Measured Period</th>
<th>trxA</th>
<th>trxB</th>
<th>trxC</th>
<th>trxD</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>3 times</td>
<td>2 times</td>
<td>1 times</td>
<td>0 times</td>
<td>8:00–03</td>
</tr>
<tr>
<td>Period 2</td>
<td>2 times</td>
<td>5 times</td>
<td>3 times</td>
<td>2 times</td>
<td>8:03–08</td>
</tr>
<tr>
<td>Period 3</td>
<td>3 times</td>
<td>3 times</td>
<td>2 times</td>
<td>0 times</td>
<td>8:08–12</td>
</tr>
<tr>
<td>Period 4</td>
<td>5 times</td>
<td>1 times</td>
<td>5 times</td>
<td>3 times</td>
<td>8:12–17</td>
</tr>
<tr>
<td>Period 5</td>
<td>4 times</td>
<td>3 times</td>
<td>3 times</td>
<td>1 times</td>
<td>8:17–22</td>
</tr>
<tr>
<td>Period 6</td>
<td>1 times</td>
<td>1 times</td>
<td>2 times</td>
<td>3 times</td>
<td>8:22–25</td>
</tr>
</tbody>
</table>

Figure 2: Examples of the two execution logs

\[
\begin{align*}
1 & : 3A + 2B + 1C + 0D + 3N \approx 97 \\
2 & : 2A + 5B + 3C + 2D + 5N \approx 259 \\
3 & : 3A + 6B + 2C + 0D + 4N \approx 210 \\
4 & : 5A + 1B + 5C + 3D + 5N \approx 272 \\
5 & : 4A + 3B + 3C + 1D + 5N \approx 209 \\
6 & : 1A + 1B + 2C + 3D + 3N \approx 180 \\
\end{align*}
\]

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

\[
\begin{align*}
1 & : 97-10\% \leq 3A + 2B + 1C + 0D + 3N \leq 97+10\% \\
2 & : 259-10\% \leq 2A + 5B + 3C + 2D + 5N \leq 259+10\% \\
3 & : 210-10\% \leq 3A + 6B + 2C + 0D + 4N \leq 210+10\% \\
4 & : 272-10\% \leq 5A + 1B + 5C + 3D + 5N \leq 272+10\% \\
5 & : 209-10\% \leq 4A + 3B + 3C + 1D + 5N \leq 209+10\% \\
6 & : 180-10\% \leq 1A + 1B + 2C + 3D + 3N \leq 180+10\% \\
\end{align*}
\]

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 time 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).
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. ...
During 15:23:39~15:24:01, 378,657,624 bytes of objects were generated

During 15:24:03~15:27:05, 429,557,472 bytes of objects were generated

<table>
<thead>
<tr>
<th>Measured period</th>
<th>Object generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 15:23:39~15:24:01</td>
<td>378,657,624 bytes</td>
</tr>
<tr>
<td>2 15:24:03~15:27:05</td>
<td>429,557,472 bytes</td>
</tr>
</tbody>
</table>

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

http://www.ibm.com/software/webservers/appserv/was/

http://perfinsp.sourceforge.net/jprof.html

