Measurement of memory usage in J2EE applications

Arupratan Santra1, I V Murali Krishna2 and Anindita Das3
1Jawaharlal Nehru Technological University, Infosys Technologies Ltd, Plot 4, DBH Nagar, 101-Santosh Nivas, Santosh Nagar, Hyderabad 500 059, India
2Jawaharlal Nehru Technological University, Hyderabad, India
3Research Center Imarat, DRDO, Hyderabad, India

Received 02 February 2009; revised 05 June 2009; accepted 08 June 2009

This paper describes different methodologies to measure memory usage in web enabled J2EE application and demonstrates proposed solutions for memory usage measurement while running multithreaded web enabled J2EE application with variable user load.

Keywords: J2EE, Endurance point, Flat load, Memory usage analysis, Ramp-up load

Introduction

High memory usage of application server is related to low performance of multithreaded application. Memory usage problems are slow down and crashing of application, transactional data corruption and very low throughput due to high memory usage1-4. Significant amount of memory is consumed by an application instance, even if it receives no requests5. Memory consumption is often related to prior application usage rather than to its current load. Because an accurate projection of future memory usage is difficult and many applications cannot run when system is out of memory, it is reasonable to use a conservative estimation of memory usage.

Java Virtual Machine (JVM) handles memory usage6,7 but run poorly due to improper optimization of JVM code. If JVM runs out of memory8, it can create problems with garbage collection4,9 or cause a Java application not to free up memory/objects, also known as a memory leak. Among different approaches to measure memory usage for Java applications, Long Soak Run10 has a fixed workload while monitoring application throughput and memory consumption. Though this approach can identify memory usage as a whole, but it does not provide sufficient information to identify memory usage for different transactions. It is very difficult to figure out memory percentage used or left over to run the application. It also cannot measure directly memory usage percentage. It also takes long time to run the test and it becomes a limitation when there are number of changes in the application and test to be run for multiple times.

This study presents a web enabled J2EE application for measurement of memory usage.

Experimental

Memory usage is measured in three phases under ramp-up and flat load conditions, with and without changing number of virtual users (VU). Web Performance Trainer (WPT)11, a performance-testing tool and websphere application server, measures different database size. Dividing complete application into small cases creates performance test script for web application. While measuring performance parameters (memory usage, throughput etc.), these cases are grouped into profiles. Web application has been developed in J2EE technology for distance calculation, route direction, address of specific locations etc. When user starts WPT, default set browser is launched in client machine. User then login to access application and performs different transactions as planned in set of performance test scripts.

Phase 1

In Phase 1, application is tested for ramp-up load condition. Number of VUs at start-up is kept at 20 and is
increased at the rate of 10 per min. Test is repeated for three different scenarios and testing time for each scenario is 5 min. So total VUs achieved after each scenario are 60.

**Scenario 1**
Server has started newly and all other application has been stopped other than application under test. System is run to perform 25 transactions continuously.

**Scenario 2**
Server is running and another 25 transactions have been performed continuously. This transaction has been performed for 2 cycles. Performance metrics have been measured by WPT after each cycle with a time gap of 2 h.

**Scenario 3**
Server is restarted. No new transaction is performed and WPT measures memory usage.

**Phase 2**
In this phase, database size and server condition are same as Scenario 3 of Phase 1. Start up VU is 50 and finally achieved VU is 200.

**Scenario 1**
Measurement is carried for extended time (8 min) with a slow ramp up load (21 VU/min).

**Scenario 2**
Measurement is carried for normal time (4 min) with a high ramp up load (50 VU/min).

**Phase 3**
In this phase, measurement is carried out with a flat load condition. Server has restarted after performing Phase 1 and Phase 2. All other applications have been stopped except J2EE. System is run to perform 25 transactions continuously. Performance metrics have been measured by WPT. VU is fixed to 5 till end of 5-min test run.

**Results and Discussion**
Sharp (20%) increase of memory usage is observed after 25 transactions (Table 1) in Phase 1. There is no memory release after restarting of server (Fig. 1). Memory usage percentage increment is due to steady increase of load on server and there are no free threads initially in the server. So it starts with a good positive value instead of low value. Memory usage increases steadily, and at the same time throughput and total hits are in decreasing trend.

In Phase 2, if time span increases for test execution then server load increases and shows result as “OVERLOADED” in WPT (Table 2). In case of decrease in time span, it shows “LOW MEMORY” in WPT. Server is crashed after Phase 2.

In Phase 3, both minimum and maximum memory usages percentage increases steadily (Fig. 2). Memory usage accumulates due to live reference object. When system reaches saturation point, throughput of server plateaus, and as server load continues to grow, response time of system also grows. Memory usage percentage increment is due to steady increase of load on server and there are no free threads initially on server. Server crashes after Phase 3. Phase 1 and Phase 2 are used for identification of memory usage problem. Phase 3 will not perform if no problem is noticed after two test runs in

**Table 1—Overall performance test metrics for Phase 1**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Server condition</th>
<th>Time gap between scenario, h</th>
<th>Hits/s (Max)</th>
<th>Total hits</th>
<th>Throughput KB/s</th>
<th>Memory usage, %</th>
<th>Server load, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Restarted</td>
<td>0</td>
<td>21</td>
<td>3222</td>
<td>849</td>
<td>45</td>
<td>58</td>
</tr>
<tr>
<td>2 (First cycle)</td>
<td>Running</td>
<td>2</td>
<td>22</td>
<td>3179</td>
<td>864</td>
<td>64</td>
<td>55</td>
</tr>
<tr>
<td>2 (Second cycle)</td>
<td>Running</td>
<td>2</td>
<td>22</td>
<td>2921</td>
<td>625</td>
<td>70</td>
<td>47</td>
</tr>
<tr>
<td>3</td>
<td>Restarted</td>
<td>0.25</td>
<td>20</td>
<td>2320</td>
<td>691</td>
<td>73</td>
<td>42</td>
</tr>
</tbody>
</table>

FIG. 1—Memory usage graph of Phase 1

Fig. 1—Memory usage graph of Phase 1
Phase 1 and Phase 2. Memory is a load independent resource. So flat load condition is better for getting correct memory usage percentage.

**Proposed Solution**

Proposed approach is a batch approach with a flat load condition. Performance metrics have been measured by WPT. Test experiment has performed with a start up VU of 5, which is fixed till end of 5-min test run. When server allowed to be idle then all session objects get released and memory gets freed. So memory usage is low and steady (Fig. 3). With a same load condition of Phase-3, it shows that memory usage is as plateaus. This proves that memory usage is due to application abnormalities but not due to VU load. Thus memory is a load independent resource. So Phase 3 is correct approach to measure accurate memory usage.

**Conclusions**

A new solution is proposed to use memory of an application server without leakage in single or multiple transactions and variable VU load conditions. Proposed solution is successfully demonstrated on web enabled J2EE architecture.

**References**