Scenario Care Load Testing in TTCN-3

Ji Wu
BeiHang University, China
Agenda

• Load testing in TTCN-3
• Load profile model
• Load control
• Test System Framework
• Virtual user Implementation
• Reuse existing test case
• Experiments
• Conclusion
Structure of Load testing

• Load testing is an important way to explore SUT load and the bottleneck.

A group of Virtual Users to implement the load on SUT

A Controller coordinates the Virtual Users

The coordination satisfies the given parameters
Load Testing in TTCN-3

• Load testing in TTCN-3
  – Test cases/functions for implementing load in the name of virtual users
  – Controller is part of load strategy

• Concerns on load testing
  – More virtual users
  – More varieties of user behaviors
  – More realistic load profile

• Need piles of TTCN-3 code to implement the concerns!
Load Testing in TTCN-3

• Load testing usually goes after successful functionality testing
• The existing test cases can guarantee
  – The varieties of behaviors
  – The realistic of load profile
• Two issues come
  – How to reuse the existing test cases?
  – How to coordinate the VUs with the test cases to get good load profile?
Load Testing in TTCN-3

• How to reuse the existing test cases?
  – a question of code conversion and generation

• How to coordinate the VUs …?
  – A question of adaptive control
    • assign test cases for VUs
    • observe load profile
    • evaluate against given parameters
    • modify the assignment
Load Profile Model

Several relative metrics of load => clusters

- Number of concurrent users
- Proportion of various VUs
- Proportion of various requests
- Byte Distribution
- Session length
- Session duration
- Session interval
- Request interval within session
- Request interval
- Response time
- Thinking time
- Overall request interval
### Cluster Domination Equations

<table>
<thead>
<tr>
<th>Equation</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>session duration = request interval within session * session length</td>
<td></td>
</tr>
<tr>
<td>request interval overall = request interval within session / number of concurrent users</td>
<td></td>
</tr>
<tr>
<td>number of concurrent users = session duration / session interval</td>
<td></td>
</tr>
<tr>
<td>session interval = overall request interval * session length</td>
<td></td>
</tr>
<tr>
<td>request interval within session = thinking time + response time</td>
<td></td>
</tr>
</tbody>
</table>
Load Profile Control Points

- Five control points in two stages of control
  - Static control (pre-testing): proportions of various VUs → controlling session length, requests distri., byte distri.
  - Dynamic control (within-testing): #concurrent users, session interval, overall request interval, thinking time
Static Load Control

- Each kind of VU has pre-determined sequence of requests
- Given the proportions of various requests (PVR), to setup the number of different kinds of VUs

<table>
<thead>
<tr>
<th>VU1</th>
<th>VU2</th>
<th>PVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>logon</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>add</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>add_s</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>list</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>detail</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>remove</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>exit</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 2 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \\ 1 \\ 1 \\ 5 \\ 1 \\ 0 \end{pmatrix} \]

\[ \text{NOM} = \begin{pmatrix} 7% \\ 14% \\ 14% \\ 31% \\ 20% \\ 7% \end{pmatrix} \]
Dynamic Load Control

• We design seven ways to control the load dynamically.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Points to Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Thinking time (distribution)</td>
</tr>
<tr>
<td></td>
<td>#concurrent users (value)</td>
</tr>
<tr>
<td>2</td>
<td>Thinking time (distribution)</td>
</tr>
<tr>
<td></td>
<td>Overall request interval (value)</td>
</tr>
<tr>
<td>3</td>
<td>Thinking time (distribution)</td>
</tr>
<tr>
<td></td>
<td>Session interval (distribution)</td>
</tr>
<tr>
<td>4</td>
<td>Overall request interval (distribution)</td>
</tr>
<tr>
<td>5</td>
<td>Thinking time (distribution)</td>
</tr>
<tr>
<td>6</td>
<td>#concurrent users (value)</td>
</tr>
<tr>
<td>7</td>
<td>Session interval (distribution)</td>
</tr>
</tbody>
</table>
Performance Testing Scenario

- Provides phase-based load profile and control chance
- Provides control on the types of virtual users
- It can construct complex test scenario with required load profile, and simulate the real usages.
Test System Framework

- **SA Delegate**: Manage the communication connections to the SUT

- **DMP-PA**: Manages the Dynamic Module Parameters (DMP)

- **TL**: Manages virtual users, Monitors the test running, measures performance
Virtual User Implementation

• Two levels of concurrency
  – Concurrency among virtual users
  – Concurrency within a virtual user

• Three levels of test components
  – MTC: manages the concurrency of virtual users
  – V-PTC: manages the concurrency within a virtual user, maintains a unique sessionID
  – PTC: manages the communication with other test component, maintains a unique componentID
Virtual User Implementation

- Performance Test Case runs on MTC
- Session runs on V-PTC
- Send-Receive Pairs runs on PTC
Load Control Implementation

• Load control is part of test system behavior.
  – A top level test case, keeps on running until test finished

• It monitors the relative ports to receive load control instruction
  – Read load parameters from DMP-PA
  – Create/Stop V-PTC
  – Start ready V-PTC with associated session
  – Control the V-PTC start interval
  – Control the number of live V-PTC
Monitoring of Virtual User Behavior

- Record the time to send and receive messages within a session
- Save as log file
- Calculate the load metrics from the log
- As feedback for adaptive load control
Performance Test Case

• Similarities with functionality test case
  – Similar test behavior (send/receive)
  – Use the same codec and adapter

• Differences from functionality test case
  – More powerful test configuration (three levels of test components)
  – Add the session hierarchy (runs on V-PTC)
  – Need dynamic module parameters management

• Automatically rewrite the test behavior in functionality test case
Conversion Rules

• Downgrade functionality test case

\[
\text{testcase } \langle \text{tc\_name} \rangle() \text{ runs on } \text{compType\_name1} \}\{\text{statements}\} \\
\text{function subst}\_\langle \text{tc\_name} \rangle(\text{charstring} \text{sessionID}) \text{ runs on } \text{compType\_name1}\{\text{statements}\}
\]

• Change references of \text{mtc} to \text{v\_ptc}

• Get module parameter value from DMP-PA

\[
\text{statementWith(ModuleParameter)}; \\
\text{var ModuleParaType } \text{mp\_subst}; \text{mp\_subst:=getMpSubst(sessionID)}; \text{statementWith(ModuleParameter)};
\]

• \text{getMpSubst} is an external function implemented in DMP-PA
Conversion Implementation

*Parsing with TRex.*
Experiment A

- Load testing for a simple chat system
Experiment A

- Load test case after automatic convert
Experiment B

![Graph showing average concurrent users and their corresponding metrics: successful request response time (0.6 second), request success rate, and average throughput (26 per second).]
Experiment B

- SUT is a web-based bookstore
  - JSP for user interface and business logic
  - MySQL for data management and access
Experiment B

- 11 functionality test cases cover the flows
- Use our own distribution platform to support the test
**Experiment B**

### Testcases

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Len</th>
<th>SIXTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>t001</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>t002</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>t003</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>t004</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>t005</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>t006</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>t007</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>t008</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>t009</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>t010</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>t011</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

### Campaign

<table>
<thead>
<tr>
<th>ID</th>
<th>Dur</th>
<th>SessionDate</th>
<th>ConcurrCPU</th>
<th>GlobalOffer</th>
<th>ThinkingTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>SessionDate</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>
Experiment B

- Response time (2.5 seconds)
- Throughput (45/second)
- Success rate
Conclusion

• This work is supported by 2 National High-Tech program projects.
• We develop the distributed load testing in TTCN-3
  – Profile based load generation
  – Adaptive load control
  – Reuse existing functionality test cases
• TTCN-3 language provides great flexibility and good structure for load testing.
Questions?

- Thanks for your attention!