Tuning and development with SIP Servlet on Mobicents

Naoki Nishihara
OKI Electric Industry Co., Ltd
Jean Deruelle
Mobicents Sip Servlets Lead
Agenda

• Introduction
• SIP Application behavior
• How to tune the JavaVM for SIP Servlet
• How to develop SIP Servlet Application with Frameworks
Introduction

– Mobicents fellow
  • Member of mobicents core team from March 2011
– Leading mobicents SSF project
  • For developing SIP Servlet Application with Spring Framework
– OKI has been involved SIP Servlet development since 2003
About OKI

• Founded
  – January 1881

• Major Operation
  – Manufacturing and sales of products, technologies, software and solutions for telecommunications systems and information systems
Current works

• Support Japanese carrier
  (They are evaluating the OSS platform)

• Customize MSS for proprietary
  \textit{ex) NOT include internal IP Address in SIP header fields}

• Developing customized SIP Load balancer
SIP Servlet Behavior
SIP Servlet Behavior (1)

• To establish one session, some messages are sent and received.
Basic B2BUA sequence

- 2 dialogs
- 6 transactions
- 12 messages
Many objects will be generated in one sequence.
Created instances

After 10,000 calls (NOT TERMINATED SESSION)

<table>
<thead>
<tr>
<th>Instance counts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>gov.nist.core. NameValueList</td>
<td>810,041</td>
</tr>
<tr>
<td>NameValue</td>
<td>330,015</td>
</tr>
<tr>
<td>HostPort</td>
<td>310,036</td>
</tr>
<tr>
<td>Host</td>
<td>310,036</td>
</tr>
<tr>
<td>DuplicateNameValueList</td>
<td>290,049</td>
</tr>
<tr>
<td>MultiValueMapImpl</td>
<td>290,049</td>
</tr>
<tr>
<td>gov.nist.javax.sip. SipURI</td>
<td>240,006</td>
</tr>
<tr>
<td>Authority</td>
<td>240,006</td>
</tr>
<tr>
<td>AddressImpl</td>
<td>210,003</td>
</tr>
</tbody>
</table>
Mobicents SIP Servlet own behavior

- JAIN-SIP, SIP Servlet, JBoss(J2EE)...

About 200 Threads run on MSS (for transport, timer, cluster...)

JUDCon 2011: Boston
Other problems related to SIP

• Retransmissions will occur, if response was sent late over 500ms with UDP transport
  – In normal case, UDP transport will be used.

• If JVM was stopped long time by GC
  – Call failure will be occurred and error handling may not work properly.
  – SIP has many timers, they could not work properly.

• It’s relatively easy to retry with HTTP, but you would have to start over from scratch to initiate session with SIP
Conclusion

• MSS will discard many objects in one session.
• When retransmissions of UDP messages occurred,
  – Many threads runs at the same time
  – Many objects are discarded
  – CPU usage go up
  – GC will be missed
  – Full GC will run

Fall in a vicious cycle
How To Tune the Java VM
Plan

• Reduce retransmissions of UDP
  – Response time less than 200ms
  – Measure the performance on tuned JVM
    • 400call/sec(1440,000BHCA)

• Effective use of multiple CPU
  – Most of server-machine have multiple CPU

• Reduce pause times by “Stop-The-World”
Test Sequence

About 16,000 sessions will remain on JVM
Recommended Sun Java VM Options

- `-server`
  - `-XX:+UseTLAB` is enabled.
- `-XX:+UseConcMarkSweepGC`
  - CMS GC reduce the “Stop the World”.
- `-XX:+CMSIncrementalMode`
  - Enable the incremental mode. You should tune the below options
    - `-XX:CMSIncrementalDutyCycle=<N>`
    - `-XX:CMSIncrementalDutyCycleMin=<N>`
Other performance options (1)

- **-XX:MaxTenuringThreshold=0 -XX:SurvivorRatio=128**
  - Object that related to SipSessions will NOT be collected in NewGC. This option will make the full NewSize available to every NewGC cycle.
- **-XX:+UseParNewGC**
  - Enable a parallel young generation GC. (for multiple cpu machine)
- **-XX:CMSInitiatingOccupancyFraction=75**
  - Set the level at which the collection is started
- **-XX:+CMSParallelRemarkEnabled**
  - Reduce remark pause with multi threads
Other performance options (2)

- `-XX:+UseStringCache`
  - Enables caching of commonly allocated strings

- `-XX:+OptimizeStringConcat`
  - Optimize String concatenation operations where possible

- `-XX:+UseCompressedStrings`
  - Use a byte[] for Strings which can be represented as pure ascii

- `-XX:+UseCompressedOops`
  - Enables the use of compressed pointers for optimized 64-bit performance
Data and results

1. Default GC
2. CMS GC without tuning the duty cycle
3. CMS GC with tuning the duty cycle
4. Add parallel NewGC and performance options

Analyze:
- GC, CPU usage, failed call, succeeded call,
- Retransmissions, Average response time,
GC options (1)

-Xms6g
-Xmx6g
-XX:PermSize=256m
-XX:MaxPermSize=256m
Default GC

Over thousands failed calls
And retransmissions

“Stop The World”
about 8 seconds

Thousands responses
took more than 200ms
GC options (2)

-Xms6g
-Xmx6g
-XX:PermSize=256m
-XX:MaxPermSize=256m

-XX:UseConcMarkSweepGC
-XX:+CMSIncrementalMode
CMS GC without tuning the duty cycle

- CPU usage increased
- 0 failed call
- GC pause time less than 100ms
- Decreased time-consuming response
GC options (3)

-XX:UseConcMarkSweepGC
-XX:+CMSIncrementalMode
-XX:-CMSIncrementalPacing
-XX:CMSIncrementalDutyCycle=100
-XX:CMSIncrementalDutyCycleMin=100
CMS GC with tuning the duty cycle

- CPU usage decreased
- Retransmissions decreased
- GC pause time less than 100ms
- Decreased time-consuming response
GC options (4)

-XX:-CMSIncrementalPacing
-XX:CMSIncrementalDutyCycle=100
-XX:CMSIncrementalDutyCycleMin=100

-XX:+UseStringCache
-XX:+OptimizeStringConcat
-XX:+UseCompressedStrings
-XX:MaxTenuringThreshold=0
-XX:SurvivorRatio=128
-XX:+UseParNewGC
-XX:+UseCompressedOops
-XX:CMSInitiatingOccupancyFraction=75
-XX:+CMSParallelRemarkEnabled
Add parallel NewGC and performance options

- CPU usage decreased
- Retransmissions increased
- Memory usage decreased
- Increased time-consuming response
Results

• Recommended VM options are useful.
  – Reduce pause times
  – No failed calls
  – Reduce retransmissions

• Tuning options of the duty cycle are very useful

• Other performance options are slightly useful
Developing SIP Servlet with frameworks
SIP Servlet Frameworks

Mobicents has 2 SIP Servlet Frameworks

• SSF (Spring Signaling Framework)
  – Based on Spring Framework

• CTF (CDI-Telco-Framework)
  – CDI Based framework

These frameworks have same concept…

**Simplify SIP Servlets development**

We want to merge these framework’s functions and support many developers!
Spring Signaling Framework
SSF
How to develop SIP Servlet application with SSF
Why using Spring Framework?

• Standard DI Container
  – Familiar to many Java Developers

• Customizable Context
  – SSF provide customized Context for SIP Servlet

• Many functionality modules are working on Spring Framework.
  – You can create the converged application with Spring Framework modules.
Create SIP Servlet with POJO

```java
@Component
public class ProxyHandler {

@Autowired
ProxyBean proxyBean;

@Autowired
CheckRequireBean checkRequire;

@SipServletRequestMapping(methods = { "INVITE", "UPDATE", "MESSAGE", "PUBLISH" })
public void handleRequest(SipServletRequest req) throws Exception {
    // start
    checkRequire.handleRequest(req);
    if (req.isCommitted()) {
        return;
    }
    proxyBean.startProxy(req);
    // end
}
....
```
Easy to add functions

• Of course, you can use AOP function
• Try to add the Call-Blocking function to the proxy service. 😊
Add Call-Blocking function (1)

• Create new POJO for Call-Blocking

```java
@Component
public class CallBlockingBean {

    public void handleRequest(SipServletRequest req) throws IOException {
        if (isBlocked(req)) {
            SipServletResponse res = req.createResponse(SipServletResponse.SC_SERVICE_UNAVAILABLE);
            res.send();
            return;
        }
    }
    ....
```
Add Call-Blocking function(2)

• Create Aspect class for Advice

```java
@Aspect
public class ProxyAround {
    @Autowired
    CallBlockingBean callBlocking;

    @Around(
        "execution(ProxyHandler.handleRequest(..)) && args(req)"
    )
    public void handleRequest(ProceedingJointPoint pjp, SipServletRequest req) throws IOException {
        callBlocking.handleRequest(req);
        pjp.proceed(new Object[]{req});
    }
}
```
Add Call-Blocking function (3)

- Add configurations for AOP

```xml
<aop:aspectj-autoproxy proxy-target-class="true"/>
<context:component-scan base-package="org.mobicents.ssfsfexamples.sip.beans">
    <context:include-filter type="aspectj" expression="org..ProxyAround*"/>
</context:component-scan>
<context:component-scan base-package="org.mobicents.ssfsfexamples.sip.sipserver"/>
```

Just Do It! 😊
CDI-Telco-Framework
CTF
CDI JSR-299

CDI is the Java standard for dependency injection and contextual lifecycle management, led by Gavin King for Red Hat, Inc. and is a Java Community Process (JCP) specification that integrates cleanly with the Java EE platform. Any Java EE 6-compliant application server provides support for JSR-299 (even the web profile).

• Loose coupling with strong typing
• Well-defined lifecycle for stateful objects bound to lifecycle contexts
• Support for Java EE modularity and the Java EE component architecture
**CDI-Telco-Framework**

Mobicents brings the power and productivity benefits of CDI into the Mobicents Sip Servlets platform providing dependency injection and contextual lifecycle management for converged HTTP/SIP applications.
CDI-Telco-Framework

Mission Statement

• simplify SipServlets development by introducing a clean programming model
• ease of development by making available SIP utilities out of the box
• providing dependency injection and contextual lifecycle management to the SipServlets.
public class SipRegistrarClient {

....
@Inject
SipFactory sipFactory;

protected void doRegister(@Observes @Register SipServletRequest req) throws ServletException, IOException {
    .... register user here ...
}
}
Contact

- Naoki Nishihara
  - nisihara562@gmail.com
- Jean Deruelle
  - jean.deruelle@gmail.com
- Georges
  - gvagenas@gmail.com
- Vladimir Ralev
  - vladimir.ralev@gmail.com
Gratitude
Appendix
Configuration

• mss-1.6.0-snapshot-jboss-jdk6-5.1.0GA-1103310643
  – With Simple B2BUA application
• SIPp as UAC & UAS
• Jdk1.6.0_24
• RHEL5 2.6.18-164.el5
Target server machine

HP PROLIANT DL360 G6(504634-291)
• Intel(R) Xeon(R) CPU E5540 @ 2.53GHz stepping 05
  – 4 core
• Memory: 32G
Default GC
Long Load
Over Load