Monitoring tomcat web application in production

Leon Rosenberg (@dvayanu)
Hello...
Todays Contents

- Basic Concepts of MoSKito
- Burgershop Demo
- Recap
- Integration (if time allows)
- Q & A
Why MoSKito?
MoSKito Core Team

Producer

Accumulator

Threshold

Journey
Collect data
Organise data
Inspect Intervals

<table>
<thead>
<tr>
<th>5m</th>
<th>5m</th>
<th>5m</th>
<th>5m</th>
<th>5m</th>
<th>5m</th>
<th>5m</th>
<th>5m</th>
</tr>
</thead>
<tbody>
<tr>
<td>15m</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15m</td>
<td></td>
</tr>
</tbody>
</table>
Discover and analyse
Scale and get in control
DEMO TIME

http://burgershop-hamburg.demo.moskito.org/burgershop/

http://burgershop-hamburg.demo.moskito.org/burgershop/moskito-inspect/mskDashboard?dashboard=Burger+Board
Why MoSKito?

- Performance data collection
- Application Insights with MoSKito Inspect
- Code and Dataflow Tracing with Journeys
- Error Management
- System overview with MoSKito Control
Why MoSKito?
The big picture
Q & A Time?

http://www.moskito.org

https://itunes.apple.com/de/app/msk-control/id688838411?l=en&mt=8

https://github.com/anotheria/moskito
https://github.com/anotheria/moskito-control
https://github.com/anotheria/moskito-demo

http://search.maven.org/#search%7Cga%7C1%7Cmoskito
Integration

- AOP / CDI / Spring
- Proxies
- WEB
- Java Agent

Guide: https://confluence.opensource.anotheria.net/display/MSK/Integration+Guide
@Monitor
public class YourClass {

@Monitor
public class YourClass {
public void firstMonitoredMethod(){...}
@Monitor public void secondMonitoredMethod(){...}
public void notMonitoredMethod(){...}

@Monitor
public class YourClass {
public void thisMethodWillBeMonitored(){...}

@DontMonitor public void thisMethodWillBeExcludedFromMonitoring(){

@Count
public class PaymentCounter {

@Count
public class PaymentCounter {

@CountByParameter
void ingredientUsed(String ingredient) {

/**
 * Electronic card payment (lastchrisfteinzug in germany).
 */
public void ec(){
/**
 * Credit card payment.
 */
public void cc(){
/**
 * Payment via paypal.
 */
public void paypal(){
}
<dependencies>
  <dependency>
    <groupId>net.anotheria</groupId>
    <artifactId>moskito-core</artifactId>
    <version>2.8.7</version>
  </dependency>
  <dependency>
    <groupId>net.anotheria</groupId>
    <artifactId>moskito-aop</artifactId>
    <version>2.8.7</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>aspectj-maven-plugin</artifactId>
      <version>1.4</version>
      <configuration>
        <aspectLibraries>
          <aspectLibrary>
            <groupId>net.anotheria</groupId>
            <artifactId>moskito-aop</artifactId>
            <version>2.8.7</version>
          </aspectLibrary>
        </aspectLibraries>
        <source>1.7</source>
        <target>1.7</target>
      </configuration>
      <executions>
        <execution>
          <goals>
            <goal>compile</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
@Monitor
@ProducerRuntime(producerId="Foo", category="my")

public class YourClass {

    @Monitor
    public class YourClass {
        @Monitor(MonitoringCategorySelector.WEB)
        public class YourClass {
            public void thisMethodWillBeMonitored(){...}
        }
    }

    @DontMonitor public void thisMethodWillBeExcludedFromMonitoring(){

        @Count
        public class PaymentCounter {

        }

        @CountByParameter
        void ingredientUsed(String ingredient) {

        }

    }

    public class PaymentCounter {
        /**
         * Electronic card payment (lastchrisfteinzug in germany).
         */
        public void ec(){
            /* Credit card payment.
            */
            public void cc()

            /* Payment via paypal.
            */
            public void paypal()
        }
    }

}
    <interceptors>
        <class>net.anotheria.moskito.integration.cdi.CountInterceptor</class>
        <class>net.anotheria.moskito.integration.cdi.CallInterceptor</class>
        <class>net.anotheria.moskito.integration.cdi.WebCallInterceptor</class>
        <class>net.anotheria.moskito.integration.cdi.ServiceCallInterceptor</class>
        <class>net.anotheria.moskito.integration.cdi.DaoCallInterceptor</class>
    </interceptors>
</beans>
public interface SimpleService{
    void doSomethingMethod();
}

public class SimpleServiceImpl implements SimpleService{
    public void doSomethingMethod(){
    }
}

SimpleService service = ProxyUtils.createServiceInstance(new SimpleServiceImpl(), "default", SimpleService.class);
<filter>
  <filter-name>RequestURIFilter</filter-name>
  <filter-class>net.anotheria.moskito.web.filters.RequestURIFilter</filter-class>
  <init-param>
    <param-name>limit</param-name>
    <param-value>1000</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>RequestURIFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

Automagically with servlet 3.0

<filter>
  <filter-name>DomainFilter</filter-name>
  <filter-class>net.anotheria.moskito.web.filters.DomainFilter</filter-class>
  <init-param>
    <param-name>limit</param-name>
    <param-value>50</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>DomainFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
Inspect

<dependency>
    <groupId>net.anotheria</groupId>
    <artifactId>moskito-inspect-remote</artifactId>
    <version>2.8.7</version>
</dependency>
Success stories

Far too many, but here are some :-)
After a release of a new version huge traffic increase on one of the databases was detected.

The database in question was used by a service. There were 20 clients (code components) using this service.

MoSKito showed that 55% of the traffic to the service came from one client. MoSKito Inspect showed which client was producing most traffic.
Closer inspection (code review) of the client revealed a bug which led to double calls to the service.

Incident solved in 30 minutes.
Application overall performance was insufficient.

With moskito journeys and call tree analysis we were able to find redundant calls to the backend and remove them.

Request duration reduced to 50% with 4 hours analysis and 4 hours coding effort.
<table>
<thead>
<tr>
<th>Function</th>
<th>Time in milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoginAPI.isLoggedln() = true</td>
<td>28</td>
</tr>
<tr>
<td>RegistrationAPI.isMeTest() = false</td>
<td>12</td>
</tr>
<tr>
<td>PaymentAPI.amlPremium() = true</td>
<td>665</td>
</tr>
<tr>
<td>PaymentBusinessServiceDiMe 1.readActivePayments(372347) = [PaymentBO(id='159538', accountid='3723...</td>
<td>585</td>
</tr>
<tr>
<td>IASSiteDataService-2.getNavItem(33) = NavItem [33] name: Hilfe, title: Hilfe, externalLink: , p...</td>
<td>21</td>
</tr>
<tr>
<td>LoginAPI.isLoggedln() = true</td>
<td>9</td>
</tr>
<tr>
<td>LoginAPI.isLoggedln() = true</td>
<td>767</td>
</tr>
<tr>
<td>LoginAPI.getLoggedUserId() = 372347</td>
<td>9</td>
</tr>
<tr>
<td>RegistrationAPI.isMyEmailConfirmed() = true</td>
<td>10</td>
</tr>
<tr>
<td>PaymentAPI.amlPremium() = true</td>
<td>10</td>
</tr>
<tr>
<td>PaymentBusinessServiceDiMe 1.readActivePayments(372347) = [PaymentBO(id='159538', accountid='3723...</td>
<td>603</td>
</tr>
<tr>
<td>LoginAPI.getMyLoginTime() = 1271890272896</td>
<td>562</td>
</tr>
<tr>
<td>LoginAPI.isLoggedln() = true</td>
<td>25</td>
</tr>
<tr>
<td>LoginAPI.isLoggedln() = true</td>
<td>2</td>
</tr>
<tr>
<td>LoginAPI.getPreviousLoginTime() = 1271651945362</td>
<td>15</td>
</tr>
<tr>
<td>LoginAPI.isLoggedln() = true</td>
<td>2</td>
</tr>
<tr>
<td>IASWebDataService-2.getBox(167) = Box [167] name: Messagesbox: Cleanup Warning, content: &lt;p&gt;Kon...</td>
<td>30</td>
</tr>
<tr>
<td>RegistrationAPI.isMeTest() = false</td>
<td>13</td>
</tr>
<tr>
<td>PaymentAPI.amlPremium() = true</td>
<td>668</td>
</tr>
<tr>
<td>PaymentBusinessServiceDiMe 1.readActivePayments(372347) = [PaymentBO(id='159538', accountid='3723...</td>
<td>605</td>
</tr>
<tr>
<td>IASWebDataService-2.getBox(1) = Box [1] name: Footer, content: &lt;p&gt;© (cal:currentYear) {text:brand...</td>
<td>13</td>
</tr>
</tbody>
</table>
A bug only reproduceable by one user.

Second page of received messages won’t show. No exception logged.
Broken Call

red color indicates that the method was terminated abnormally (exception)

possible reason - user didn’t exist in the db and the code didn’t know how to handle it.
End of Data

Thank you.

http://www.moskito.org

https://itunes.apple.com/de/app/msk-control/id688838411?l=en&mt=8

https://github.com/anotheria/moskito
https://github.com/anotheria/moskito-control
https://github.com/anotheria/moskito-demo

http://search.maven.org/#search%7Cga%7C1%7Cmoskito