MoSKito

for users.

by Leon Rosenberg
BEDCON 2013
@dvayanu
Why MoSKito
Problem finding and fixing

Availability

Hosting needs

Seasonal traffic

Service Level

RAM/CPU Requirements

Availability

Product pivots

New markets

Marketing campaigns

Cost per user

Provider change

Feature changes

ARCHitectural changes

BUGs
Seasonal traffic
Service Level
Availability
Problem finding and fixing
Marketing campaigns

BUGs
new features
architectural changes
new markets

Change Monitoring

Collect

Inspect

Analyze

Disaster recovery
RAM/CPU Requirements
Hosting needs
Cost per user
Provider change

SLA Monitoring

Capacity Monitoring
MoSKito is a multi-purpose, non-invasive, interval based monitoring system kit for collection, storage and instant analysis of application’s performance and behavior data.
Key Features

- Collect and Store.
- Inspect and Monitor.
- Analyze and Alert.
- Continuous production profiling without performance impacts with Journeys.
Interval based.

- The behavior of a system depends on hour, weekday, weather, holidays and good karma.
- Large amounts of collected data make monitoring nonsensitive to anomalies.
- Inspection of short intervals offers more understanding about system’s behavior.
Core Concepts


Statistic of a use case, i.e. method name, url, cumulated producer statistics

Value type, i.e. request count, avg duration, error count, cache hits, payments etc.

Container for different values for intervals
Core Sections

- Producers and Stats - gather monitoring data.
- Thresholds - monitor changes in critical sections of the application.
- Accumulators - builds trends and allow visual analysis.
- Journeys - make inner life of the application visible.
A picture is worth 1000 words...

... and live presentation is worth 1000 pictures.
Integration

- AOP / CDI
- Proxies
- WEB

Guide: https://confluence.opensource.anotheria.net/display/MSK/Integration+Guide
@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 {

    public class PaymentCounter {

        public void ec(){}
        /**
         * Electronic card payment (lastchrifteinzug in germany).
         */
        public void cc(){}
        /**
         * Credit card payment.
         */
        public void paypal(){}
    }
}

/**
 * Electronic card payment (lastchrifteinzug in germany).
 */
public void cc(){}
/**
 * Credit card payment.
 */
public void paypal(){}
}
<dependencies>
  <dependency>
    <groupId>net.anotheria</groupId>
    <artifactId>moskito-core</artifactId>
    <version>2.2.2</version>
  </dependency>
  <dependency>
    <groupId>net.anotheria</groupId>
    <artifactId>moskito-aop</artifactId>
    <version>2.2.2</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>2.2.2</version>
          </aspectLibrary>
        </aspectLibraries>
        <source>1.6</source>
        <target>1.6</target>
      </configuration>
      <executions>
        <execution>
          <goals>
            <goal>compile</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
public interface SimpleService{
    void doSomethingMethod();
}

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

SimpleService service = ProxyUtils.createServiceInstance(new SimpleServiceImpl(), "default", SimpleService.class);

SimpleService unmonitoredInstance = new SimpleServiceImpl();
MoskitoInvocationProxy proxy = new MoskitoInvocationProxy(
    unmonitoredInstance,
    new ServiceStatsCallHandler(),
    new ServiceStatsFactory(),
    "SimpleService",
    "service",
    "test-sub-system",
    SimpleService.class
);
SimpleService monitoredInstance = (SimpleService)proxy.createProxy();
<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>

<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>
web.xml

<!--  MOSKITO UI-->
<!--  Adding filter to moskito ui which redirects requests to /mui/* to moskito user interface -->
<filter>
    <filter-name>MoskitoUIFilter</filter-name>
    <filter-class>net.anotheria.moskito.webui.MoskitoUIFilter</filter-class>
    <init-param>
        <param-name>path</param-name>
        <param-value>/mui/</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>MoskitoUIFilter</filter-name>
    <url-pattern>/mui/*</url-pattern>
</filter-mapping>

<!-- / MOSKITO UI END -->

<!-- somewhere else --->

<listener>
    <listener-class>net.anotheria.moskito.webui.util.StartStopListener</listener-class>
</listener>
<listener>
    <listener-class>net.anotheria.moskito.web.session.SessionCountProducer</listener-class>
</listener>
<listener>
    <listener-class>org.anotheria.moskitodemo.threshold.presentation.listener.SetupThresholds</listener-class>
</listener>

Freitag, 5. April 13
What else

- MoSKito Central
- MoSKito Control
- MoSKito @ Barbecue
MoSKito Central

- Central server for snapshot storage.
- Attachable storages can store data into sql- or nosql- based databases, xml/json files etc.
- Runs in remote or embedded mode.
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. With MoSKito inspection we were able to detect 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>LoginAPI.isLogedIn() = true</th>
<th>same call over net repeated thrice</th>
<th>3</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegistrationAPI.isMeTest() = false</td>
<td></td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>PaymentAPI.isMeTest() = false</td>
<td></td>
<td>665</td>
<td>79</td>
</tr>
<tr>
<td>PaymentBusinessServiceDiMe 1.readActivePayments(372347) = [PaymentBO(id='159538', accountid='3723...</td>
<td>585</td>
<td>585</td>
<td></td>
</tr>
<tr>
<td>IASSiteDataService-2.getNavItem(33) = NavItem [33] name: Hilfe, title: Hilfe, externalLink: , p...</td>
<td>21</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>LoginAPI.isLogedIn() = true</td>
<td></td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>presentation.profile.handler.AboutMeHandler-C-57.process</td>
<td></td>
<td>767</td>
<td>92</td>
</tr>
<tr>
<td>LoginAPI.isLogedIn() = true</td>
<td></td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>LoginAPI.getLoggedUserId() = 372347</td>
<td></td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>RegistrationAPI.isMyEmailConfirmed() = true</td>
<td></td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>PaymentAPI.amiPremium() = true</td>
<td></td>
<td>603</td>
<td>40</td>
</tr>
<tr>
<td>PaymentBusinessServiceDiMe 1.readActivePayments(372347) = [PaymentBO(id='159538', accountid='3723...</td>
<td>562</td>
<td>562</td>
<td></td>
</tr>
<tr>
<td>LoginAPI.getMyLoginTime() = 1271890272896</td>
<td></td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td>LoginAPI.isLogedIn() = true</td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>LoginAPI.getMyPreviousLoginTime() = 1271651945362</td>
<td></td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>LoginAPI.isLogedIn() = true</td>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>IASWebDataService-2.getBox(167) = Box [167] name: Messagesbox: Cleanup Warning, content: &lt;p&gt;Konta...</td>
<td>30</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>RegistrationAPI.isMeTest() = false</td>
<td></td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>PaymentAPI.amiPremium() = true</td>
<td></td>
<td>668</td>
<td>62</td>
</tr>
<tr>
<td>PaymentBusinessServiceDiMe 1.readActivePayments(372347) = [PaymentBO(id='159538', accountid='3723...</td>
<td>605</td>
<td>605</td>
<td></td>
</tr>
<tr>
<td>IASWebDataService-2.getBox(1) = Box [1] name: Footer, content: &lt;p&gt;© {cal:currentYear} {text:brand...</td>
<td>13</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>
End of Data.

Thank you.

http://moskito.anotheria.net/


https://github.com/anotheria/moskito-examples

https://github.com/anotheria/moskito-jboss

http://search.maven.org/#search%7Cga%7C1%7Cmoskito
Chart for PhotoService REQ PhotoService AVG

Chart for Page WP AVG Page Home AVG
End of Data.

Thank you.

http://moskito.anotheria.net/


https://github.com/anotheria/moskito-examples

https://github.com/anotheria/moskito-jboss

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