Profiling Scout Applications

Useful tools and methods we use for profiling our Scout applications

Judith Gull

Scout User Group Meeting – 27.10.2014

Why is it so slow?

- ➔ Business Logic on Client or Server
- ➔ External Webservices
- ➔ Database Queries
- ➔ Network Connection

Will the system still be usable with many concurrent users?

- ➔ Can we measure and tune before productive use
- ➔ How do we need to setup application servers/database

Agenda

3 common ways we profile Scout applications at BSI

- ➔ Very simple profiling with TuningUtility
- ➔ Custom Scout Profiler on Service-Level
- ➔ Loadtests with Apache JMeter

Scout TuningUtility

TuningUtility

Simple Timing for Development

```java
TuningUtility.startTimer();
try {
   SERVICES.getService(IPersonProcessService.class).load(formData);
} finally {
      TuningUtility.stopTimer("load person formData");
}
```

```
#TUNING: load person formData took 125.671268ms
```

TuningUtility – Repeated Calls

```
try {
      for (int i = 0; i < 100; i++) {
        TuningUtility.startTimer();
        codeToMeasure();
        TuningUtility.stopTimer(«repeatCode", false, true);
      }
    } finally {
      TuningUtility.finishAll();
    }
```

Do not print yet

add to batch

stop batch

```
#TUNING: repeatCode[100] sum=1114.534945ms min=7.799033ms
avg=11.145349ms median=10.510715ms max=29.425593ms
[without 1 smallest and 1 largest:…
```

without smallest/largest 1%

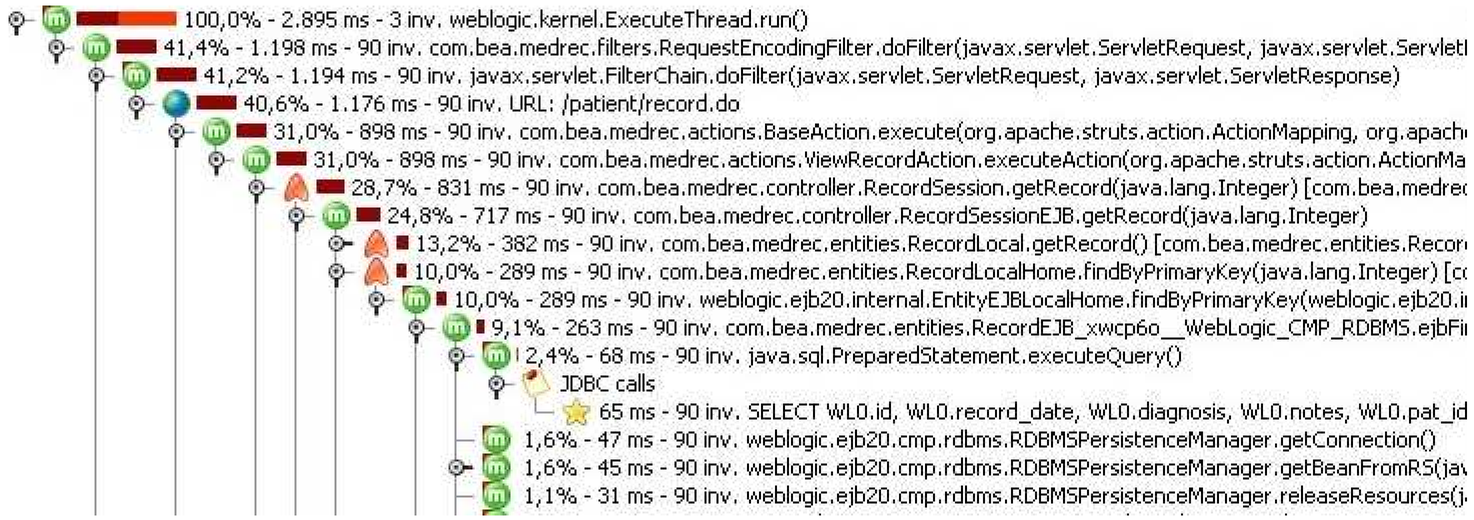
Summary – TuningUtility

- ➔ Very easy to use
- ➔ Do not use in production!

Custom Scout Profiler

Why a custom profiler?

➔ General purpose measure every method -> complicated



```
100,0% - 2.895 ms - 3 inv. weblogic.kernel.ExecuteThread.run()
  41,4% - 1.198 ms - 90 inv. com.bea.medrec.filters.RequestEncodingFilter.doFilter(javax.servlet.ServletRequest, javax.servlet.Servletl
    41,2% - 1.194 ms - 90 inv. javax.servlet.FilterChain.doFilter(javax.servlet.ServletRequest, javax.servlet.ServletResponse)
      40,6% - 1.176 ms - 90 inv. URL: /patient/record.do
        31,0% - 898 ms - 90 inv. com.bea.medrec.actions.BaseAction.execute(org.apache.struts.action.ActionMapping, org.apach
          31,0% - 898 ms - 90 inv. com.bea.medrec.actions.ViewRecordAction.executeAction(org.apache.struts.action.ActionMa
            28,7% - 831 ms - 90 inv. com.bea.medrec.controller.RecordSession.getRecord(java.lang.Integer) [com.bea.medre
              24,8% - 717 ms - 90 inv. com.bea.medrec.controller.RecordSessionEJB.getRecord(java.lang.Integer)
                13,2% - 382 ms - 90 inv. com.bea.medrec.entities.RecordLocal.getRecord() [com.bea.medrec.entities.Recor
                10,0% - 289 ms - 90 inv. com.bea.medrec.entities.RecordLocalHome.findByPrimaryKey(java.lang.Integer) [c
                  10,0% - 289 ms - 90 inv. weblogic.ejb20.internal.EntityEJBLocalHome.findByPrimaryKey(weblogic.ejb20.i
                    9,1% - 263 ms - 90 inv. com.bea.medrec.entities.RecordEJB_xwcp6o__WebLogic_CMP_RDBMS.ejbFi
                      2,4% - 68 ms - 90 inv. java.sql.PreparedStatement.executeQuery()
                        JDBC calls
                          65 ms - 90 inv. SELECT WL0.id, WL0.record_date, WL0.diagnosis, WL0.notes, WL0.pat_id
                      1,6% - 47 ms - 90 inv. weblogic.ejb20.cmp.rdbms.RDBMSPersistenceManager.getConnection()
                      1,6% - 45 ms - 90 inv. weblogic.ejb20.cmp.rdbms.RDBMSPersistenceManager.getBeanFromRS(ja
                      1,1% - 31 ms - 90 inv. weblogic.ejb20.cmp.rdbms.RDBMSPersistenceManager.releaseResources(j
```

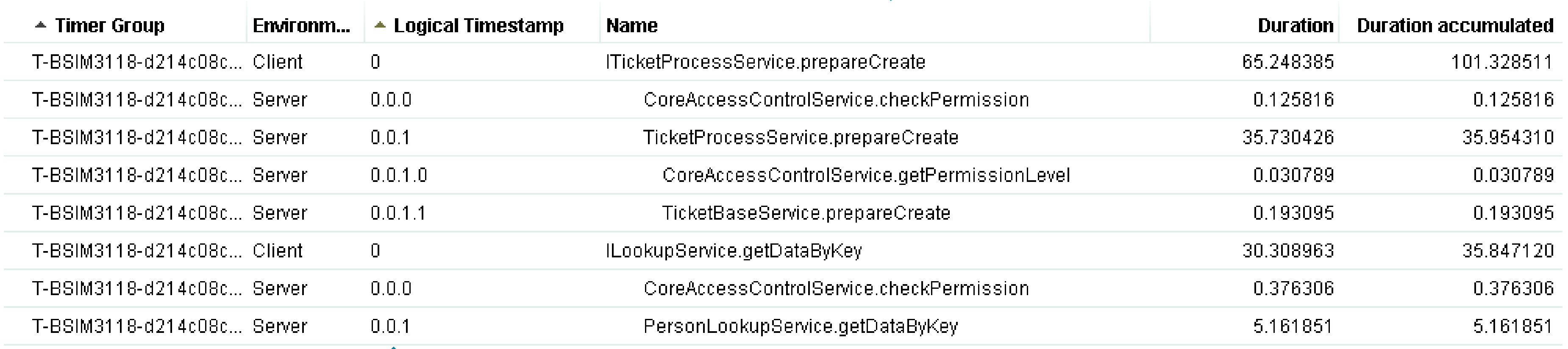
➔ Must be connected to JVM (often remotly with JMX)

➔ Difficult to profile both client and server

Profiling on Service Level

Displays server and client durations in single table

Only shows service methods (relevant for business)

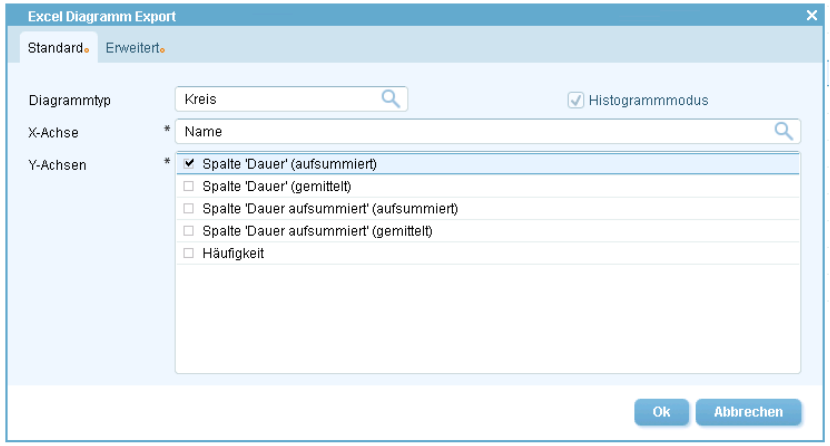


Timer Group	Environm...	Logical Timestamp	Name	Duration	Duration accumulated
T-BSIM3118-d214c08c...	Client	0	ITicketProcessService.prepareCreate	65.248385	101.328511
T-BSIM3118-d214c08c...	Server	0.0.0	CoreAccessControlService.checkPermission	0.125816	0.125816
T-BSIM3118-d214c08c...	Server	0.0.1	TicketProcessService.prepareCreate	35.730426	35.954310
T-BSIM3118-d214c08c...	Server	0.0.1.0	CoreAccessControlService.getPermissionLevel	0.030789	0.030789
T-BSIM3118-d214c08c...	Server	0.0.1.1	TicketBaseService.prepareCreate	0.193095	0.193095
T-BSIM3118-d214c08c...	Client	0	ILookupService.getDataByKey	30.308963	35.847120
T-BSIM3118-d214c08c...	Server	0.0.0	CoreAccessControlService.checkPermission	0.376306	0.376306
T-BSIM3118-d214c08c...	Server	0.0.1	PersonLookupService.getDataByKey	5.161851	5.161851

Measures durations in client and server
Works with different server/client time

-> much simpler than general purpose java profilers!

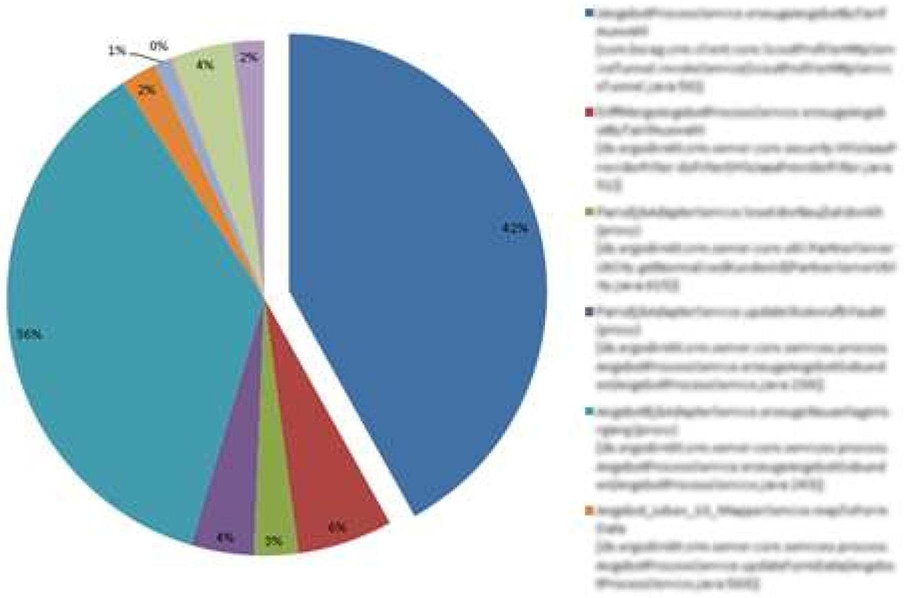
Diagram Export


Excel Diagramm Export

Standard | Erweitert

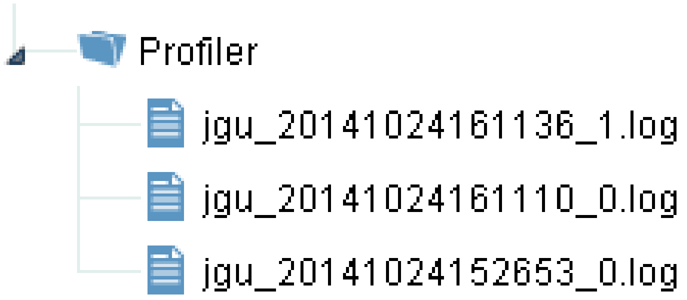
Diagrammtyp		Kreis
X-Achse	*	Name
Y-Achsen	*	[✓] Spalte 'Dauer' (aufsummiert)
		[ ] Spalte 'Dauer' (gemittelt)
		[ ] Spalte 'Dauer aufsummiert' (aufsummiert)
		[ ] Spalte 'Dauer aufsummiert' (gemittelt)
		[ ] Häufigkeit

[✓] Histogrammmodus



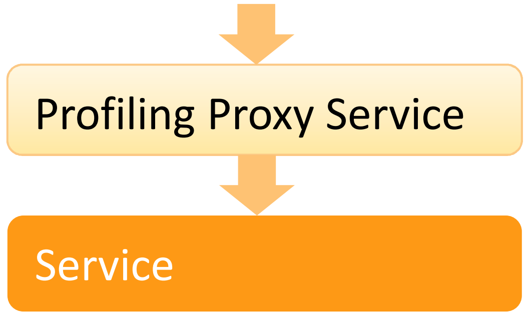
Profiling in Production

- ➔ stores profiling data as files on the server
- ➔ possible to enable per user session or global
- ➔ possible to run in production


```
Profiler
    jgu_20141024161136_1.log
    jgu_20141024161110_0.log
    jgu_20141024152653_0.log
```

Implementation

- On profiling start: Register profiling proxy service for every service
- Proxy Captures time and delegates to “real” service



Additional Profiling Tasks

➔ Add any additional profiling tasks where needed

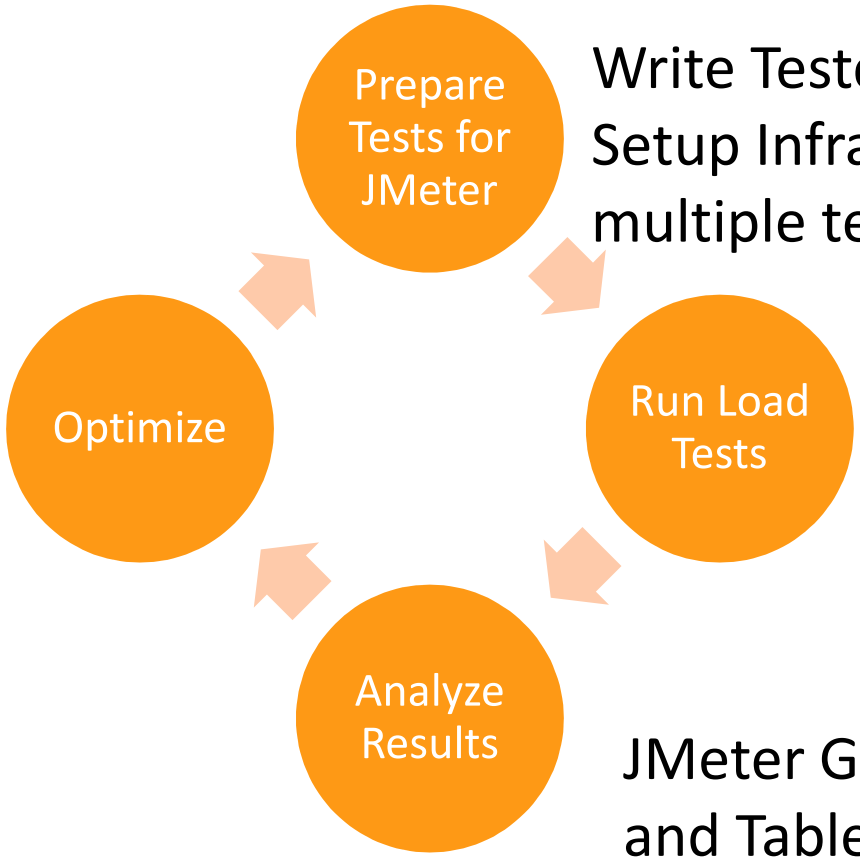
```
Profiler jobProfiler =
Profiler.beginTask(getJobName(), "JOB");
 try {
   execRun(jobRunDesc);
 }
 finally {
   jobProfiler.endTask();
 }
```

Summary - Custom Scout Profiler

- ➔ Easy to use with deployed applications
- ➔ Mostly sufficient to get an idea where the performance problems are
- ➔ Learnings: Measurements can be quite different (use multiple)

Load Testing with Apache JMeter

Load Testing Scout Applications



Write Testcases, Plan
Setup Infrastructure,
multiple test clients

Multiple phases,
Include manual
Testers

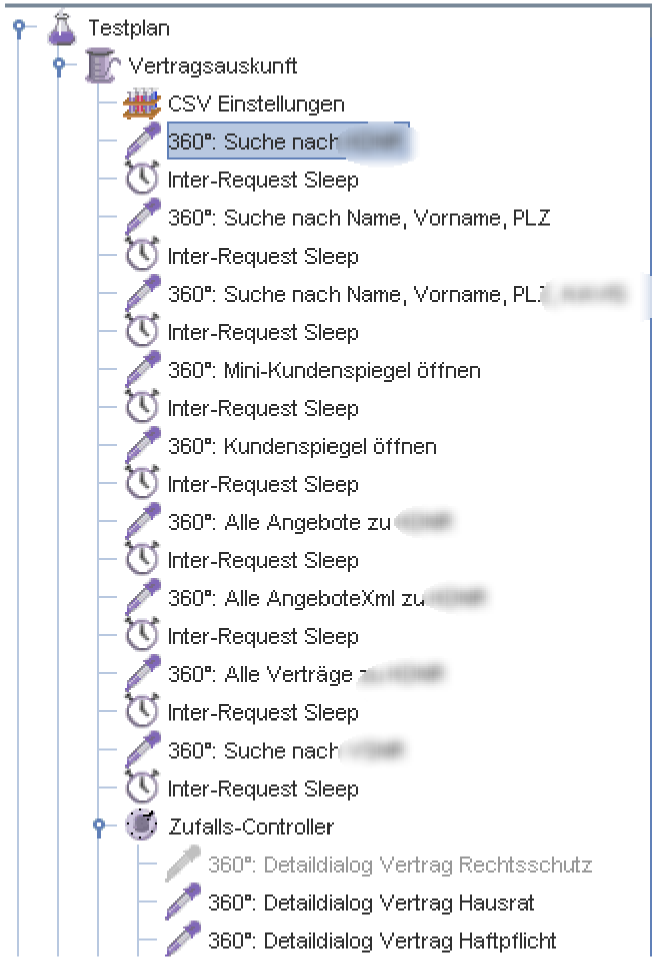
JMeter Graphs
and Tables

Prepare test client for JMeter

- ➔ Small Scout Extension for JMeterTests
  (creating session, formatting output)

- ➔ Prepare Testcases: Implement most common use cases

Configure Tests in JMeter



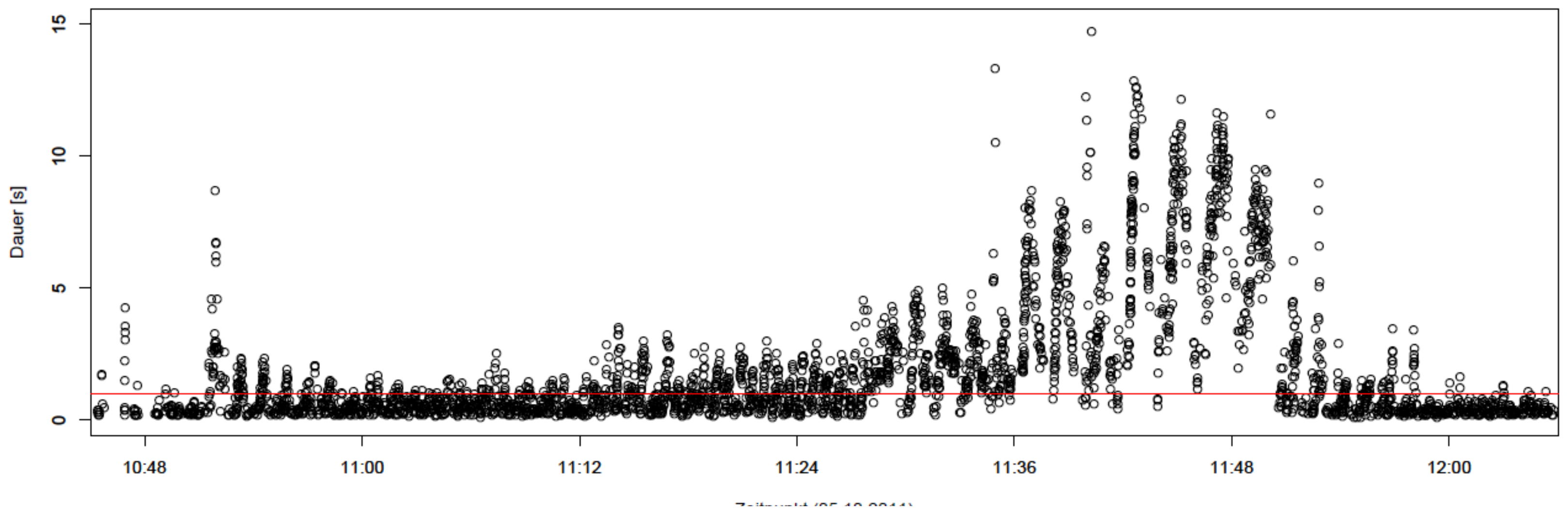
- Testplan
  - Vertragsauskunft
    - CSV Einstellungen
    - 360°: Suche nach [redacted]
    - Inter-Request Sleep
    - 360°: Suche nach Name, Vorname, PLZ
    - Inter-Request Sleep
    - 360°: Suche nach Name, Vorname, PLZ [redacted]
    - Inter-Request Sleep
    - 360°: Mini-Kundenspiegel öffnen
    - Inter-Request Sleep
    - 360°: Kundenspiegel öffnen
    - Inter-Request Sleep
    - 360°: Alle Angebote zu [redacted]
    - Inter-Request Sleep
    - 360°: Alle AngeboteXml zu [redacted]
    - Inter-Request Sleep
    - 360°: Alle Verträge [redacted]
    - Inter-Request Sleep
    - 360°: Suche nach [redacted]
    - Inter-Request Sleep
    - Zufalls-Controller
      - 360°: Detaildialog Vertrag Rechtsschutz
      - 360°: Detaildialog Vertrag Hausrat
      - 360°: Detaildialog Vertrag Haftpflicht

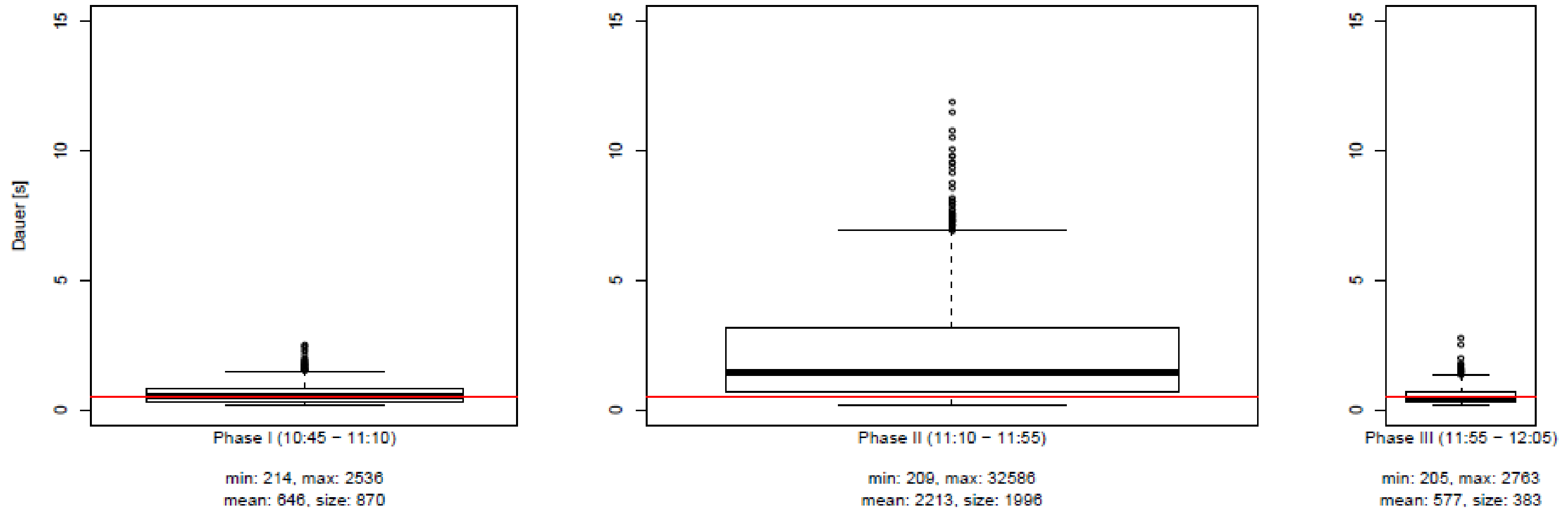
- ➔ Create Plans to simulate real users
- ➔ Configure random executions
- ➔ CSV Test Properties for multiple executions

Distribute Load Test Client

- ➔ Export special product file with Scout JMeter extension, project specific test cases, configuration
- ➔ Headless application (GUI rendering is not measured)

Simulation with increasing load over time





Summary – Load Tests

- Load tests are useful
  - there is usually room for improvement
  - problems are not always obvious
- Load tests are not free, require careful planning, significant amount of time to prepare
- Difficult to map “simulated users” to actual users

Questions