Open Source Performance Testing Tools

Alan Gordon
January 2010
Agenda

Open Source Tools – The Big Questions

- What tools are out there?

- When should we use them?

- What can we expect to achieve with them?

- How should we alter our strategy?
What Tools Are Out There?


- Allmon
- **Apache JMeter**
- benerator
- CLIF
- curl-loader
- Database Opensource Test Suite
- DBMonster
- Deluge
- Dieseltest
- Faban
- FunkLoad
- FWPTT
- **Grinder**
- Hammerhead 2 - Web Testing
- Hammerora
- httpperf
- http_load
- IxoraRMS
- JChav
- JCrawler
- Lobo, Continuous Tuning
- MessAdmin
- NTime
- **OpenSTA**
- OpenWebLoad
- p-unit
- PandoraFMS
- Pylot
- Raw Load Tester
- Seagull
- Siege
- Sipp
- **SLAMD**
- Soap-Stone
- stress_driver
- TestMaker
- TPTEST
- Tsung
- Valgrind
- Web Application Load Simulator
- Web Polygraph
- **WebLOAD**
A Few Notable Tools

Apache Jmeter
- Java project
- HTTP, FTP, LDAP, TCP, web services

The Grinder
- “A Java Load Testing Framework” – Java / Jython – entire Java/Python library available
- Eclipse to debug scripts

OpenSTA
- Originally written by Cyrano
- Web only, Windows only

WebLOAD
- Open source version of a commercial tool
- Development sponsored by the commercial tool vendor – Radview

SLAMD
- Java-based
- Developed at Sun Microsystems, 2002-2004
- LDAP, SMTP, POP
Example: The Grinder

Financial Markets System

- We used the Grinder to simulate various server-to-server calls within the application stack
  - Simulated WebSphere (WAS) calls to the data layer servers
  - Simulated other calls between IBM DataPower and 3rd party services
  - SOAP / HTTP requests to a web service via a SOAP API

- Our findings:
  - Grinder was easy to set up & use – given that we had programming skills in the team
  - Rewrote the GrinderAnalyzer engine for analysing the results
  - Slight problems with scalability – perhaps due to our own script complexity!
  - Slight problems with validity of request – had to work hard to simulate WebSphere calls precisely

- Overall the Grinder worked really well
- By and large “does what it says on the tin” – gives you a framework to run your scripts
- A lot of value has been gained for a project lasting 9-12 months
Use Them When They Fit Your Organisation/Project Criteria

- Technology / protocol – outside Web area, options may be limited
- Size of test (scalability)
- Complexity of test – the simpler the technology, the more likely to be suitable for open source
- Skills & experience of your team – programming, test design, results analysis
- Long or short-term goals – for a short-term project, need a “quick start” tool
- Can use them to complement existing commercial tools – e.g. to simulate background load

Don’t Use Them Just Because You’re Desperate To Save Cash

- Investment required in team skills
- May take a while to get off the ground – learning & development needed
- Maintenance costs could be higher

- It is worth trying a few tools to ensure you choose the right one
- Better to choose a tool that has a reasonable user base than a “one man project”
Open Source Performance Testing Tools

What Can We Expect To Achieve With Them?

- With enough time & effort, many things will be achievable with an open source tool
- However, knowledge can be difficult to find
  - No help manual, no dedicated support – you get what you pay for!
  - “No one is under any obligation to fix your problem. If all else fails, you have the source.”
  - Sources of information on web may be inconsistent
- That said, user base of the major tools is growing – there is even a book about Jmeter!

- I recommend that you invest in creating harnesses & frameworks to make your tests easier
- We created a framework for testing web services with OpenSTA
- Data driven & keyword driven – like the classic GUI automation frameworks
- Effort is reduced by up to 95% - once framework is developed

- With script-driven tools such as the Grinder, your output will be as good as your programmers – but remember to make sure they know how to test too!
Effort Estimation

- The table below shows the estimated effort increase for a “typical” open source tool against a “typical” commercial tool (when implementing it on a new project).
- Of course, there aren’t any such things, and the comparison is dependent on many factors.
- However, if you wish to use an estimate for planning, you could do worse than use this...

<table>
<thead>
<tr>
<th>Test Phase</th>
<th>% Change (compared to a typical commercial tool)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td>200 – 400% increase</td>
</tr>
<tr>
<td>Execution</td>
<td>0 – 50% increase</td>
</tr>
<tr>
<td>Analysis</td>
<td>200 – 500% increase</td>
</tr>
</tbody>
</table>

Estimation Factors – Summary of some considerations mentioned in previous slides:

- Existing skills in team & propensity for learning (especially programming)
- Technology to be tested and whether it is supported by the tool, and how good that support really is
- Whether any previous frameworks are available or whether you are starting from scratch
- The complexity of your test requirements (business processes, output required)
- Amount of maintenance required for future test cycles – is this a one-off or do we need to maintain it?
Summary

Greater Success Is Likely If You...

- Allow months not days to get up to speed
- Depending on your choice of tool – make sure you have a programmer on the team
- Document as you go – create your own help file
- Aim strategically – reuse throughout your organisation
- Aim to complement commercial tools with open source, not completely replace them