CLIF is a Load Injection Framework (v 1.1)

a guided tour of ObjectWeb's generic load test platform

Bruno Dillenseger
France Telecom, Research & Development division
bruno.dillenseger@francetelecom.com

Fifth Annual ObjectWeb Conference, January 31 - February 2nd, 2006
CNIT, Paris La Défense, FRANCE
Big picture of (distributed) load testing

**Load injectors**:
- send requests, wait for replies, measure response times
- according to a given scenario
- for example, emulating the load of a number of real users

**Probes** measure usage of arbitrary computing resources

Execution control and monitoring of load injectors and resource probes.
CLIF is a Load Injection Framework...

CLIF is dedicated to:

- high-level (distributed) load injection
- performance measurement (response time, error occurrence, etc.)
- resources consumption measurement (probes)

... open and flexible...

- the framework is independent from:
  - the scenario definition mode,
  - system under test (invocation protocols, etc.)
  - the type of observed resources
- the architecture uses Fractal Component Model
- CLIF is open source software (http://clif.objectweb.org/)

... easy to use

- centralized test deployment, control and monitoring of distributed tests
- support for a variety of user skills and needs
- 100% Java (1.4) + integration with Eclipse
CLIF distributed architecture

- **deploy and control injectors and probes**
- **invoke target system**
- **system under test**
- **resource probe**
- **collect all data**
- **observed data:**
  - performance (response time, error, result...)
  - resources usage (CPU, memory...)
  - lifecycle events (test deploy, start, alarms...),
- **data store**

**Post mortem analysis**

**Monitoring (statistic data)**

**Supervision console**

(interactive or batch mode)
Guided tour of CLIF version 1.1 – Bruno Dillenseger, France Telecom, February 2006
PART 2

Tools for defining load test scenarios
For Fractal-aware Java programmers: implement your own injection component

An injector must conform to the following component type:

- **injector activity control** (init, start, stop, suspend, etc.)
- **filtering and statistical monitoring of measurements**
- **local buffering and final collection of measurements**

- see org.objectweb.clif.server.api.BladeType.fractal
- see org.objectweb.clif.server.lib.Blade.fractal ("common blade architecture" helper)
MTScenario is a generic (abstract) injector component for handling:

- the creation of a given number of sessions (threads)
- and their life-cycle (init, start/stop, suspend/resume...)
- during a given test duration
- with a given ramp-up time

Used through sub-classing

- the programmer just has to define the “actions” to be performed by each session
- the Webtest example features a simple Web test

Takes a list of URLs as parameter (GET/POST). Based on Jakarta HttpClient.
"ISAC is a Scenario Architecture for Clif"

- provides a generic, formal and user-friendly way to define an injector load scenario
- a scenario combines
  - the definition of sequential behaviors (typically representing the SUT users) with control constructs: delay, if-then-else, loop, probabilistic choice, preemption
  - with a load profile specification for each behavior

number of active instances = f(time)
ISAC is generic and extensible through a plug-in feature

Each behavior defines a generic logic, making use of imported plug-ins to implement:

- delays (think time: constant, random with arbitrary distribution...)
- actions (requests on the system under test)
- conditions (used by if, while, and preemption)
- controls (for specific administration purpose of ISAC plug-ins)
- external data provisioning (to parameterize scenarios with external data sets)
ISAC execution engines

First generation engines (for history...)

- 1 behavior instance (i.e. 1 virtual user)  1 execution thread
- practical maximum limit regarding computing resources: thousands of virtual users

Second generation engine (as of latest version)

- all behavior instances  fixed pool of threads (cooperative scheduler)
- new, advanced features
  - the size of the thread pool may be changed on demand during scenario execution
  - the population of any behavior may be changed on demand during scenario execution
    (then, the specified load profile for this behavior, if any, is disabled)
  - alarms are raised when think times are longer than specified (with tunable tolerance)
  - support for external data sets (data provider plug-ins)

- order of magnitude: million of virtual users, million of requests/second

The actual limits depend on the behaviors' think times and the consumption of computing resources by the imported ISAC plug-ins for each behavior instance
PART 3

Using CLIF
Overview of load testing process

1. run a CLIF console and start the “CLIF registry” to be able to register “CLIF servers” where you plan to deploy probes and/or load injectors
2. run the CLIF servers on every computer you want to use
3. define your test plan
   → list of load injectors and/or probes with their parameters and the target CLIF server
4. deploy the test plan
5. perform one or several test executions
   → initialize, start, monitor execution (stop, suspend, resume if needed)
   → collect results when execution is done (aborted, stopped or complete)
   → analyze results
### Test Plan Editor

#### Injectors and probes

All injectors and probes in the test plan

<table>
<thead>
<tr>
<th>Id</th>
<th>Server</th>
<th>Role</th>
<th>Class</th>
<th>Arguments</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>local host</td>
<td>injector</td>
<td>IsacRunner</td>
<td>j2eedo.xls</td>
<td></td>
</tr>
</tbody>
</table>

#### Properties

Manage injector and probe properties

<table>
<thead>
<tr>
<th>Id</th>
<th>Server</th>
<th>Role</th>
<th>Class</th>
<th>Arguments</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>local host</td>
<td>injector</td>
<td>IsacRunner</td>
<td>j2eedo.xls</td>
<td></td>
</tr>
</tbody>
</table>
Test plan deployment and initialization
Test plan execution and monitoring

Test plan execution and monitoring

Test Commands

Injectors and probes

All injectors and probes in the test plan

<table>
<thead>
<tr>
<th>Id</th>
<th>Server</th>
<th>Role</th>
<th>Class</th>
<th>Arguments</th>
<th>Comm</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>local host</td>
<td>injector</td>
<td>IsacRunner</td>
<td>helloworld.xml</td>
<td></td>
<td>initialized</td>
</tr>
<tr>
<td>1</td>
<td>g-necmi5-199</td>
<td>injector</td>
<td>Autotest</td>
<td>100 10 10 100</td>
<td></td>
<td>initialized</td>
</tr>
</tbody>
</table>

Test plan execution and monitoring

Test 1 - 20 septembre 2005 3:53:26

CPU injector

Display Collect Blade Time

%CPU

Drawing timeframe: 100 sec Polling period: 1 sec Refresh Reset

Guided tour of CLIF version 1.1 – Bruno Dillenseger, France Telecom, February 2006

www.objectweb.org
Using the command-line tools

- **deploy** - Deploys a new test plan (probes and injectors) as defined by a given test plan file.
  
ant -Dtestplan.name=name -Dtestplan.file=file.ctp deploy

- **init** - Initializes all probes and injectors from a given test plan, or just a sub-set of them if mentioned.
  
ant -Dtestplan.name=name -Dtestrun.id=testId [-Dblades.id=id1:id2:....:idn] init

- **start** - Starts probes and injectors. They must be initialized before.
  
ant -Dtestplan.name=name [-Dblades.id=id1:id2:....:idn] start

- **suspend, resume, stop** - Respectively suspends, resumes or stops running or suspended blades.
  
ant -Dtestplan.name=name [-Dblades.id=id1:id2:....:idn] suspend/resume/stop

- **join** - Waits until the execution of probes and injectors is terminated (aborted, stopped or completed).
  
ant -Dtestplan.name=name [-Dblades.id=id1:id2:....:idn] join

- **collect** - Collects results of completed or stopped probes and injectors.
  
ant -Dtestplan.name=name [-Dblades.id=id1:id2:....:idn] collect

- **run** - Short-cut for init, start, join, collect on probes and injectors.
  
ant -Dtestplan.name=name -Dtestrun.id=testId [-Dblades.id=id1:....:idn] run

- **params** - Lists all parameters that can be changed during execution for a given probe or injector.
  
ant -Dtestplan.name=name -Dblade.id=id params

- **change** - Changes a parameter value for a given probe or injector.
  
ant -Dtestplan.name=name -Dblade.id=id -Dparam.name=param -Dparam.value=value change
Available probes

- System probes are available for Linux and Windows, based on LeWYS
  - cpu (processor usage),
  - memory (RAM and swap usage),
  - jvm (JVM memory usage)
- simple and low footprint design
- Native code for Windows, pure Java for Linux (using /proc pseudo file system)
- 2 parameters: polling period in ms, execution duration in seconds.

- Simple framework to define any kind of probe

- Currently, CLIF's probe architecture is about to be re-engineered, considering other technologies wherever relevant:
  - Eclipse TPTP, JMX, Fractal JMX...
  - Good advices, contributions, discussions are welcome
Test results

- Monitoring gives a lot of information at runtime
  - min/max/average response time
  - requests and errors throughput, error rate
  - specific measurements from probes
  - low footprint design

- Post mortem analysis
  - all measurements from injectors and probes are available as CSV-formatted files
  - support tools: work in progress but nothing released yet
  - it is also possible to use CLIF just for massive, background load generation without storing results, and use other dedicated tools for accurate performance analysis.
Conclusion
Guided tour of CLIF version 1.1 – Bruno Dillenseger, France Telecom, February 2006

Current status

- CLIF is internally used by France Telecom's R&D division for load testing and performance evaluation
  - middleware (application servers, persistence), DNS, DHCP, specific telco platform
- Development plans for this year:
  - Eclipse Wizards to create probes and ISAC plug-ins
  - improved probe system and more probes
  - result analysis tools
  - more ISAC plug-ins (including SIP support)
  - capture of HTTP sessions and replay with ISAC HTTP Injector
- Research plans for next years: autonomic load testing and performance evaluation
Questions & Answers

http://clif.objectweb.org/