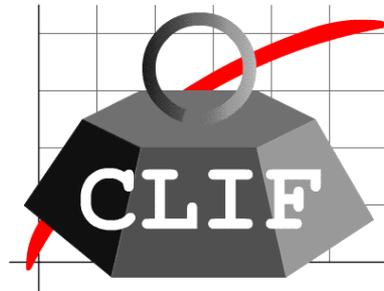


CLIF is a Load Injection Framework (v 1.1)



*a guided tour of ObjectWeb's
generic load test platform*

Bruno Dillenseger

France Telecom, Research & Development division

bruno.dillenseger@francetelecom.com



france telecom



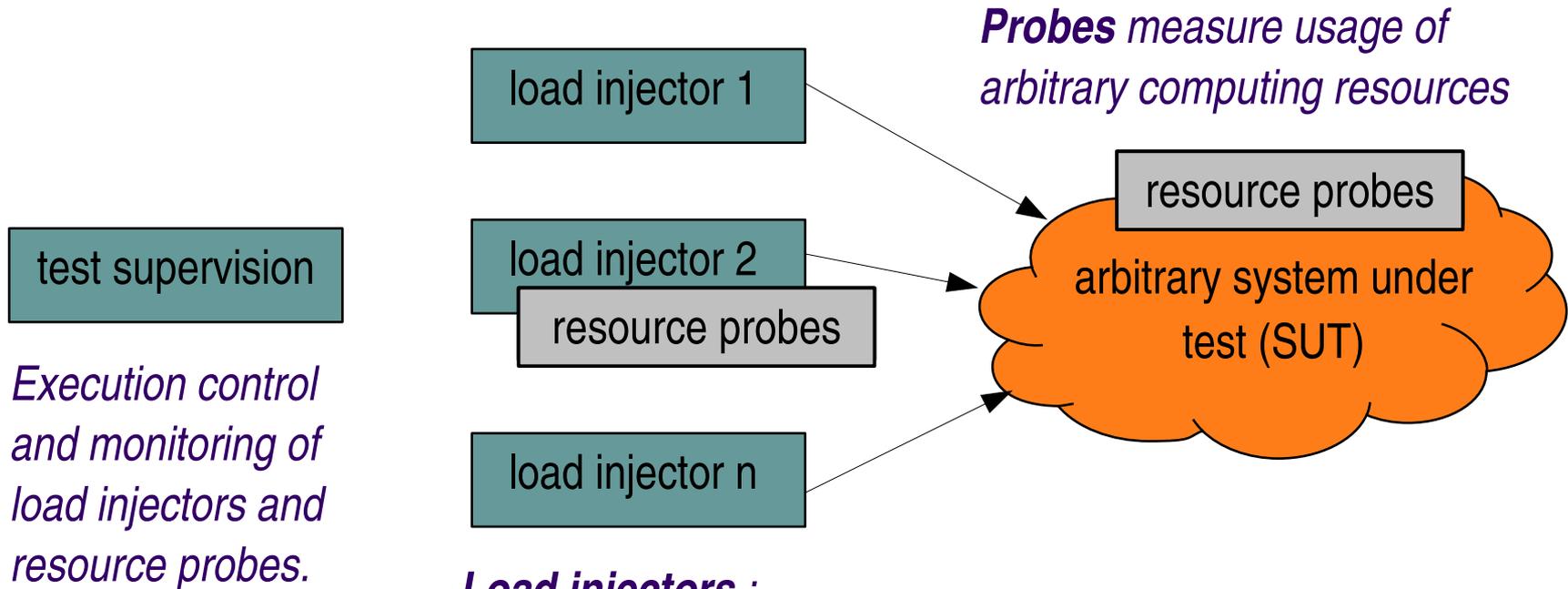
Fifth Annual ObjectWeb Conference, January 31 - February 2nd, 2006
CNIT, Paris La Défense, FRANCE

PART 1

Overview of Concepts, Principles and Architecture



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*

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)

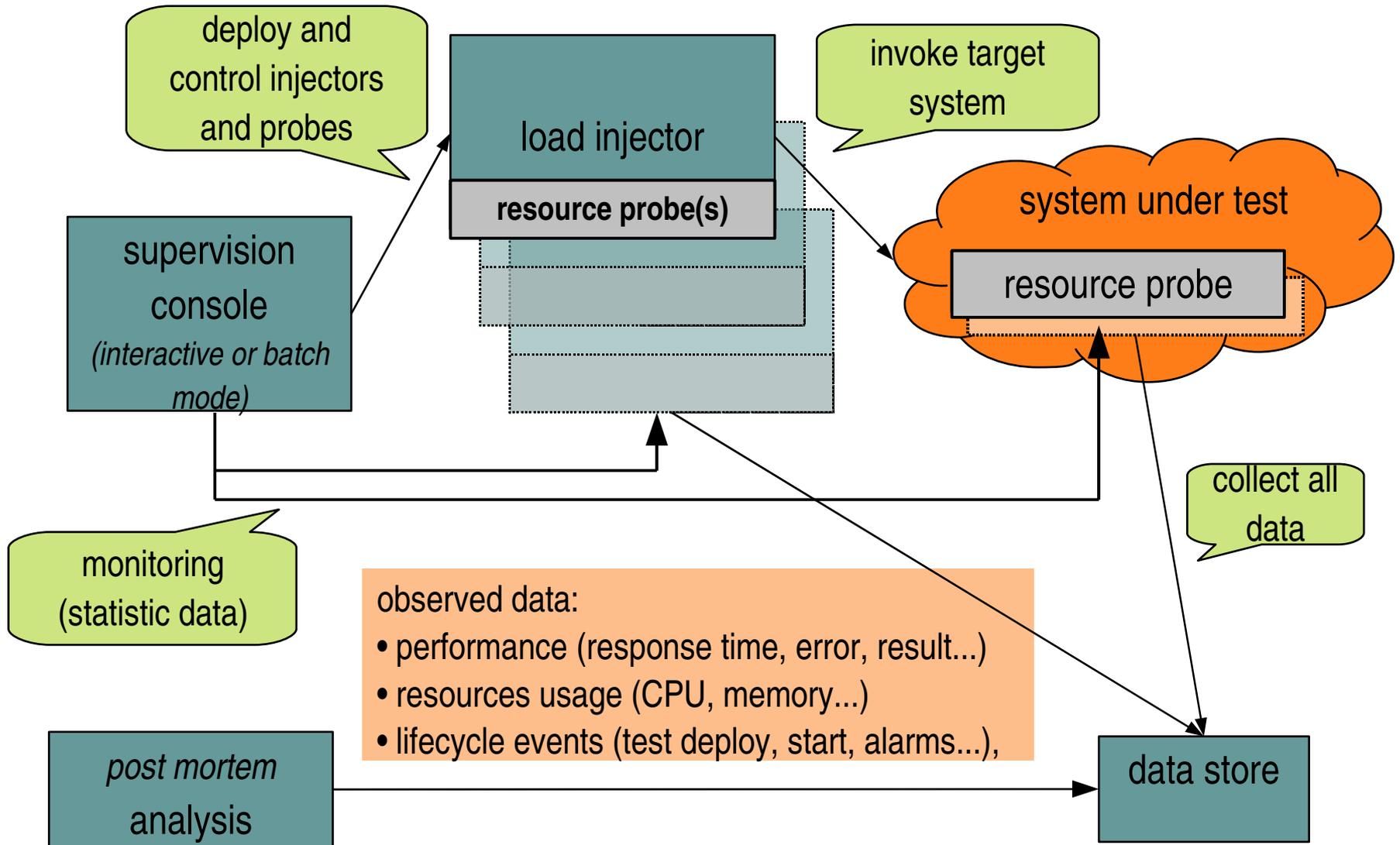
... open and flexible...

- 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



CLIF is a Load Injection Framework

File Test plan Tools ?

probeTP.prop

Add a blade Remove a blade Remove all blades

Blade id	Server	Role	Blade class	Blade argument	Comment	State
3	g-smithp2.rd.francetelecom.fr	probe	system	1000 150	sonde systeme	completed
2	g-smithp2.rd.francetelecom.fr	probe	cpu	1000 150	sonde CPU	completed
0	g-smithp2.rd.francetelecom.fr	injector	Autotest	100 100 50 100	injecteur	completed
1	g-smithp2.rd.francetelecom.fr	probe	memory	1000 150	sonde memoire	completed

memory cpu system injector

Display	Collect	Blade
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0

action throughput (actions/...)

Drawing timeframe: 507 sec. Polling period: 1 sec. Set/Draw Stop Reset

stopped - elapsed time 0:2:40

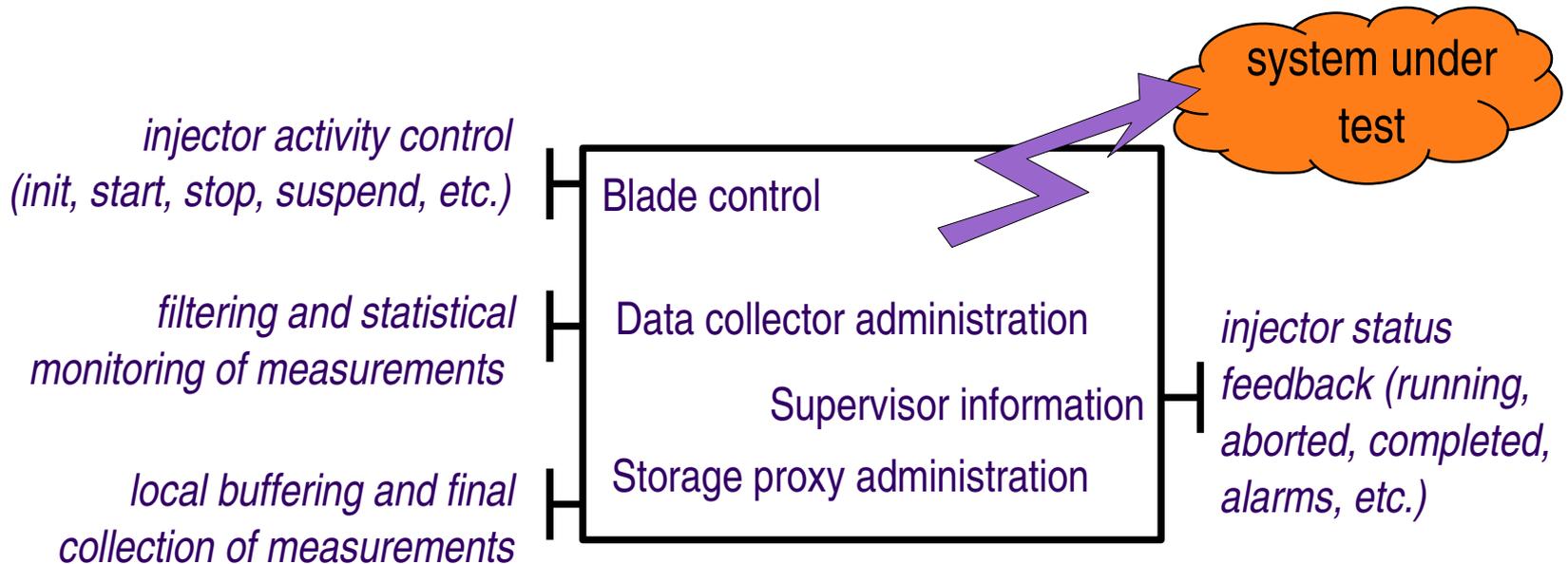
PART 2

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:

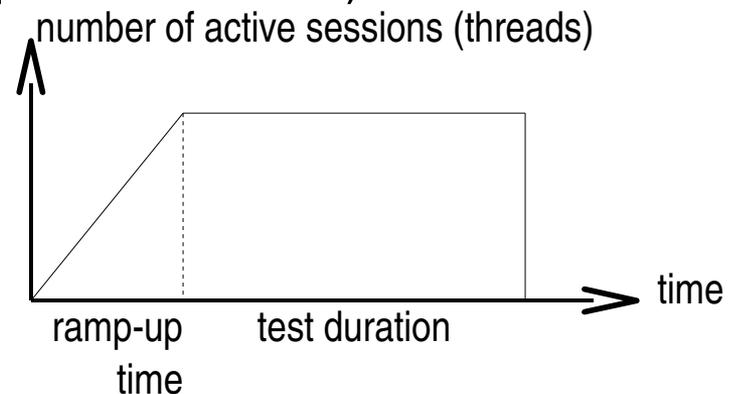


- see `org.objectweb.clif.server.api.BladeType.fractal`
- see `org.objectweb.clif.server.lib.Blade.fractal` (“common blade architecture” helper)

For Java programmers and simple scenarios: MTScenario class

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



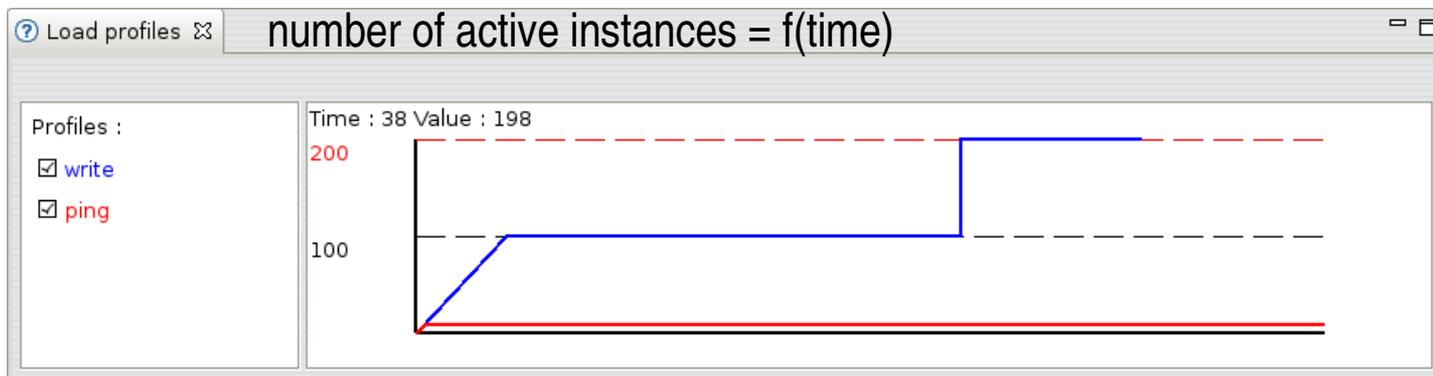
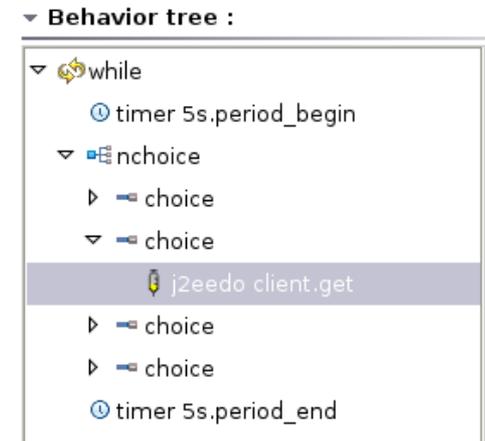
Used through sub-classing

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

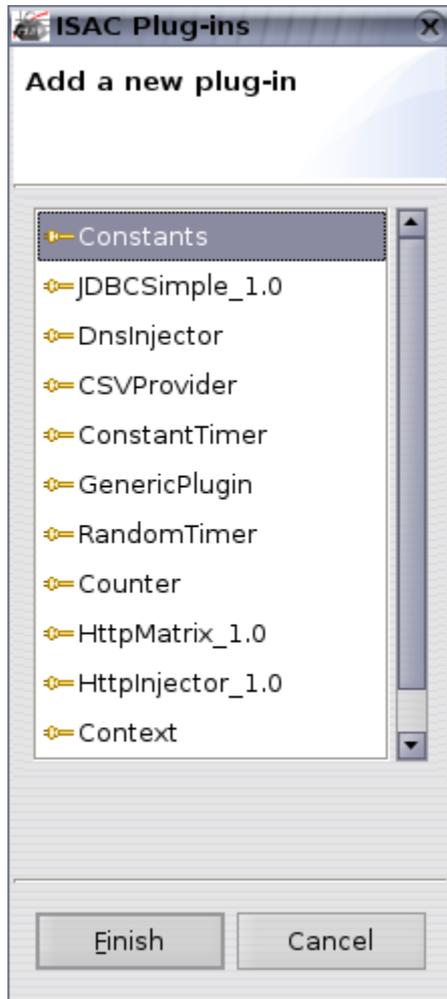
Takes a list of URLs as parameter (GET/POST). Based on Jakarta HttpClient.

"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



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)

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

The screenshot shows the ClifConsole application window. The title bar reads "ClifConsole" and the menu bar includes "File", "Edit", "Search", "CLIF", "Window", and "Help". The interface is divided into several panes:

- Navigator:** A tree view on the left showing a project structure: "my_test_campaign" containing ".project", "helloworld.xis", "j2eedo.ctp", "j2eedo.xis", and "scenario1.xis".
- Edition page for behavior description:** The main workspace. It features:
 - A "Behavior id" field containing "write".
 - Buttons for "New", "Duplicate", and "Remove".
 - A "Load profile" section with a dropdown menu.
 - A "Behavior tree" showing a hierarchical structure: "while" (containing "timer 5s.period_begin", "nchoice", and "timer 5s.period_end"), "nchoice" (containing "choice" and "choice"), and "choice" (containing "j2eedo client.get", "choice", and "choice").
 - Buttons for "Add", "Remove", "Clear", "Up", and "Down" next to the behavior tree.
- Properties:** A panel on the right titled "Manage plug-ins properties" showing:
 - A "sample" field with the value "HttpInjector_1.0.get".
 - An "id" dropdown menu set to "j2eedo client".
 - A "Configure the Sample" section with a "URI (required)" field containing "http://g-dellrack1:9000/doActio".
 - A "Set automatic redirection (optional)" section with an "enabled" checkbox.
 - An "Enter Parameters (scheme : 'name=value')(opt)" section with "Add field" and "Remove field" buttons. Below are three parameter fields: "field 0 : action=newProj", "field 1 : container=false", and "field 2 : withTrans=false".
- Design Source Import Behavior ping Behavior write:** A tabbed interface at the bottom of the main workspace.
- Load profiles:** A panel at the bottom left showing a list of profiles: "write" and "ping", both with checked checkboxes.
- Graph:** A line graph on the bottom right showing "Time : 38 Value : 198". The y-axis ranges from 0 to 200. A blue line starts at (0,0), rises to (10,100), stays flat until (30,100), then jumps to (30,198) and stays flat. A red dashed horizontal line is at y=200.

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

ClifConsole

File Edit Search CLIF Window Help

Navigator

my_test_campaign

- .project
- j2eedo.ctp
- j2eedo.xis

ClifTreeView

- local host
 - jvm 3
 - injector 0
- g-ong
 - jvm 2
 - memory 1

Test Plan Editor

Injectors and probes

All injectors and probes in the test plan

jvm injector memory

Id	Server	Role	Class	Arguments	Comment
0	local host	injector	IsacRunner	j2eedo.xis	

Add
Remove
Remove All

Properties

Manage injector and probe properties

Id* : 0

Server* : local host Refresh

Role* : injector

Class* : IsacRunner

Arguments : j2eedo.xis

Comment :

Edit Test

The screenshot shows the ClifConsole application window. The main area displays the 'Test Commands' section, specifically 'Injectors and probes'. A table lists the injectors and probes in the test plan:

Id	Server	Role	Class	Arguments	Comment	State
<input checked="" type="checkbox"/> 0	local host	injector	IsacRunner	j2eedo.xis		undeployed

Below the table, there are several buttons: Deploy, Initialize, Start, Suspend, Stop, Collect, and Parameters. The 'Global state' is indicated as 'undeployed' in red text.

The screenshot displays the CLIF Console interface, which is used for managing and monitoring test plans. The interface is divided into several sections:

- Navigator:** Shows the project structure, including 'Proj1' and 'new_test_plan.ctp'.
- Test Commands:** Displays the 'Injectors and probes' section, listing all injectors and probes in the test plan. The table below shows the details of the injectors:

Id	Server	Role	Class	Arguments	Comm	State
0	local host	injector	IsacRunner	helloworld.xml		initialized
1	g-necmi5-199	injector	Autotest	100 10 10 100		initialized

Buttons for 'Select All', 'Deselect All', and 'Global state: initialized' are visible on the right. Below the table are buttons for 'Deploy', 'Initialize', 'Start', 'Suspend', 'Stop', 'Collect', and 'Parameters'.

Monitor: Shows the test execution progress. The test is titled 'Test1 - 20 septembre 2005 3:53:29'. The monitor displays a graph of CPU usage over time. The graph shows a blue line representing CPU usage, which fluctuates between approximately 20% and 100%. The x-axis represents time in seconds (0 to 80), and the y-axis represents %CPU (0 to 100). The current time is 27 seconds, and the value is 8. The graph is titled 'Time : 27 Value : 8'.

Below the graph, there are controls for 'Display', 'Collect', 'Blade', and 'Time'. The 'Display' and 'Collect' checkboxes are checked. The 'Blade' value is 2, and the 'Time' value is 0. The graph is labeled '%CPU'.

At the bottom of the monitor section, there are controls for 'Drawing timeFrame : 100 sec. Polling Period : 1 sec.' and buttons for 'Refresh' and 'Reset'.

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

- **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

→ **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



- **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**

Thank you for your attention

Questions & Answers



<http://clif.objectweb.org/>

