Using CLIF load injection framework for performance evaluation - a use case with Speedo JDO implementation.

Contact :

Bruno Dillenseger, Sébastien Chassande-Barrioz, Yoann Bersihand
bruno.dillenseger@francetelecom.com,sebastien.chassandebarrioz@francetelecom.com,
yoann.bersihand@francetelecom.com
France Télécom, R&D division, MAPS/AMS lab
28, chemin du Vieux Chêne
BP 98
38243 MEYLAN cedex
FRANCE

Context and technological elements (CLIF, ISAC and Speedo)

About benchmarking and associated tools

Today's middleware offer is very large, and benchmarking has become a necessity to assess and compare similar implementations (such as, in our use case, implementations of JDO specifications or other similar object-oriented Java-based persistence support), as well as to evaluate the overall performance of the applications based on these middlewares. As a matter of fact, while hardware performance was tremendously increasing over the past decade, software has also become considerably performance consuming for the sake of advanced capabilities and complexity management.

The issue is that benchmarking itself is a complex task, requiring computing power to generate load (e.g. by emulating user traffic), support tools, experience in tuning and analyzing measurements, as well as a lot of time to deploy, run and . If a lot of tools exist, whatever commercial or open source, then the question that raises is: why are there so many of them? Basically because there is a great variety of needs:

- variety of target systems/protocols, sometimes standard (HTTP, LDAP, CORBA, NFS, etc.), sometimes specific/proprietary
- variety of users skills (programmers, software integrators and architects, testing professionals)
- variety of seek information/benchmarks

Very few platforms are able to handle this variety, and if we concentrate on open source platforms, the number is quite small. Moreover, the genericness of such platforms is likely to introduce unnecessary complexity according to the needs in some cases. As a result, for the sake of cheapness and simplicity, many programmers prefer to develop their specific, home-made, benchmarking platform.

CLIF, the Load Injection Framework

The aim of ObjectWeb community's CLIF project (http://clif.objectweb.org/) is to provide a Java-based, open source, generic infrastructure to generate load on any kind of systems, and gather performance measurements (requests response times, computing resources usage, etc.). It consists of a component framework handling the following concepts:

- an arbitrary *System Under Test* (SUT), composed of one or several execution nodes ;
- *load injector* components, to generate requests on the SUT and measure the corresponding response times or detect erroneous situations (e.g. reply time-out, reply specifying an exception, bad reply) ;
- *probe* components, to measure the consumption of arbitrary resources (typically dispatched among the SUT nodes, but also on the injection nodes to check they are not saturating) ;
- a central *supervision console* component, to handle distributed probes and load injectors (may be an interactive, GUI or text-based console, or a set of batch execution control commands) ;
- a storage component, to store the measurements produced by the probes and load injectors
- analyzer components, to browse, analyze and build test reports.

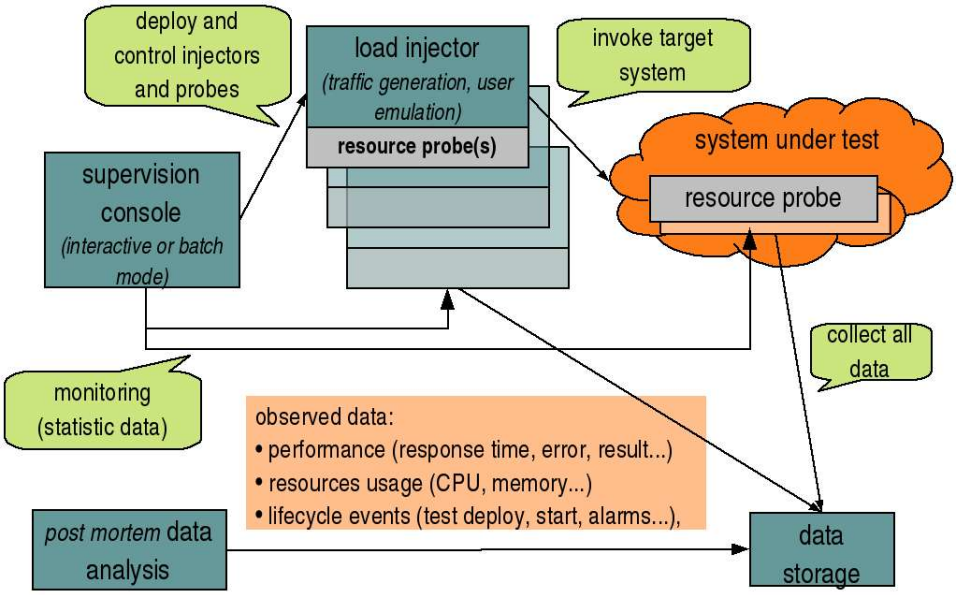

figure 1. CLIF's component-based architecture

ISAC, a Scenario Architecture for CLIF

As previously mentioned, CLIF is independent from the way traffic is specified and generated by load injectors. However, several load injection components are provided as independent helpers for defining traffic scenarios. Those helpers are independent from the SUT invocation protocols, and manage a population of execution threads performing requests on the SUT in order to emulate the traffic of a given number of parallel client activities.

The most advanced helper is ISAC. ISAC provides a formal support to describe elementary behaviors (e.g. typical behaviors of virtual SUT users). Those behaviors are basically sequences of requests on the SUT and delays (think times), including conditional loops, branches and preemption. Those scenarios may be written in XML, or generated either from a GUI (see figure 2) or by capturing real sessions through a proxy mechanism. A scenario combines a definition of those behaviors with a load profile specifying the population (i.e. number of instances) of each behavior as a function of time.

ISAC uses a plug-in pattern to resolve its genericness and actually invoke the SUT using the appropriate protocols, as well as to implement specific conditions, specific timers, and external data sources.

Speedo, an implementation of Java Data Objects specification

Speedo is a JDO personality of the ObjectWeb Open Source Persistence Framework (http://speedo.objectweb.org). It allows persistent application objects to be mapped to any type of data stores (relational, files, etc.).

From an architectural point of view, Speedo is built on top of several ObjectWeb frameworks:

- JORM, a framework for the mapping of objects onto a persistent support, such as a relational database;
- MEDOR, a query framework supporting federation and distribution;
- Perseus, a persistence framework managing several aspects such as caching, pooling, concurrency control;
- Fractal, the ObjectWeb component model, and Julia, one of its implementation;
- ASM, a byte code manipulation framework;

The lock management at the memory level or at the database level (clustering support), the cache of persistent objects with various replacement policies configurable per class are part of the main benefits of Speedo.

Speedo provides a JMX console to monitor the core of the JDO container and a set of Eclipse plugins to tune Speedo features and to drive a JDO model made persistent with Speedo. It also provides a set of JCA resource adapters, enabling the integration of Speedo in J2EE servers such as JOnAS, JBoss or Weblogic.

A recent evaluation ("Ejboo" benchmark), with low caching effect, shows that Speedo is a real efficient solution when directly compared to JDBC or Hibernate 2.1.

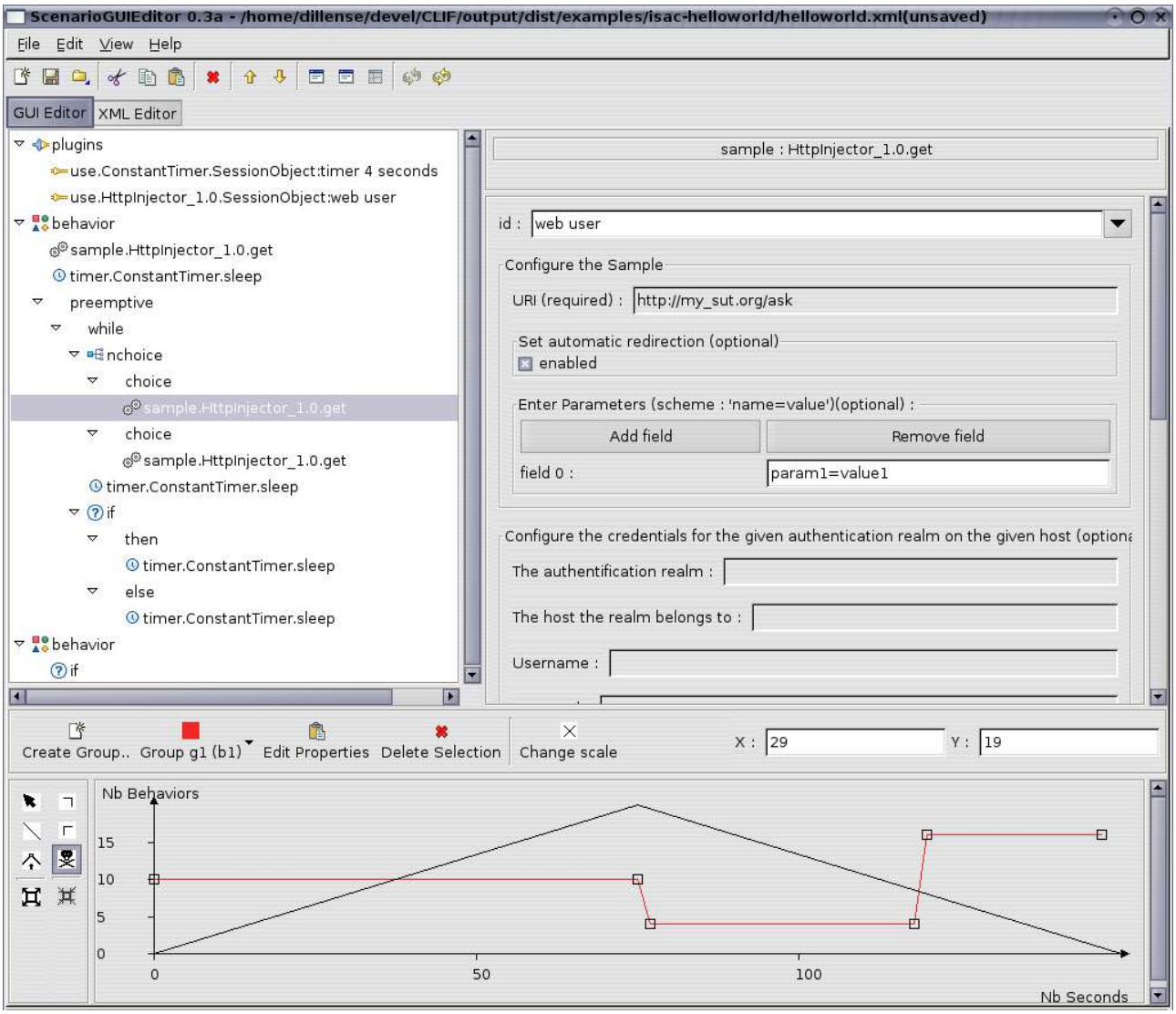

figure 2. ISAC scenario editor: plug-in imports and elementary behaviors definition (upper left corner), invocation parameters (upper right corner), load profiles (bottom part).

Fractal, ObjectWeb's component model

CLIF and Speedo are based on the Fractal component model (see http://fractal.objectweb.org/).

Open source middleware

CLIF, ISAC, Speedo and Fractal are open source software from ObjectWeb community.

Demonstration outline

Showing a load injection and performance evaluation platform without a target SUT makes no sense. Similarly, demonstrating a persistence middleware like a JDO implementation alone is almost impossible. So, this demonstration will give a meaningful opportunity to show both technologies.

The demonstration will :

- explain the objectives and architecture of the CLIF load injection framework, as shown by figure 1;
- introduce Speedo as well as a simple project management application based on Speedo;
- show how to use ISAC's scenario edition GUI (figure 2), together with the Speedo-specific ISAC plug-in, to define elementary client behaviors;
- show how to use the brand new Eclipse-based CLIF console (figure 3) to define, run and monitor a test plan (i.e. a distributed set of load injectors and probes with their respective parameters), and finally gather and browse the results;
- show how to monitor the application via the Speedo JMX console to get better performance, these modifications being reported on the CLIF console.

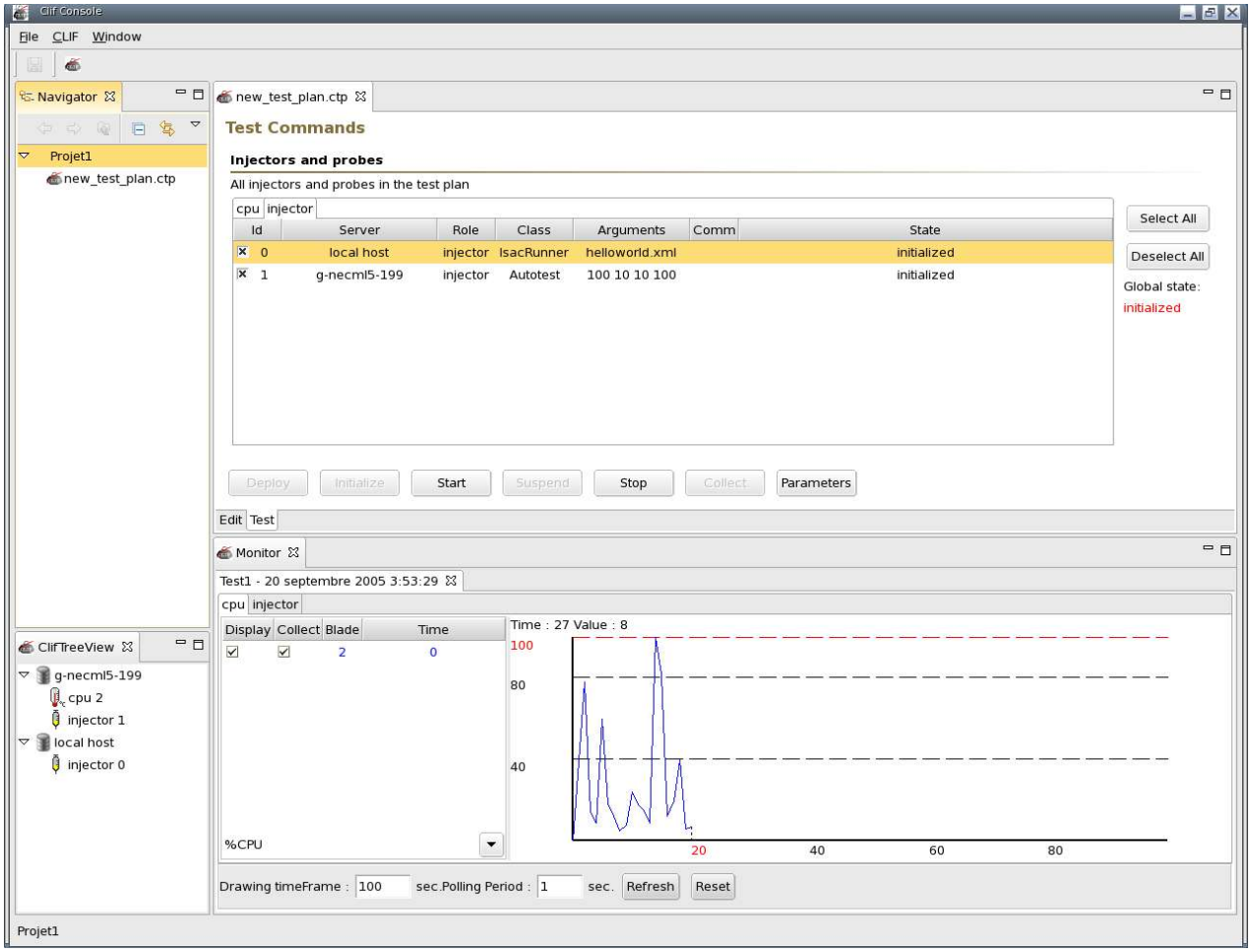

figure 3. CLIF graphical console based on Eclipse: project containing test plan definitions (upper left); test plan edition, deployment and execution control (upper right); test pan tree view (bottom left), monitoring of running load injectors and probes (bottom right).