CLIF is a Load Injection Framework

a new initiative of ObjectWeb consortium's Java Middleware Open Benchmarking project

Bruno Dillenseger, France Telecom R&D, Distributed Systems Architecture lab
bruno.dillenseger@francetelecom.com

Emmanuel Cecchet, INRIA
emmanuel.cecchet@inria.fr
Outline

➨ JMOB project overview
➨ Lessons learned with RUBiS
➨ Requirements for a load injection and performance measurement framework
➨ CLIF: design and current status
➨ Conclusion
ObjectWeb and the Java Middleware Open Benchmarking (JMOB) project
ObjectWeb & JMOB

ObjectWeb
- consortium fostering the development of open source middleware
- a « European Apache » with a W3C organization

JMOB
- middleware benchmarks
- open source implementations
- online experimental results
- place to exchange benchmarking experiences and software
JMOB current projects

Stock-Online
- stock market simulation
- J2EE benchmark
- reports from CSIRO (Australia)

RUBiS
- eCommerce web site modeled after eBay.com
- PHP, Servlets, 7 EJB implementations available
- ongoing JDO and .Net implementations
- Oopsia’02 paper

RUBBoS
- site modeled after slashdot.org
- PHP and Servlets
- Middleware’03 paper
Outline

- JMOB project overview
- Lessons learned with RUBiS
- Requirements for a load injection and performance measurement framework
- CLIF: design and current status
- Conclusion
Auction site modeled after eBay
26 interactions
Browsing mix: read-only mix
Bidding mix: 15% read-write interactions
Database: 1.4 GB
1 million users, ~500000 comments
>500000 items, 330000 active bids
RUBiS main results

➡ Academic side (Oopsla’02)
  ➢ Design pattern determines performance
  ➢ Reflection limits scalability
  ➢ Communication is the main CPU consumer

➡ Technical side
  ➢ Benchmarking is a nightmare
  ➢ 6 months of cpu time to obtain the results
  ➢ What is going wrong is the hardest question
  ➢ Load injection is as complex as the benchmark app.
RUBiS main results

→ **Load injection**
  - monitoring necessary not only on SUT (System Under Test)
  - one client emulator with a bottleneck resource at any point in time will give unstable results
  - online monitoring is (too?) expensive

→ **Resources needed for injection**
  - up to 5 clients machines to saturate one server
  - how many for a cluster of servers ?
  - how to scale ?
RUBiS main results

RUBiS load injection design

- ad-hoc client emulator using a transition state matrix
- distributed Java application
- monitoring using Linux specific tools (sar)
  - dump into temporary directory
  - post-mortem analysis of monitoring information
- awk+bash scripts for HTML reports generation
Outline

- JMOB project overview
- Lessons learned with RUBiS
- Requirements for a load injection and performance measurement framework
- CLIF: design and current status
- Conclusion
Existing solutions and projects

A couple of very good commercial solutions exist:

- high level loads, user-friendly, multiple target protocols

**but:**
- license costs are high, protocol-specific and depend on the required load level
- OS-bound (e.g. running on Windows only)
- how about testing custom protocols? (free extendibility issue)

Other solutions:

- more than 400 platforms/projects found on the Web! (not all alive)
- reduced load levels, target protocols, user-friendliness, supported OS
Typical technical limitations (1)

OS-specific...

- development for optimized injection performance
- Graphical User Interface
- access to system information about resources usage
- distribution and deployment support

Poor injection performance and scalability

- virtual machine execution overhead
- interpretation of scripting languages
- lack or bad tuning of distributed injection support
Typical technical limitations (2)

Fixed, single or reduced...

- load scenario definition tools/modes
  - scripting, GUI, XML-based or ad hoc configuration file, probabilistic state transition matrix, coarse-grain load profiles (ramps, peaks...)
- target protocols

Untransparent distribution management

- deployment of injectors and test scenarios
- collection of test results
The platform we are dreaming about is...

- **OS-independent**
- **versatile:**
  - target protocol independent and extendible
  - enables any kind of load profile to be generated
- **scalable to generate high load levels**
- **user-friendly**
  - for a variety of users
  - plain users, advanced users, developers...
  - handling cumbersome tasks
  - test deployment, results synthesis...
- **cheap**
Outline

- JMOB project overview
- Lessons learned with RUBiS
- Requirements for a load injection and performance measurement framework
- CLIF: design and current status
- Conclusion
CLIF: yet another benchmarking tool?

CLIF aims at providing the platform we are dreaming about:

- based on Java for hardware and OS independence
- distributed to enable high loads, with transparency support (deployment, control, data collection...)
- based on ObjectWeb's Fractal component model to have a neat, open architecture for straightforward adaptability (test scenarios, protocols)

- open source!
The big picture

- test deployment and control
- supervision console (interactive or batch mode)
- CLIF server (load injector and/or resource probe)
- target system invocation
- tested system resource probe(s)
- statistic data (polling)
- data analysis tool
- data: resources usage (CPU, memory, network...), test lifecycle events (deploy, init, start, end...), test reports (response time, error, result...)
- all data
- data storage
Fractal Component Model

ComponentIdentity  ContentController  BindingController  LifeCycleController

component’s controller

component’s content

shared component

component interface

server interface (offered service)  client interface (required service)
Focus on scenario component

**ScenarioControl:**
- init
- start
- stop
- suspend
- resume
- join

**(re-)deployed at runtime**

**DataCollector:**
- gives test data
  - response time, error, result, scenario events...

**ScenarioResponse:**
- gives information about scenario status
  - ready, running, suspended, finished...

**Collector**

**Engine**

**Clif server**
Current status

CLIF is a distributed load injection framework in Java, based on Fractal component model
multi-OS, open to any target protocol/system, open to any test scenario definition

Next development steps
- asynchronous collection, storage and analysis of data
- generic scenario tools (scripts, GUI, load profiles...)
- library of utilities for a variety of target protocols
- resource probe only implemented for Linux
- distributed log and user-defined alarm management
- state-of-art HTTP test tool (with integration of Rubis)
Future work

Further design work

- complete archival and exploitation support for full test campaigns
- scalability for very high loads (e.g. hierarchical organization of servers for test monitoring and control)
Conclusion

- Load injection is a complex distributed application
- CLIF is a framework for load injection
- Contributions and feedback welcome

Join us: jmob@objectweb.org
- Itanium-2 processors
- 104 nodes (Dual 64 bits 900 MHz processors, 3 GB memory, 72 GB local disk) connected through a Myrinet network
- 208 processors, 312 GB memory, 7.5 TB disk
- Connected to the GRID network
- Linux OS (RedHat Advanced Server)
- First Linpack experiments at INRIA (Aug. 2003) have reached a 560 GFlop/s performance

- Applications: Grid computing experiments, classical scientific computing, high performance Internet servers, …