Performance Measurement and Applications Benchmarking with Erlang

11th Erlang User Conference
November 10, 2005

Mickaël Rémond

Process One

The benchmarking challenge

- **Applications benchmarking and performance mesurement requires:**
  - To be able to simulate an important number of users playing real-life scenarii.
  - To be able to do near real-time and high throughput mesurements to provide reliable figures.
- **Tsunami is a distributed load testing tool that has been designed as a heavy duty benchmarking tool and framework**
- **The software is protocol-independent. It currently can be used to stress HTTP, SOAP and Jabber/XMPP servers, but other protocols can be added.**
- **This talk presents Tsunami main achievements, along with real life use cases, and explore possible framework extensions and improvements.**

Process One

Summary — What is Tsunami ?

- **Tsunami main strength rely in its ability to simulate:**
  - a huge number of simultaneous users from a single CPU,
  - an heavier load in cluster mode.
- **When used on cluster you can generate a really impressive load on a server with a modest cluster, easy to set-up and maintain.**
- **Tsunami is developed in Erlang and this is where the power of Tsunami relies. Tsunami is based on the Erlang OTP (Open Transaction Platform) and inherits several characteristics from Erlang:**
  - **Performance:** Erlang has been made to support hundred thousands of lightweight processus in a single virtual machine.
  - **Scalability:** Erlang development environnement is naturally distributed, promoting the idea of processus location transparence.
  - **Fault-tolerance:** Erlang has been built to develop robust, fault-tolerant systems. As such, wrong answer sent from the server to Tsunami does not make the whole running benchmark crash.

Process One

Tsunami background: A strong simulation model

- **Tsunami has been developed by Nicolas Niclausse**

- **It is an industrial implementation of a stochastic model for real users simulation. User events distribution is based on a Poisson Law. This model is used to closely simulate real-world user behaviour.**

- **Tsunami is being developed since 2001, first by IDEALX and now by Process-one.**

- **This model has already been tested in the INRIA Wagon project (Web trAffic GeneratOr and beNchmark).**

  The Wagon project has been developed in the context of MISTRAL. Its main objectif was to simulate various types of Internet trafic to study server behaviours. WAGON has been used in the context to the French national VHTD (Vraiment Très Haut Débit) project.

- **Tsunami is based on the result of Nicolas Niclausse PHD Thesis:**

  Modeling, performance analysis and dimensioning of the WWW

Process One

Tsunami main features (1/2)

- **High Performance:**
  - Tsunami can simulate a huge number of simultaneous users per physical computer. It can simulates up to 10000 users on a single CPU. Traditionnal injection tools can hardly go further than 200 users.
- **Distributed:**
  - The load can be distributed on a cluster of client machines
- **Multi-Protocols using a plugin system:**
  - HTTP (both standard web traffic and SOAP) and Jabber are currently supported. LDAP and SMTP are on the TODO list.
- **SSL support**
- **Several IP addresses can be used on a single machine using the underlying OS IP Aliasing**
- **OS monitoring:**
  - CPU, memory and network trafic can be monitored using Erlang agents on remote servers or SNMP

Process One

Tsunami main features (2/2)

- **XML configuration system**
- **Mixed behaviours:**
  - Several sessions can be used to simulate differents type of users during the same benchmark. You can define the proportion of the various behaviours in the benchmark scenario.
- **Stochastic processes:**
  - In order to generate a realistic trafic, user thinktimes and the arrival rate can be randomize using a probability distribution (exponential currently)
- **Adaptative scenarii:**
  - Scenarii can have dynamic part, that depends on the result of the current scenario request or can be generated by Erlang code.
- **Complete statistic set**

HTTP related features

- HTTP/1.0 and HTTP/1.1 support
- GET and POST requests
- Cookies: Automatic cookies management
- 'GET If-modified since' type of request
- WWW-authentication Basic
- Proxy mode to record sessions using a Web browser
- SOAP support using the HTTP mode (the SOAPAction HTTP header is handled).

Jabber related features

- Authentication, presence and register messages
- Chat messages to online or offline users
- Roster set and get requests
- Global users' synchronisation can be set on specific actions

Complete reports set

- **Mesures and statistics produced by Tsunami are extensive. They are all represented as a graphe. Tsumami produces statistics regarding:**
  - **Performance:** response time, connexion time, decomposition of the user scenario based on request grouping instruction, requests per second
  - **Errors:** Statistics on page return code to trace errors
  - **Target cluster behaviour:** An Erlang agent can gather information from the target cluster. Tsunami produce graphes for CPU and memory consumption and network traffic. SNMP is also supported.
- **Note that Tsunami take care of the synchronisation process by itself. Gathered statistics are «synchronized».**
- **It is possible to generate graphes during the benchmark as statistics are gathered in real-time. This make it possible to see if the benchmark has to be stopped before the end of the benchmark.**

HTTP benchmark approach

- Record scenario: tsunami start_recorder
- Edit / organise scenario
- Write small code for dynamic parts if needed and place dynamic mark-up in the scenario
- Test and adjust scenario to have a nice progression of the load. This is highly dependent of the application and of the size of the target cluster. Calculate the normal duration of the scenario and use the interarrival time between users and the duration of the phase to estimate the number of simultaneous users for each given phase.
- Launch benchmark with your first application parameters set-up: tsunami start
- Analyse results, change parameters and launch another benchmark

Understanding `tsunami.xml` file: File structure

- **Scenarii are enclosed into tsunami tags:**

```
<?xml version="1.0"?>
<tsunami loglevel="info" dumptraffic="false">
...
</tsunami>
```

Understanding `tsunami.xml` file: Clients and server

- **Scenarii start with a clients (Tsunami cluster) and server definition:**

```
<clients>
   <client host="louxor" weight="1" maxusers="500">
       <ip value="10.9.195.12"></ip>
       <ip value="10.9.195.13"></ip>
   </client>
   <client host="memphis" weight="3" maxusers="250" cpu="2">
       <ip value="10.9.195.14"></ip>
   </client>
</clients>

<server host="10.9.195.1" port="8080" type="tcp"></server>
```

- **Several virtual IP can be used to simulate more machines. This is very useful when a load-balancer use the client's IP to distribute the traffic amoung a cluster of servers. In this example, a second machine is used in the Tsunami cluster, with a higher weight, and 2 cpus. Two Erlang virtual machines will be used to take advantage of the number of CPU.**

- **The server is the entry point into the cluster (Only one server should be defined).**

Understanding `tsunami.xml` file: Monitoring

- **Scenarii can contain optional monitoring informations. For example, here is a cluster monitoring definition based on Erlang agents, for a cluster of 6 computers:**

```
<monitoring>
  <monitor host="geronimo" type="erlang"></monitor>
  <monitor host="bigfoot-1" type="erlang"></monitor>
  <monitor host="bigfoot-2" type="erlang"></monitor>
  <monitor host="f14-1" type="erlang"></monitor>
  <monitor host="f14-2" type="erlang"></monitor>
  <monitor host="db" type="erlang"></monitor>
</monitoring>
```

- **The type keyword snmp can replace the erlang keyword, if SNMP monitoring is prefered. They can be mixed. erlang is the default value for monitoring.**

- **Note: For Erlang monitoring, monitored computers need to be accessible through the network. SSH needs to be configured to allow connection without password on**
  - (See: Tutorial: Erlang - Starting a set of cluster nodes on Erlang-projects.org for details)

Understanding `tsunami.xml` file: Defining the load progression

- **The load progression is set-up by defining several arrival phases:**

```
<arrivalphase phase="1" duration="10" unit="minute">
  <users interarrival="2" unit="second"> </users>
</arrivalphase>

<arrivalphase phase="2" duration="10" unit="minute">
  <users interarrival="1" unit="second"> </users>
</arrivalphase>

<arrivalphase phase="3" duration="10" unit="minute">
  <users interarrival="0.1" unit="second"> </users>
</arrivalphase>
```

Understanding `tsunami.xml` file: Default values

- **Default values can be set-up globally: thinktime between requests in the scenario and ssl cipher algorithms. These values overrides those set in session configuration tags.**

```
<default name="thinktime" value="3" random="false"/>
<default name="ssl_ciphers"
        value="EXP1024-RC4-SHA,EDH-RSA-DES-CBC3-SHA"/>
```

- **Default values for specific protocols can be defined. Here is an example of default values for Jabber:**

```
<default type="ts_jabber" name="global_number" value="5" />
<default type="ts_jabber" name="userid_max" value="100" />
<default type="ts_jabber" name="domain" value="jabber.org" />
<default type="ts_jabber" name="username" value="glop" />
<default type="ts_jabber" name="passwd" value="glop" />
```

Understanding tsunami.xml file: Sessions (1/2)

- **Sessions define the content of the scenario itself. They describe the requests to execute.**

```
<session name="http-example" popularity="70"
        persistent="true" messages_ack="parse" type="ts_http">

  <request> <http url="/" method="GET" version="1.1">
                </http> </request>
  <request> <http url="/images/logo.gif"
             method="GET" version="1.1"
             if_modified_since="Fri, 14 Nov 2003 02:43:31 GMT">
            </http></request>

  <thinktime value="20" random="true"></thinktime>

  <transaction name="index_request">
   <request><http url="/index.en.html"
                       method="GET" version="1.1" >
            </http> </request>
   <request><http url="/images/header.gif"
                       method="GET" version="1.1">
            </http> </request>
  </transaction>
```

Understanding tsunami.xml file: Sessions (2/2)

```
    <thinktime value="60" random="true"></thinktime>
    <request>
      <http url="/" method="POST" version="1.1"
              contents="bla=blu">
      </http> </request>
    <request>
       <http url="/bla" method="GET" version="1.1"
             contents="bla=blu&name=glop">
       <www_authenticate userid="Aladdin"
                         passwd="open sesame"/></http>
    </request>
  </session>

  <session name="backoffice" popularity="30" ...>
  ... </session>
```

- **The popularity is the frequency of this type of session. This is used to decided which session a new user will execute. The sum of all session's popularity must be 100.**

- **This example show several features of the HTTP protocol support in Tsunami: GET and POST request, basic authentication, transaction for statistics definition, ... The same approach can be used for defining Jabber/XMPP session.**

Understanding `tsunami.xml` file: Dynamic substitutions (1/2)

- Dynamic substitution are mark-up placed in element of the scenario. For HTTP, this mark-up can be placed in basic authentication (www_authenticate tag: userid and passwd attributes), URL (to change GET parameter) and POST content.

- Those mark-up are of the form `%%Module:Function%%`. Substitutions are executed on a request-by-request basis, only if the request tag has the attribute subst="true".

- When a substitution is asked, the substitution mark-up is replaced by the result of the call to the Erlang function: Module:Function(Pid).

- Here is an example of use of substitution in a Tsunami scenario:

```
<session name="rec20040316-08:47" popularity="100"
         persistent="true" messages_ack="parse" type="ts_http">
 <request subst="true">
  <http url="/echo?symbol=%%symbol:new%%" method="GET">
  </http></request>
</session>
```

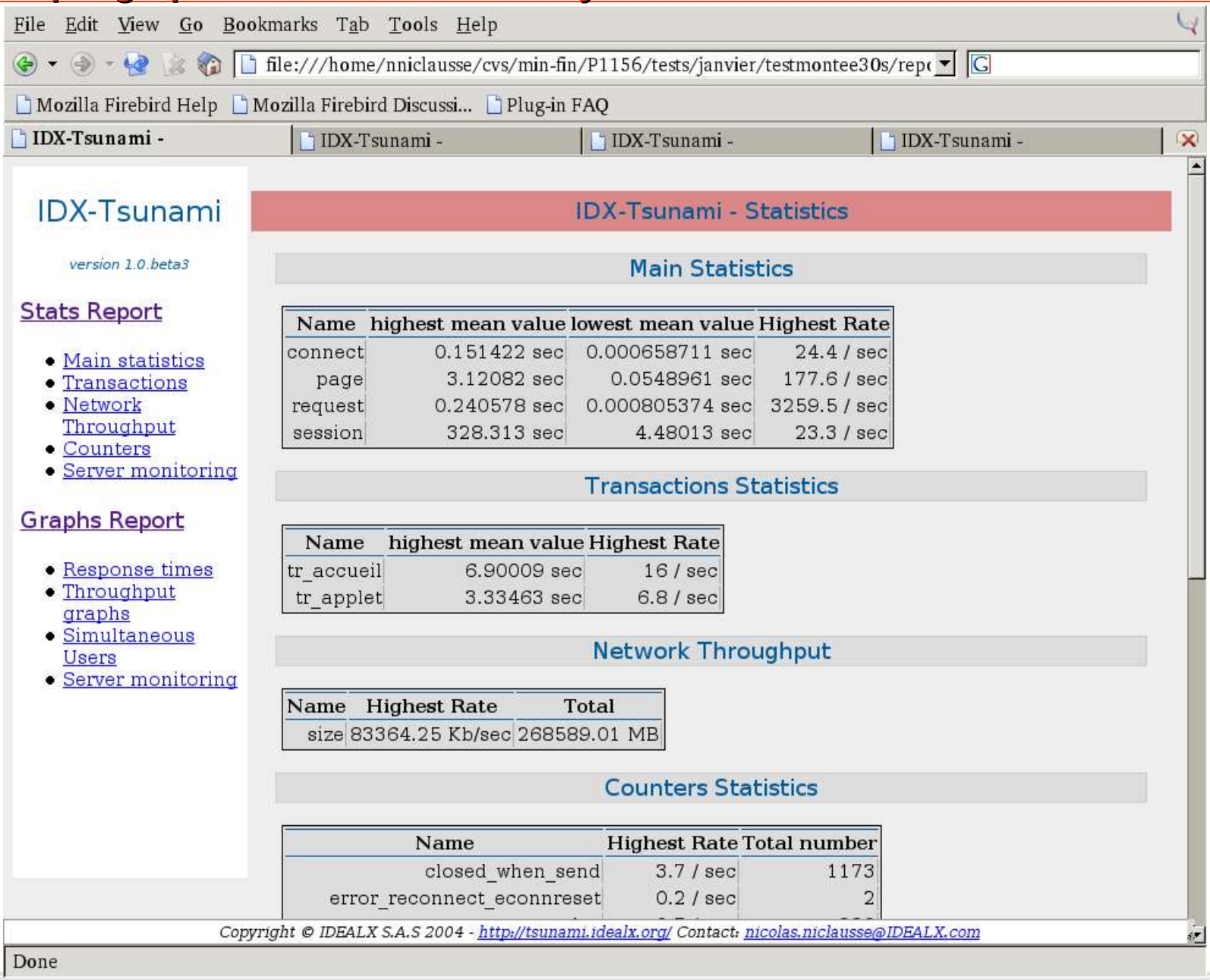
Understanding `tsunami.xml` file: Dynamic substitutions (2/2)

- **Here is the Erlang code of the module used for dynamic substitution:**

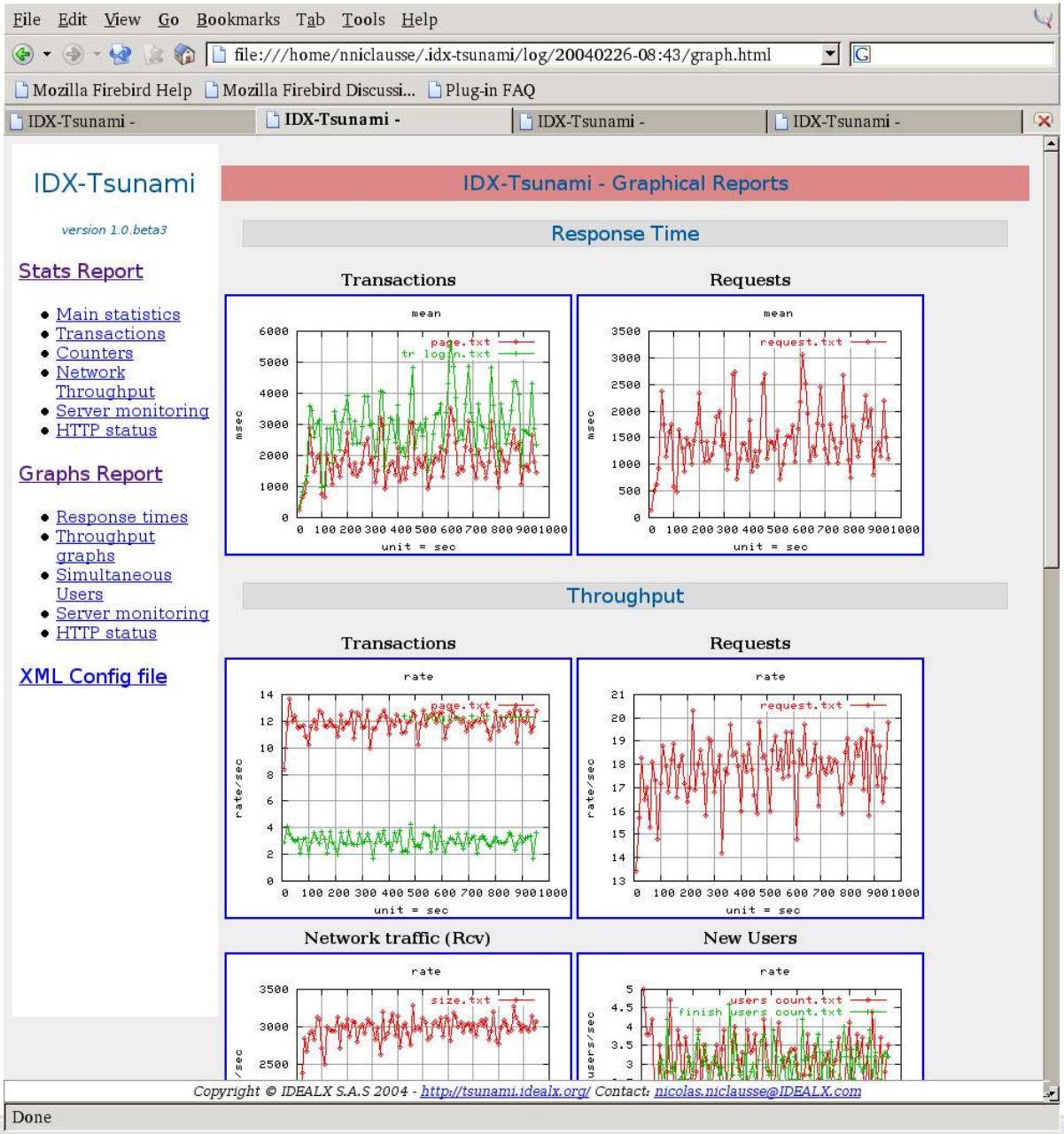
```
-module(symbol).
-export([new/1]).

new(Pid) ->
    case random:uniform(3) of
        1 -> "IBM";
        2 -> "MSFT";
        3 -> "RHAT"
    end.
```

- **As you can this, writing scenario with dynamic substitution is trivial.**

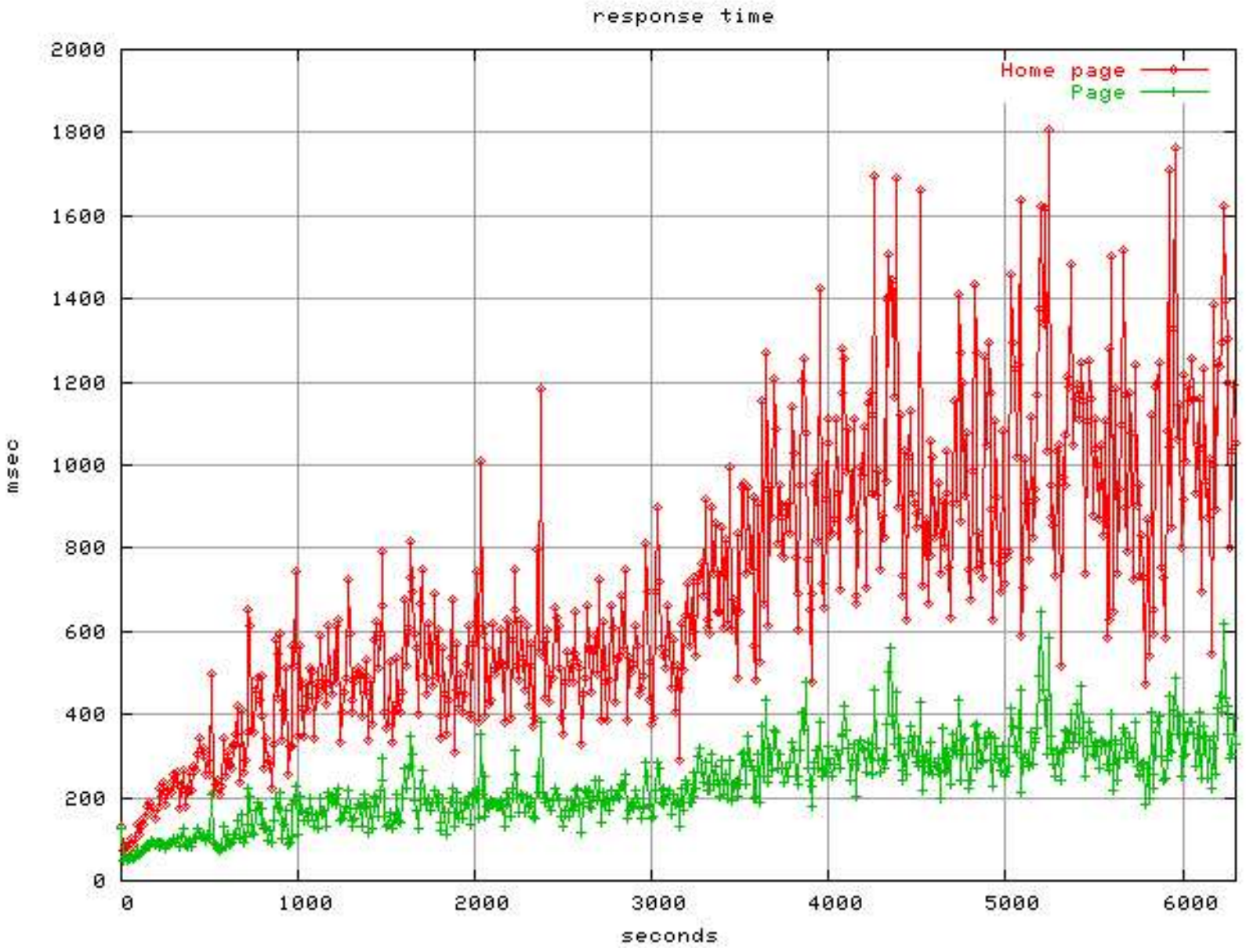
Example graphes: Tsunami summary



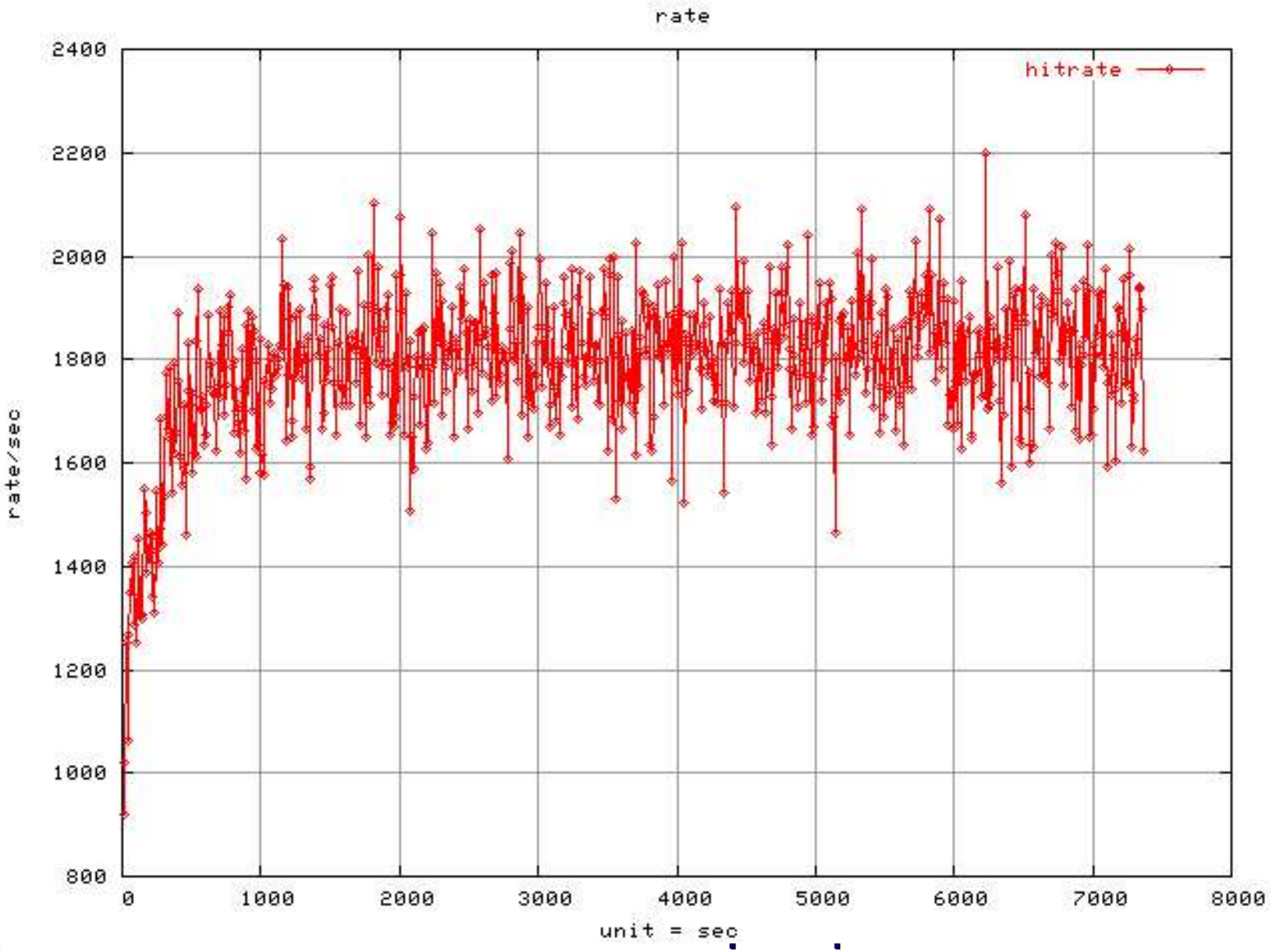
Example graphes: Statistics overview



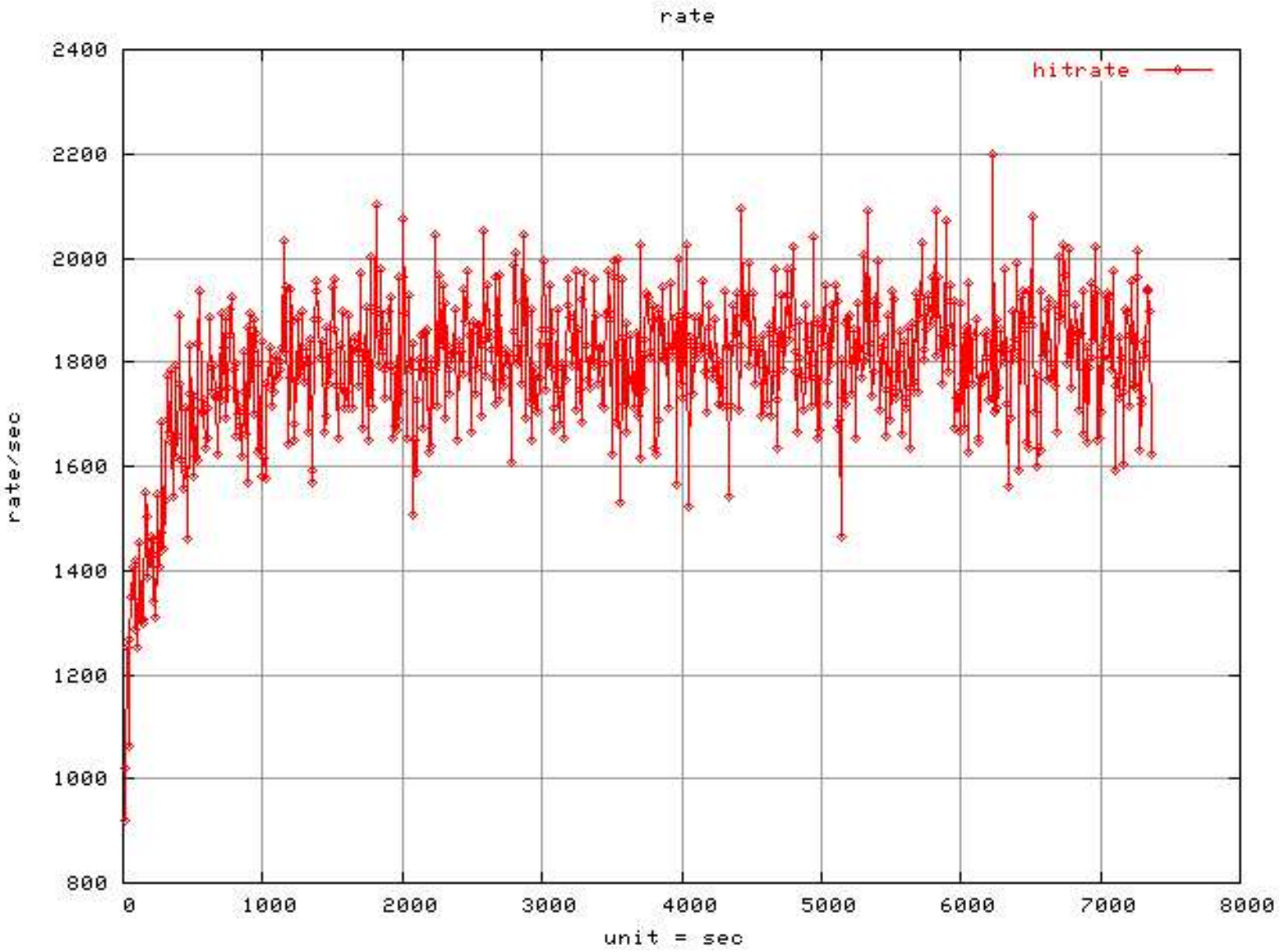
Example graphes: Response time



Example graphes: Hit rate



Example graphes: Network traffic



Figures and organisations using Tsunami

- **Tsunami has been successfully used for huge benchmark:**
  - Jabber protocol: 10 000 simultaneous users. Tsunami were running on a 3-computers cluster (CPU 800Mhz)
  - HTTP and HTTPS protocol: 25 000 simultaneous users. Tsunami were running on a 4-computers cluster. The tested platform reached 3 000 request per second.

- **Tsunami has been used for benchmark at:**
  - DGI (Direction Générale des impôts): French finance ministry
  - Cap Gemini Ernst & Young
  - IFP (Institut Français du Pétrole): French Research Organisation for Petroleum
  - LibertySurf
  - ...

Conclusion

- **Tsunami has several advantages over other injection tools:**
  - Outstanding **performance** and distributed benchmark
  - **Ease of use:** The hard work is already done for all supported protocole No need to write complex scripts. Dynamic scenarii only require small trivial piece of code. Tsunami scenarii realisation is mostly based on
  - **Multi-protocol support:** Tsunami is for example one of the only tool to benchmark SOAP applications
  - **Monitoring** of the target cluster to analyse the behaviour and find bottlenecks. For exemple, I did use it to analyse cluster symmetry (is the load properly balanced ?) and to determine the best combination of machines on the three cluster tiers (Web engine, EJB engine and database)