Erlang Solutions Ltd.

Inviso and Exago

Tracing and log analysis in multiple-node environments

Ulf Wiger, Aniko Nagyné Víg, Bartłomiej Puzoń
Erlang Solutions Ltd

Erlang Factory, San Francisco, 25 Mar 2010

Initial Problem

The Protest project:
EU-funded research on Property-based Testing

Tracing and log analysis work package:

How to conduct safe and efficient run-time trace analysis on distributed systems?

How to do advanced post-mortem log analysis?

(or indeed log analysis on running systems?)

Eventually reuse high-level properties from testing

ProTest property based testing

Erlang

Tracing support in Erlang

The trace() BIFs

Low-level trace message generation

Dynamic control using Match Specifications

The DBG application

Command-line wrappers around the trace BIFs

(Redbug, a dbg alternative made by Mats Cronqvist)

Observer, Trace Tool Builder, etop, et, pman

Various loosely connected utilities

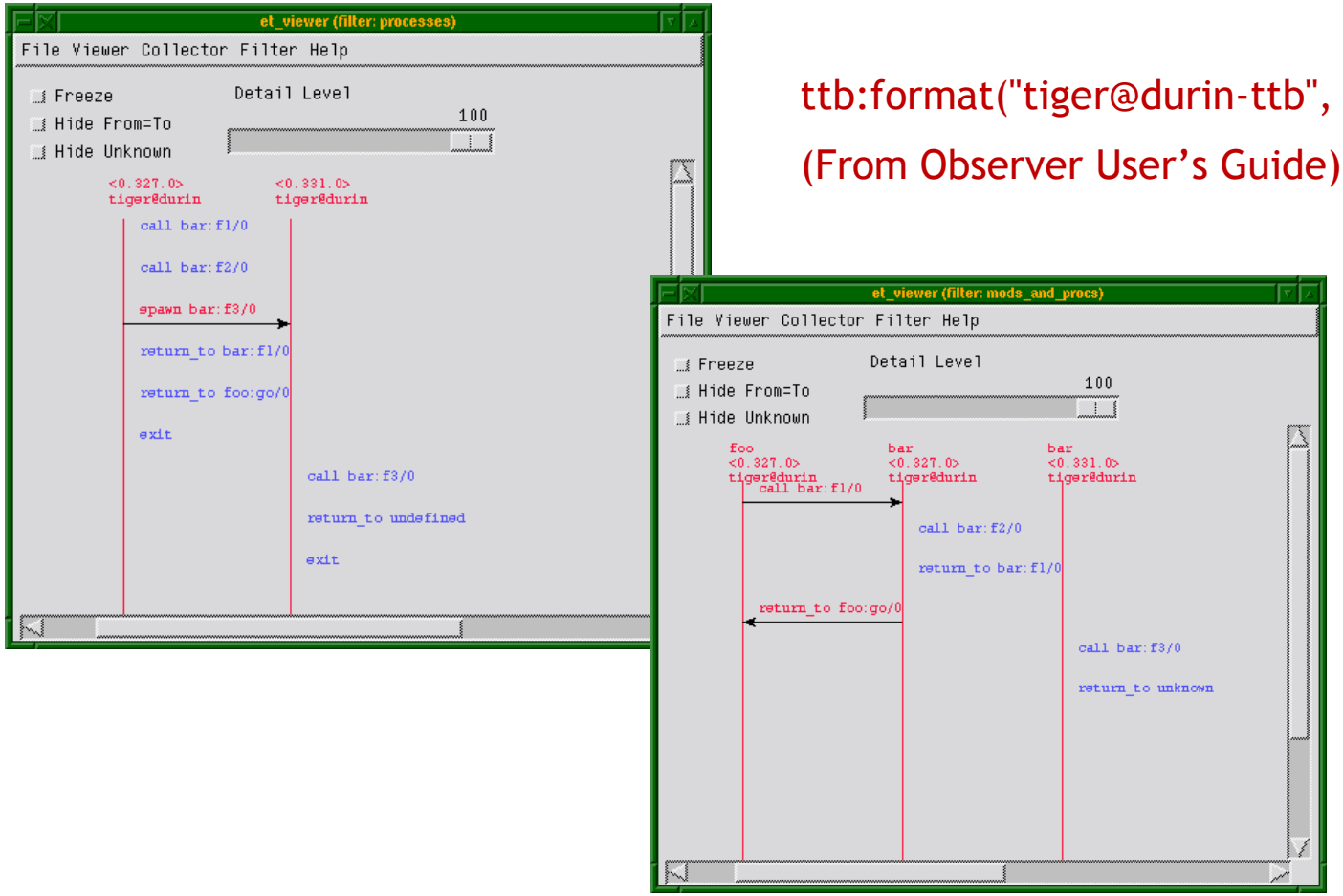
Percept, eprof, cprof, fprof, instrument

Profiling tools with different characteristics

ProTest property based testing

Erlang

Lots of functionality, Hard to Grep



ttb:format("tiger@durin-ttb", [{handler, et}])

(From Observer User's Guide)

ProTest property based testing

Erlang

“Inviso” – Latin: “to visit; inspect; look at”

Inviso

Inviso – Distributed Trace Tool Builder

Part of OTP

Partly unfinished

API mainly intended for tool makers

Protest project contributions

Inviso Tracing tutorial:
http://www.trapexit.org/Inviso_tracing_tutorial

Onviso, a user-friendler API
http://www.trapexit.org/Tracing_with_Onviso

ProTest property based testing

Erlang

Inviso Architecture

Inviso Features

Start tracing simultaneously on multiple nodes

Collect and merge traces

Predefine “trace cases”

Meta-tracing

Pid/regname translation

Call/return value matching

“autostart” – tracing starts automatically at boot time

Overload protection – stop tracing if overload detected

ProTest property based testing

Erlang

Onviso

User-friendly API to Inviso

Set up and run tracing using only two commands

Shortcuts for commonly used trace patterns (inspired by Redbug)

Additional functionality

Non-destructive merge of trace logs

Log merge can be used for property-based testing

Useful defaults for merging and overload protection

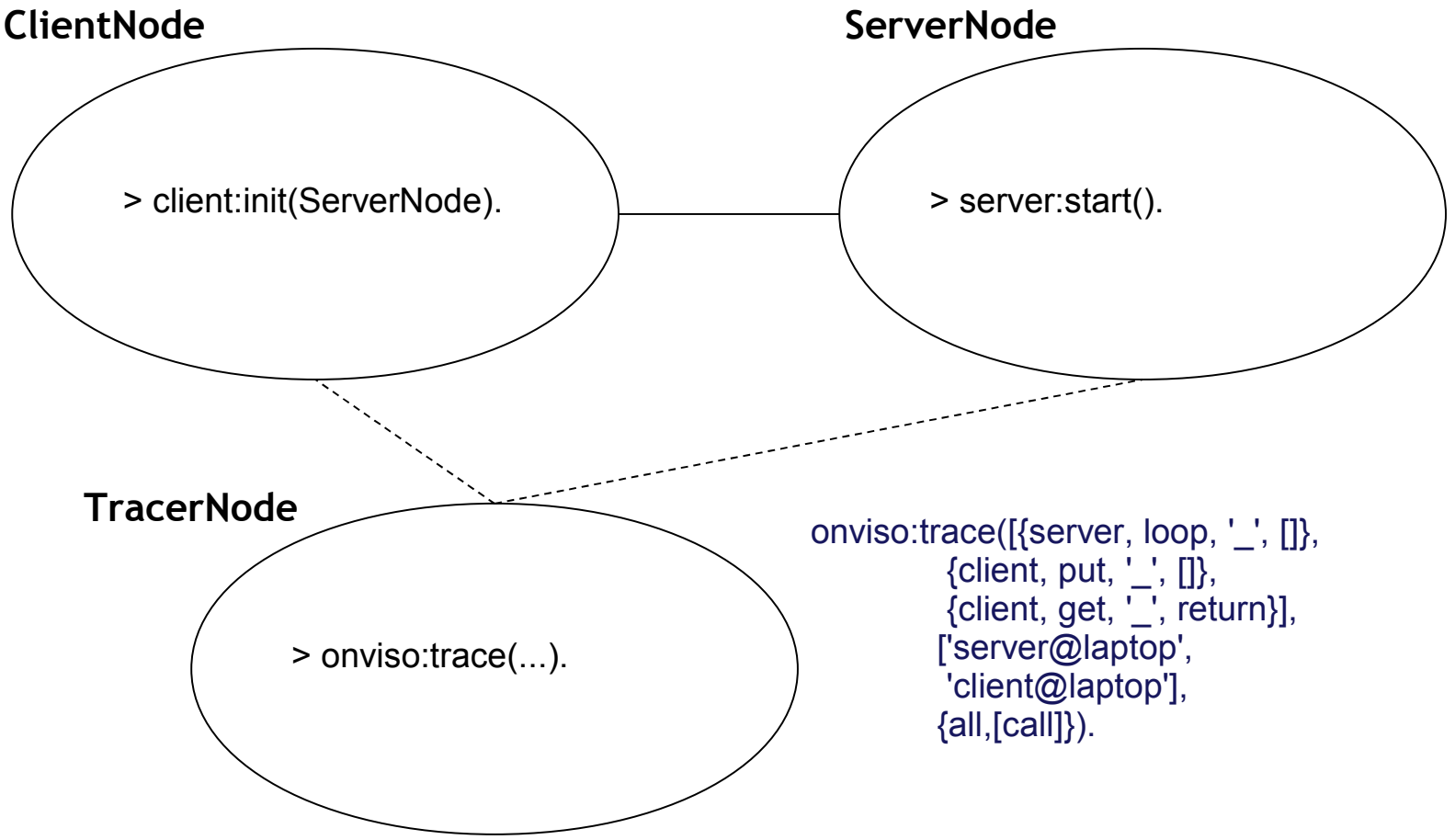
Trace node automatically reconnects to restarting target nodes

Status: Work in Progress

ProTest property based testing

Erlang

Demo – Starting the Nodes



ProTest property based testing

Erlang

Interrupting a Trace

One of the nodes can be restarted:

client@laptop> init:restart().

client@laptop> client:init('server@laptop').

By default Onviso will reconnect and resume tracing on the client node.

If the node restarts abruptly, some of the trace data may be lost (as the trace buffers might not be flushed to the files).

Inviso (and thus, Onviso) can handle incomplete trace logs.

Stopping a Trace

Every trace call returns a trace reference identifier.
This id can be used to stop or merge a trace

> onviso:stop(Id).

The traces are collected to files and
distributed back to the Inviso control node

Onviso Log Merge – Defaults

```
> onviso:merge(Id, void, void, shell). % result in the shell

> onviso:merge(Id, void, void, file).
      % result in "outputId.txt" file
      % other easy options {file, Name}, {file_prefix, Prefix}
```

You can merge the trace files more than once

Tracing is stopped automatically when merge is called

ProTest property based testing

Erlang

Merge a Trace - custom function

```
BeginFun = fun(_InitData) ->
             {ok, 0}
           end.
```

(Initialize state)

```
WorkFun = fun(_Node, _Trace, _PidMapping, Count) ->
            {ok, Count + 1}
          end.
```

(Format/filter trace data)

```
EndFun = fun(Count) ->
           io:format("Got ~p traces.~n", [Count])
         end.
```

(Typically write to file)

```
> onviso:merge(1, BeginFun, WorkFun, EndFun).
```


ProTest property based testing

Erlang

Onviso Command line interface

Example of a higher-level trace tool

Help testers and support staff define and/or execute trace cases

```
(inviso@debian)6> cli:start().
Onviso Demo GUI
========================================


 > Main Menu
-------------------------------------
 1) Add trace case
 2) List/Run trace cases
 3) Save configuration to file
 4) Load configuration from file
 5) Set the magic cookie
 6) Exit
[Q] Choice [1-6] : 6

 Exiting...ok
```


ProTest property based testing

Erlang

εξαγω Ανχιεντ Γρεεκ: βρινγ φορτη

Exago

A “log mining” Approach

Identify a set of logs

Tell Exago how to extract timestamp and session id from each

Possibly define a time offset per log

Simplify log entries using an “abstraction function”

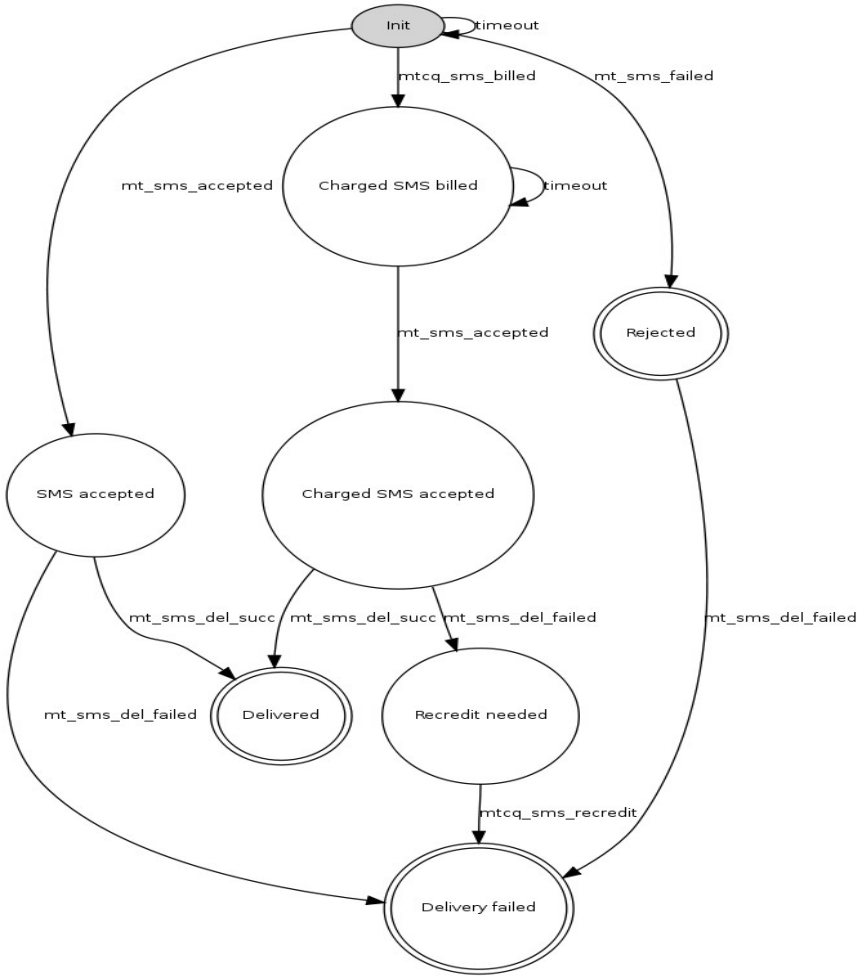
Define a finite state machine (FSM) for the analysis

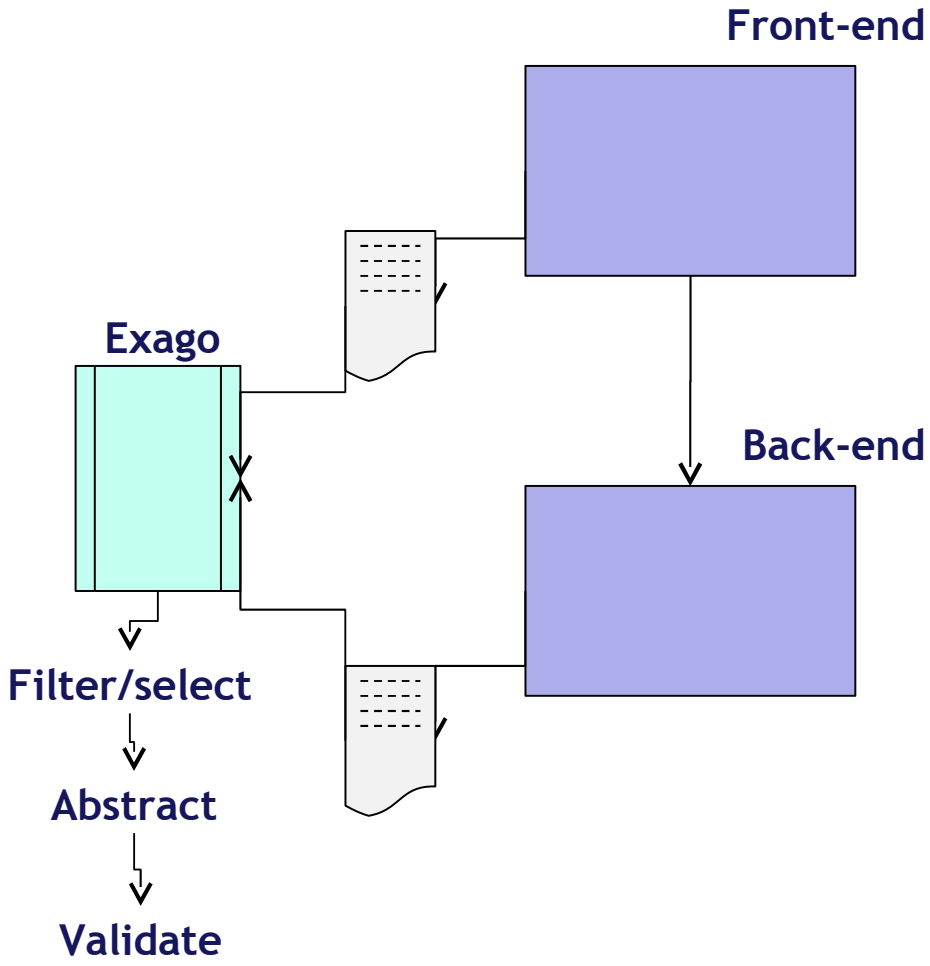
Exago will present sessions that do not conform with the FSM

ProTest property based testing

Erlang

Log Correlation





ProTest property based testing

Erlang

Abstraction Functions

```
transaction_conf = none,
session_conf =
  #sess_conf{
     sort = timestamp,
     abstr_fun =
        fun(Trs) ->
           lists:map(
              fun({Timestamp, AbstrVal, File}) ->
                 L = string:tokens(filename:basename(File),"_."),
                 AbstrCommand =
                    case L of
                       ["mt", "sms" | _Rest] ->
                          mt_sms_accepted;
                       ["mtcq", "sms" | _Rest] ->
                          mtcq_sms_billed;
                        ...
                    end,
                 {Timestamp, AbstrCommand}
              end, Trs)
        end,
     ...},
```

ProTest property based testing

Erlang

Case Study: SMS Gateway

```
[{"2008-08-07_05:34:10:862",mtcq_sms_billed},
 {"2008-08-07_05:34:15:864",timeout},
 {"2008-08-07_05:34:15:864",{mt_sms_del_failed,{"timedout"}}},
 {"2008-08-07_05:34:21:275",mt_sms_accepted},
 {"2008-08-07_05:34:29:010",mt_sms_del_succ}]
```

Gateway times out, delivers a failure report to user

SMSC finally reports successful delivery, gateway forwards it

User gets conflicting reports + could interfere with SMS retry

2 occurrences among 20,000 sessions in the log

Exago pilot duration: 2 days

System had been in production for two years...

ProTest property based testing

Erlang

Exago Status

Open Source release imminent

Two case studies

Finding bugs in a well-tested stable system

Using Exago in the early stages of development

Need to support more log formats

Improve usability

Test scalability

Investigate applying QuickCheck's Temporal Relations

ProTest property based testing

Erlang

Thank you

Questions?