



# Inviso/Onviso

Aniko Nagyné Víg

Stockholm, 13 Nov 2009



# Contents

Short Onviso command summary

Basic exercises client-server

Elevator exercises

## Onviso commands one pager

0. Start up all the target and monitor nodes, set the cookies.

1. Start the trace by calling `onviso:trace/2-4` functions with

- Pattern list: ex. `{client, get, '_', return}`
- Node list: ex. `'client@laptop'` - optional
- Flags: ex. `{all,[call]}` jskdhks
- Overload protection

2. Merge or stop:

```
onviso:stop(Id)
```

```
onviso:merge(Id,BeginFun, WorkFun, EndFun)
```

Copyright 2008



## Erlang trace BIFs possible settings

Dynamically enabled. Filtering on pids: existing, new, all, pid()

### Flags

- `'send'`, `'receive'` for message passing events
- `'running'` for scheduling events: `'in'` and `'out'` messages sent when a process is scheduled or preempted
- `'exiting'` for scheduling exiting processes. Message tags: `'in_exiting'`, `'out_exiting'`, and `'out_exited'`.
- `'procs'` for process-related events. Message tags: `'spawn'`, `'exit'`, `'link'`, `'unlink'`, `'register'`, `'unregister'`, `'getting_linked'`, `'getting_unlinked'`
- `'call'` for function calls. Message tags: `call`, `return_from` It can be combined with `arity`, `return_to` and `silent` flags. In `trace_pattern` `match_specifications` can be defined.
- `'set_on_spawn'`, `'set_on_first_spawn'`, `'set_on_link'`, `'set_on_first_link'` sets the inheritance of the trace flags in the new processes
- `'garbage_collection'`. Message tags: `'gc_start'`, `'gc_stop'`.
- `'timestamp'` and `'cpu_timestamp'` for including timestamps to each trace message.

Copyright 2008



## Match specifications

MatchExpression ::= [ MatchFunction, ... ]

MatchFunction ::= { MatchHead, MatchConditions, MatchBody }

### Examples:

- Match an argument list of three where the first and third arguments are equal:

```
[{ '$1', '_', '$1', [], [] }
```

- Match all objects with arity > 1 and the first element is 'elem', return the 2. element.

```
[{ '$1', [{ '=' , elem, { element, 1, '$1' } }, { '>' , { size, '$1', 2 } } ],
```

```
  [ { element, 2, '$1' } ] }
```

Copyright 2008



## Contents

Short Onviso command summary

Basic exercises client-server

Elevator exercises

## Client Server Setup

Start and initialise all nodes in the system: in this example the server and client nodes (Based on the client.erl, server.erl files and both containing the runtime application)

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

```
> server:start().
```

Start a separate monitoring invisio node (running invisio and onvisio applications)

Set the same cookie on every node:

```
> erlang:set_cookie(node(), invisio).
```

Copyright 2008



## Exercise 1.1

Start up the traces on the invisio node on the following functions:

```
server:loop, client:get, client:put
```

- Start tracing on the function calls
- Start tracing on the return values of the function calls in the client
- Start tracing on the message passing

When starting the trace, try to write first the Patterns, then the node list and the decide the flags.

NOTE: Don't forget to generate some traffic on the nodes, before merging/stopping.

For stop and merge use the default for all the traces:

```
invisio:merge(Id, void, void, shell).
```

Copyright 2008



## Exercise 1.1 - Solution

- a. `onviso:trace([server, loop, '_', []],  
                {client, put, '_', []},  
                {client, get, '_', []}],  
                ['server@laptop', 'client@laptop'],  
                {all, [call]}).`
- b. `onviso:trace([server, loop, '_', []],  
                {client, put, '_', []},  
                {client, get, '_', return}],  
                ['server@laptop', 'client@laptop'],  
                {all, [call]}).`
- c. `onviso:trace([server, '_', '_', []],  
                {client, '_', '_', []}],  
                ['server@laptop', 'client@laptop'],  
                {all, ['send', 'receive']}).`

Copyright 2008



## Exercise 1.2

Use any of the previous traces, but instead of printing to the shell, try to write it to a file.

Try to set the filename in the next step.

Copyright 2008



## Exercise 1.2 - Solution

NOTE: You don't need to run the trace again, you can simple reuse the Id from a previous trace.

For starting the trace for example:

```
onviso:trace([server, loop, '_', []],
             {client, put, '_', []},
             {client, get, '_', []}],
             ['server@laptop', 'client@laptop'],
             {all, [call]}).
```

The solution for merge:

```
onviso:merge(Id, void, void, file).
onviso:merge(Id, void, void,
             {file_prefix, "NewMerge" }) % or {file, "New.txt" })
```

Copyright 2008



## Exercise 2

Set up a second client node.

Count how much messages were sent by the clients based on the traces.

Try to think different solutions.

Copyright 2008



## Exercise 2 - Solution

### Trace:

```
onviso:trace([client, put, '_'], []],  
             ['client1@laptop', 'client2@laptop'],  
             {all, [call]}).
```

### Merge:

```
> BeginFun = fun(_InitData) -> {ok,0} end.  
  
> WorkFun = fun(_Node, _Trace, _PidMapping, Count) ->  
{ok, Count + 1} end.  
  
> EndFun = fun(Count) -> io:format("The clients sent ~p messages.\n", [Count])  
end.  
  
> onviso:merge(1, BeginFun, WorkFun, EndFun).
```

Copyright 2008



## Contents

Short Onviso command summary

Basic exercises client-server

Elevator exercises

## Elevator Setup

Start the elevator node in the system: the elevator application  
(Based on the elevator files and containing the runtime application and gs)

Start a separate monitoring invisio node (running invisio and onvisio applications)

Set the same cookie on every node:

```
> erlang:set_cookie(node(), invisio).
```

Copyright 2008



## Exercise 3.1

Count how many floors the lifts travelled during the examined period.

(Going up 3 levels and coming down 2 results in 5 this case)

The elevators can be started by

```
util:start(Id::int(), Floors::int, Elevators::int())
```

Copyright 2008



## Exercise 3.1 Solution

Trace:

```
onviso:trace([elevator, closed, '_'],  
[{{{{'$1','_','_'}},[{'='},move,'$1'}],[]}}],['elevator@aniko-laptop'],{all,[call]}).
```

Merge:

```
> BeginFun = fun(_InitData) -> {ok,0} end.  
  
> WorkFun = fun(_Node, _Trace, _PidMapping, Count) ->  
{ok, Count + 1} end.  
  
> EndFun = fun(Count) -> io:format("The clients sent ~p messages.\n", [Count])  
end.  
  
> onviso:merge(1, BeginFun, WorkFun, EndFun).
```

Copyright 2008



## Exercise 3.2

Count how many times stopped the n. elevator on the m. or higher floor.

Copyright 2008



## Exercise 3.2 Solution

Trace:

```
onviso:trace([elevator, closed, '_'],  
[[[{$1,$2},{ $3,'_'}],[{:=,'at_floor',$1},{>=,'$2',M},{:=,'$3',N}],[]]],['elevator@aniko-laptop'],{all,[call]}).
```

Merge:

```
> BeginFun = fun(_InitData) -> {ok,0} end.  
  
> WorkFun = fun(_Node, _Trace, _PidMapping, Count) ->  
{ok, Count + 1} end.  
  
> EndFun = fun(Count) -> io:format("The clients sent ~p messages.\n", [Count])  
end.  
  
> onviso:merge(1, BeginFun, WorkFun, EndFun).
```

Copyright 2008



## Exercise 4

Try to set and use any of the above traces with the command line interface

The command line interface can be started by cli:start().

Copyright 2008

