Profiling Erlang Programs using Percept2

Huiqing Li and Simon Thompson
Erlang Factory Lite, London
8 November 2012
Overview

- Erlang multicore … RELEASE
- Percept … and Percept2
- Case study from Wrangler
- Conclusions
Erlang multicore
• Scalable distributed Erlang: systems and tools.
• Groups and implicit placement.
Tools

How to use these systems effectively?
Existing tools

• Erlang tracing
• Htop
• DTrace
• Percept
Percept
Percept

- **Percept**: **Erlang concurrency profiling tool.**
- In the Erlang/OTP distribution …
- … written by Björn-Egil Dahlberg.
- Monitors process runnability using `erlang:trace/3, erlang:system_profile/2`.
- Offline tool … ‘replay’ a computation.
Percept facilities

• Histogram of active processes vs. time.
• ‘Drill down’ to process information via Pid.
• Start/end time, parent/child processes, time spent waiting for messages, … .
• View individual process runnability.
Percept
Percept to Percept2

- Concurrency, but multicore is ‘invisible’.
  - Run queue migration,
- Analysis
  - Runnable vs running.
- Presentation of information
  - Clarify / colour.
Percept to Percept2

- Parallelisation
  - Dynamic callgraph information.
- Scalability of the system.
  - Usability (process tree)
  - Parallel processing
Percept2
Functionality
Scheduler activity

The number of schedulers active at any time
Process info tree

Process info presented in expandable tree format.
Process migration

<table>
<thead>
<tr>
<th>Pid</th>
<th>&lt;0.341.0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>undefined</td>
</tr>
<tr>
<td>Entrypoint</td>
<td>sim_code_v3:init_ast_loop/1</td>
</tr>
<tr>
<td>Arguments</td>
<td>[&lt;0.340.0&gt;]</td>
</tr>
<tr>
<td>Timetable</td>
<td>Start{1352,219904,767820} 0.0163 Stop undefined 8.0810</td>
</tr>
<tr>
<td>Parent</td>
<td>&lt;0.37.0&gt;</td>
</tr>
<tr>
<td>Children</td>
<td></td>
</tr>
<tr>
<td>RQ_history</td>
<td>[1,4,1,3,1,2,1,2,1,3,1,4,3,2,1,2,1,3,1,2,1,2,1]</td>
</tr>
<tr>
<td>#msg_received, avg_msg_size</td>
<td>{217,7280}</td>
</tr>
<tr>
<td>#msg_sent, avg_msg_size</td>
<td>{216,167}</td>
</tr>
<tr>
<td>accumulated runtime (in milliseconds)</td>
<td>1678</td>
</tr>
<tr>
<td>Callgraph/time</td>
<td>show call_graph/time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>percentage of total waiting time</th>
<th>total mean stddev #recv module:function/arity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x</td>
<td>4.8195 0.6024 0.8377 8 sim_code_v3:ast_loop/1</td>
</tr>
<tr>
<td>2 x</td>
<td>0.1195 0.0011 0.0041 108 sim_code_v3:insert_hash/2</td>
</tr>
</tbody>
</table>
Message passing stats

No. of msgs, and average msg size, sent / received.
Information aggregation
Process tree

Parent-child relationships between processes.
Dynamic function info

Call relationships b/w fns during the program run, and number of calls.
## Accumulated runtime

Accumulated time during which a function within this process is in a running state.

<table>
<thead>
<tr>
<th>Pid</th>
<th>&lt;0.037.0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entrypoint</td>
<td>undefined</td>
</tr>
<tr>
<td>M:F/A</td>
<td>sim_code_v3:check_clone_candidates/3</td>
</tr>
<tr>
<td>Call count</td>
<td>1</td>
</tr>
<tr>
<td>Accumulated time (in milliseconds)</td>
<td>1873</td>
</tr>
</tbody>
</table>

### Callers

```
module:function/arity call count
sim_code_v3: sim_code_detection 1/6 1
```

### Called

```
module:function/arity call count
sim_code_v3: start_clone_check_process/0 1
sim_code_v3: examine_clone_candidates/4 1
sim_code_v3: combine_clones_by_au/1 1
sim_code_v3: stop_clone_check_process/1 1
sim_code_v3: check_clone_candidates/3-1c$s^0/1-0-'/1 1
```
Process activity
Scalability
Compression
Parallelise

- Parallelise Percept itself.
- Multiple log files …
- … that can be processed in parallel …
- … and integrated into a single result.
Distribution
Case study
Clone detection in Wrangler

- parse files
- flatten trees
- identify candidates
- check candidates
Active Processes
Processes
Process activity
Critical process
### Active functions

<table>
<thead>
<tr>
<th>pid</th>
<th>module:function/arity</th>
<th>active period</th>
<th>call count</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:sim code detection/8</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:sim code detection/4</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:sim code detection 1/6</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash ast/5</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash ast/5-lc$s^0/1-0-0$'/'</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash file ast 1/6</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash file ast 1/6-fun-0-0'/'</td>
<td>379</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash file ast 1/6-fun-1-0'/'</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash function ast/5</td>
<td>245</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash function ast/5-fun/1</td>
<td>245</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash function ast/5-fun/1'/'</td>
<td>245</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:absolute to relative loc/2</td>
<td>245</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:absolute to relative loc/2-fun/0-0'/'</td>
<td>12072</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:abs to relative loc/2-lc$s^0/1-0-0$'/'</td>
<td>12027</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:abs to relative loc in ann/2</td>
<td>80446</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:abs to relative loc in ann/2-lc$s^0/1-0-0$'/'</td>
<td>12027</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:abs to relative loc in ann/2-lc$s^1/1-1-1$'/'</td>
<td>12027</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:to relative/2</td>
<td>57209</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:to relative/2-fun-0-0'/'</td>
<td>11782</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:abs to relative loc in ann/2-lc$s^2/1-2-2$'/'</td>
<td>3418</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash function ast/1/7-fun-0-0'/'</td>
<td>12027</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:generalise and hash body/6</td>
<td>899</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:insert to ast tab/2</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:gen initial clone candidates/3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>&lt;0.37.0&gt;</td>
<td>sim code v0:get clone candidates/3</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Callgraph timeslice
Refactoring #1

\[
\text{[generalise_and_hash_file_ast_1(}
\text{File, Threshold, ASTPid, true, SearchPaths, TabWidth)}
\text{||File<-Files].}
\]

\[
\text{para_lib:pmap(}
\text{fun(File) ->}
\text{generalise_and_hash_file_ast_1(}
\text{File, Threshold, ASTPid, true, SearchPaths, TabWidth)}
\text{end, Files).}
\]
Active Processes
Processes
Refactoring #2

lists:foreach(fun (Form) -> F(Form) end, Forms).

para_lib:pforeach(fun (Form) -> F(Form) end, Forms, 5).
Active Processes
Processes
Callgraph
Refactoring #3

examine_clone_candidates([],_Thresholds,CloneCheckerPid,_Num) ->
    get_final_clone_classes(CloneCheckerPid);

examine_clone_candidates([C|Cs],Thresholds,CloneCheckerPid,Num) ->
    output_progress_msg(Num),
    C1 = get_clone_in_range(C),
    NewClones=examine_a_clone_candidate(C1,Thresholds),
    add_new_clones(CloneCheckerPid,{C1, NewClones}),
    examine_clone_candidates(Cs,Thresholds,CloneCheckerPid,Num+1).

examine_clone_candidates(Cs, Thresholds, CloneCheckerPid) ->
    NumberedCs = lists:zip(Cs, lists:seq(1, length(Cs))),
    para_lib:pforeach( 
        fun({C, Nth}) ->
            examine_a_clone_candidate( 
                {C,Nth},Thresholds,CloneCheckerPid)
            end,NumberedCs),
        get_final_clone_classes(CloneCheckerPid).

examine_a_clone_candidate({C,_Nth},Thresholds,CloneCheckerPid) ->
    output_progress_msg(Nth),
    C1 = get_clone_in_range(C),
    NewClones=examine_a_clone_candidate(C1,Thresholds),
    add_new_clones(CloneCheckerPid,{C1,NewClones}).
Active Processes
Parallel clone detection in Wrangler
Next steps

• Looking at the front end … HTML5 version.
• Online SD Erlang Profiling
• Refactoring Tools
• Dialyzer parallelisation at Uppsala …
• … paper in Erlang Workshop 2012.
• BenchErl: benchmarking
• Extended tracing in Erlang and DTrace.
Conclusions

• RELEASE project.
• Tool to support scalable parallelisation.
• SD Erlang release and tools in 2013.
• Thanks to the percept team!
Get involved!

git://github.com/huiqing/percept2.git