LATEST NEWS FROM THE ERLANG/OTP TEAM

KENNETH LUNDIN

ERLANG FACTORY LONDON JUNE 11, 2010

POSITIVE REFELXIONS

- The positive effect of being on GitHub continues
  - More user involvement
  - Example: 51 number of contributions by 32 contributors (from Nov 25 to Feb 24)

The use of Erlang is really taking off!

ERLANG ALL OVER THE WORLD

RELEASE PLANS 2010

- Next release is a new major release (R14)

June 16: R14A, a beta release

Sept 01: R14B first drop for commercial use

Service releases R14B01, 02, 03 etc. with ~2-3 months intervals

MORE DETAILS ABOUT R14
NEW FEATURES

Search in binaries (as of EEP-31)

new module called **`binary`** with functions:

```
match, matches, split, replace,
 longest_common_prefix, ... part, at, copy,
 first, last
```

Optimization of receive for common special case

- reveive statements that can only match a newly created reference are now specially optimized
- will execute in constant time regardless of the number of messages in the receive queue for the process. That optimization will benefit calls to `gen_server:call()`. (See gen:do_call/4 for an example of a receive statement that will be optimized.)

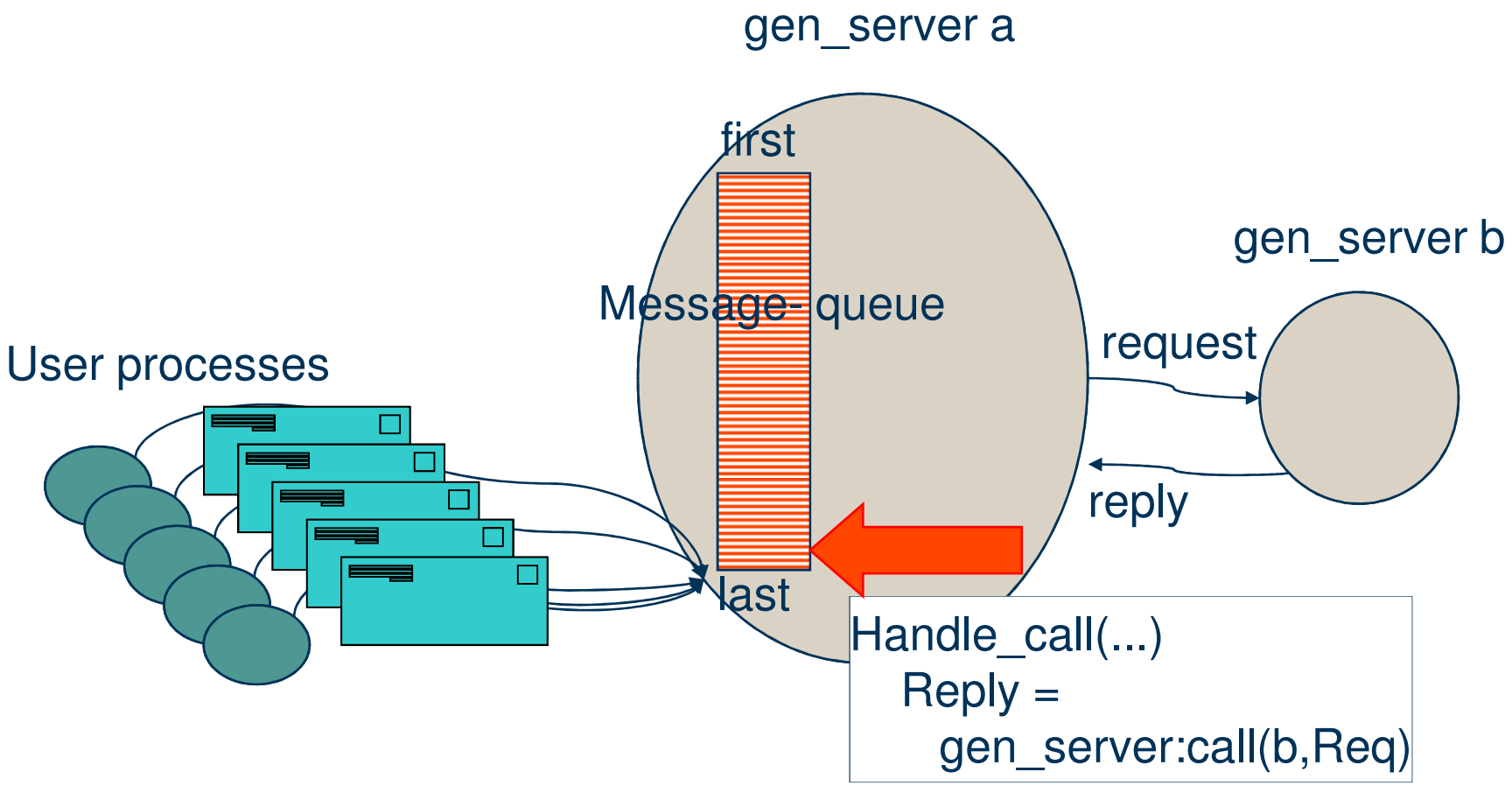
MORE DETAILS ABOUT R14

NEW FEATURES

```
Mref = erlang:monitor(process, Process) of
erlang:send(Process, {Label, {self(), Mref},
  Request},[noconnect]),
receive
    {Mref, Reply} ->
        erlang:(Mref, [flush]),
        {ok, Reply};
    {'DOWN', Mref, _, _, noconnection} ->
         exit({nodedown, Node});
    {'DOWN', Mref, _, _, Reason} ->
         exit(Reason)
     after Timeout ->
         erlang:demonitor(Mref),
         …
```

MORE DETAILS ABOUT R14

NEW FEATURES



MORE DETAILS ABOUT R14

NEW FEATURES

Improvements regarding NIFs
Native Implemented Functions written in C

- Send messages from a NIF, or from thread created by NIF, to any local process (`enif_send`)
- Store terms between NIF calls (`enif_alloc_env, enif_make_copy`)
- Create binary terms with user defined memory management (`enif_make_resource_binary`)
- And some incompatible changes made to the API. For more information see the warning text in erl_nif(3).
- **crypto** application as NIFs, was previously implemented as a driver.

New SSL ready to replace old SSL

- Is built in pure Erlang except for the encryption routines which are from OpenSSL via NIFs in the crypto module.
- ”new” SSL will be the default.
- ”old” SSL will be around until next major release (R15)

Major improvements in Erlang profiler `eprof`

- ~5-84 times faster than before
- Only ~0-5 times slower when profiling with `eprof` than running with full speed.
- Scales over multiple schedulers which it did not do before
- **`cprof`** is also significantly faster and does also scale

MORE DETAILS ABOUT R14

POTENTIAL INCOMPATIBILITIES

- -define(MACRO,m) will require closing parenthesis

- call to a local function with same name as an **auto-imported** BIF will call the local function not the BIF as it is today. A warning will be issued

-
```
-module(m).
-export([foo/0]).
foo() ->
    binary_to_list(<<10,20>>).
binary_list(Bin) ->
     ....
```

- `erlang:max/2, erlang:min/2 and binary_to_term/2,` are now autoimported

MORE DETAILS ABOUT R14 (R14B AND LATER)

- Tentative
  - Parameterized modules officially supported and with more efficient implementation.
- Half-word 64-bit Erlang VM
  - 4 Gbytes process heaps (in total)
  - max size of Erlang term 4 Gbytes
  - ets tables and binaries in separate space can utilize the full 64 bit address space
- Misc
  - Change from Erlang Public License to something more well known. It takes a long time for decisions like this! Work under progress

MORE DETAILS ABOUT R14 (R14B AND LATER)

- Multi-core performance improvements
  - optimized rwlocks
  - delayed deallocation
  - ”lock-free” process table
- **`-type`**, **`-spec`** officially supported
- edoc with support for **–type/-spec**

LONGER TERM PLANS

- More multi core performance improvements
  - lock free pre-allocators (thread specific pre allocated buffers)
  - Scheduler specific `mseg_alloc` reducing lock contention and necessary for future NUMA optimizations. Intel Nehalem, AMD Opteron

- Clustered shared heap or other solution to allow parallell computing on large sets of data avoiding copying.

- New XML-schema/dtd validator complementing the XML SAX parser we already have.
- SMP optimizations in existing applications (Mnesia, ASN.1 ...)

CONTRIBUTIONS ARE WELCOME

- JSON encode/decode as a NIFs (based on EEP-18, http://www.erlang.org/eeps/eep-0018.html)

- Suggestions and implementations for better layout and search in documentation

- A replacement for DETS which is better and can handle data > 4 Gbyte

ERICSSON