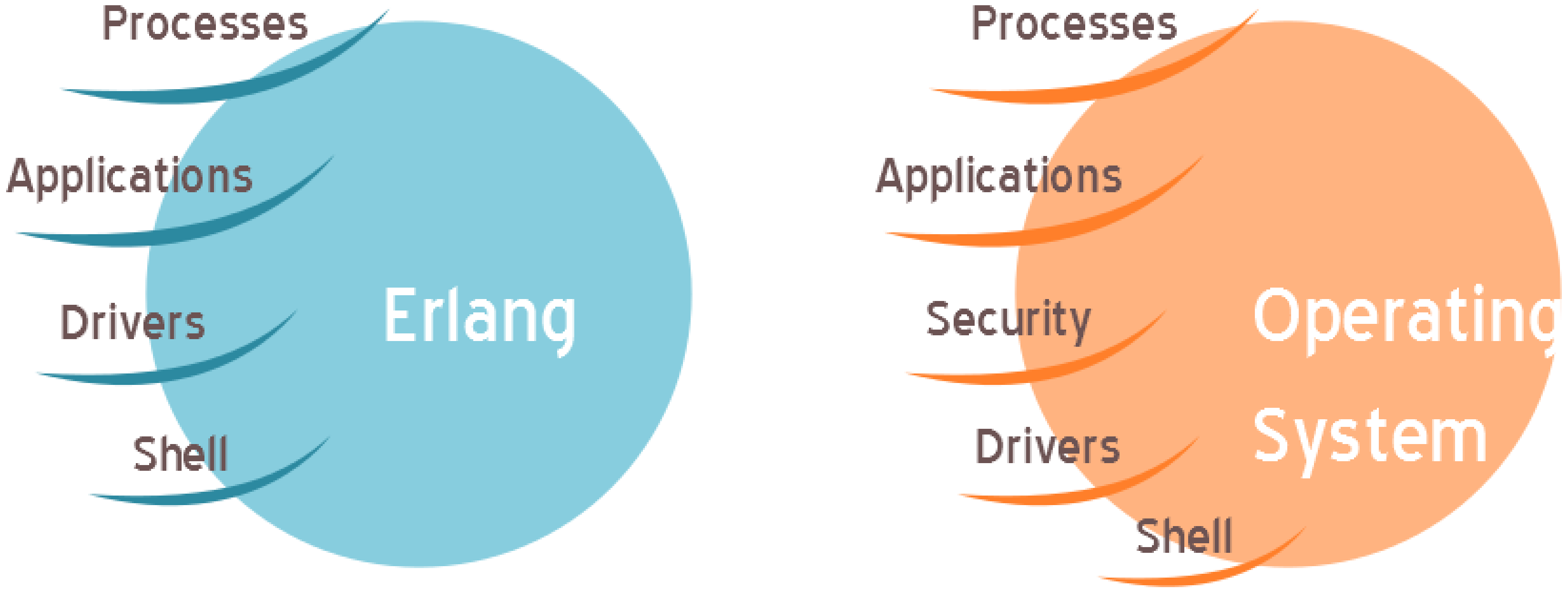
Erlang on Xen

A quest to lower startup latency

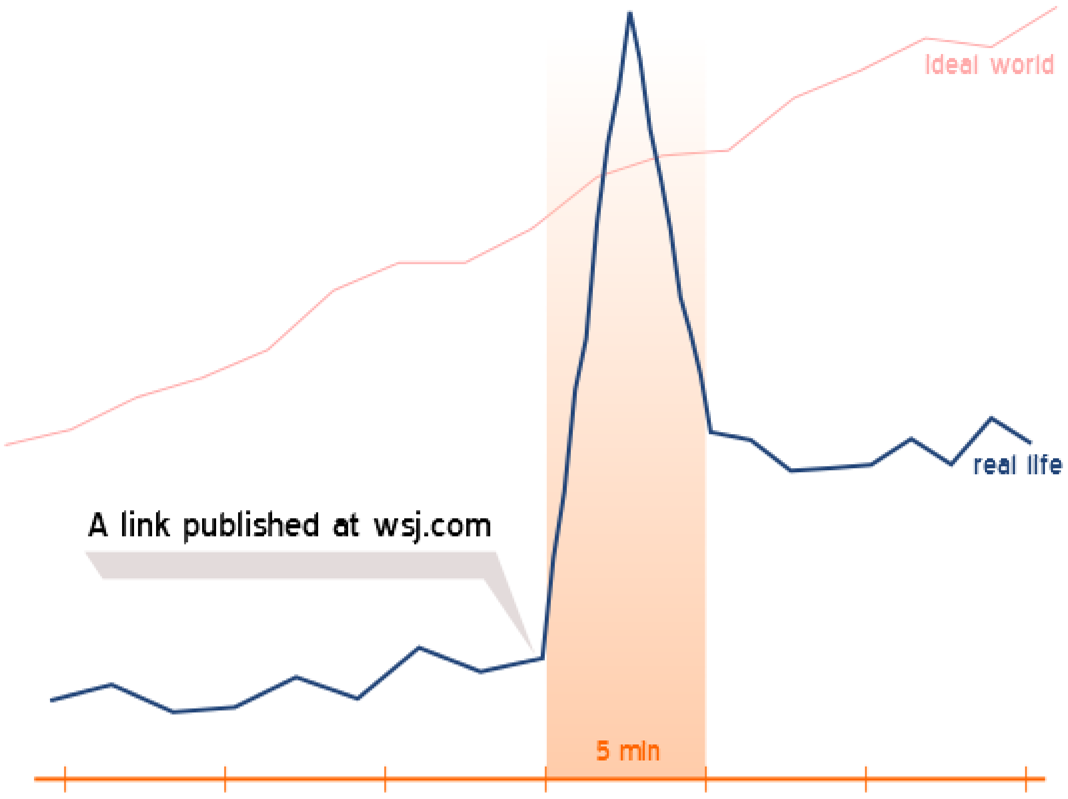
Erlang Factory SF Bay Area 2012

Erlang as OS



Erlang duplicates many OS features in user space – it even dumps core when crashes

“Slashdot Effect”



A traffic grows in spikes, so sharp that they may even go unnoticed by monitoring

Broken Promises

- The greatest promise of rented IT infrastructure: start small and grow as needed – is unfulfilled

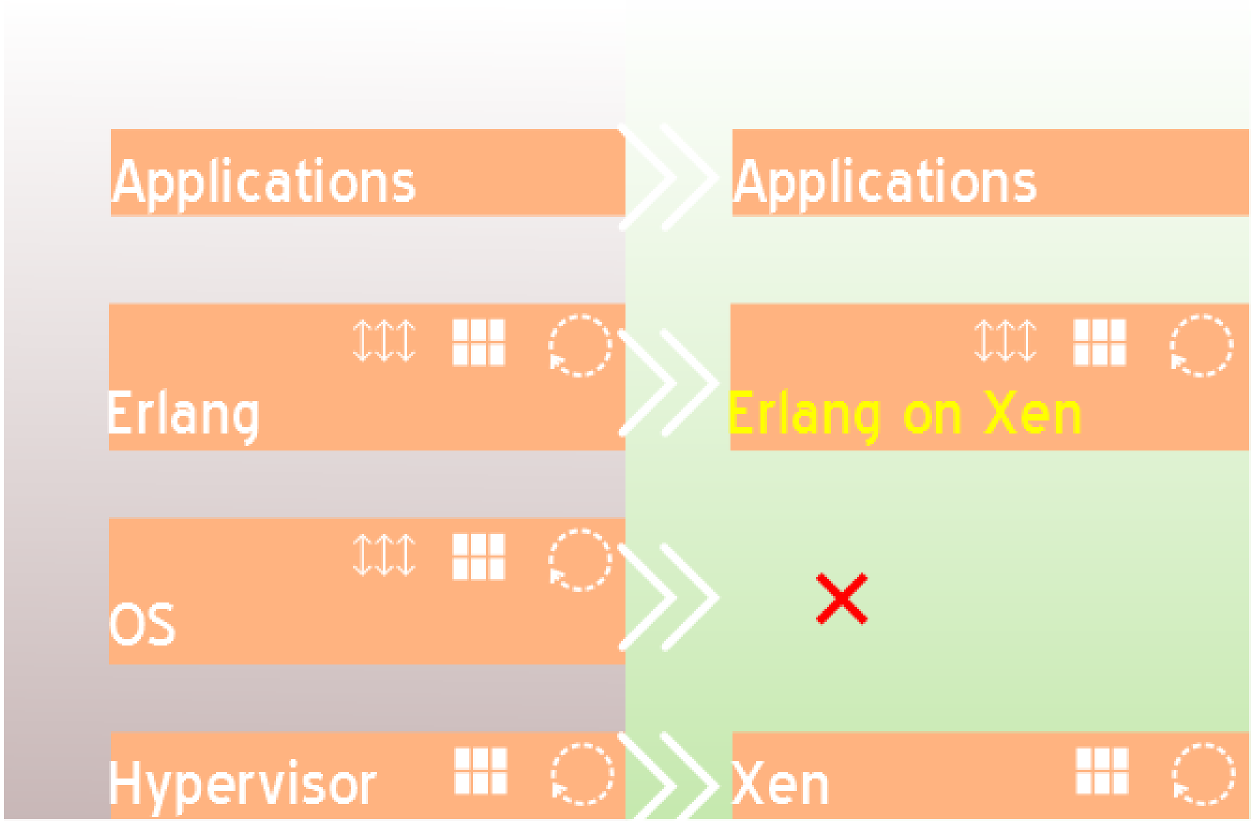
- “Slashdot effect” make it impossible to provision new computing nodes in real time

- In practice options are limited:
  - Have a x10-x100 redundancy upfront
  - Restrict dynamic content
  - Built a scalable cloud in house

Internet quotes about EC2: *“If a new server is needed, it takes around 40 seconds to start up.”*

“The time before you can log in to a Windows instance – 40min”

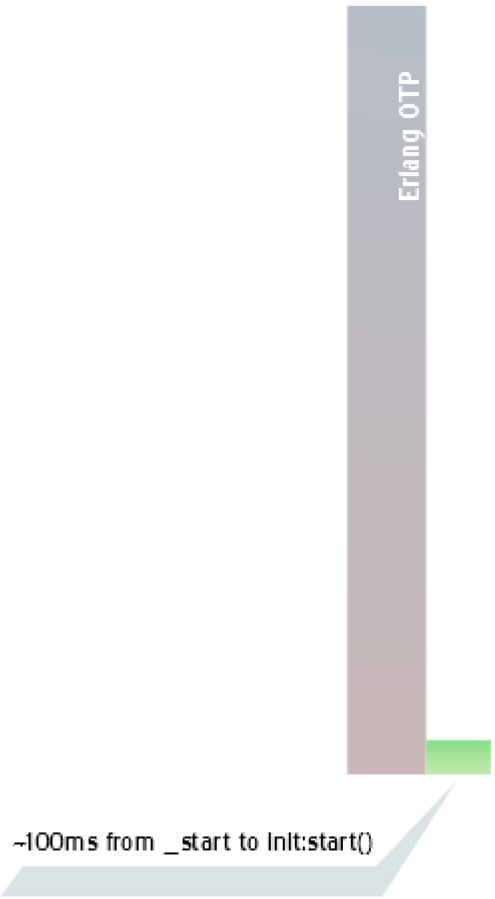
Enter Erlang on Xen



Running Erlang as a Xen guest removes a redundant OS layer

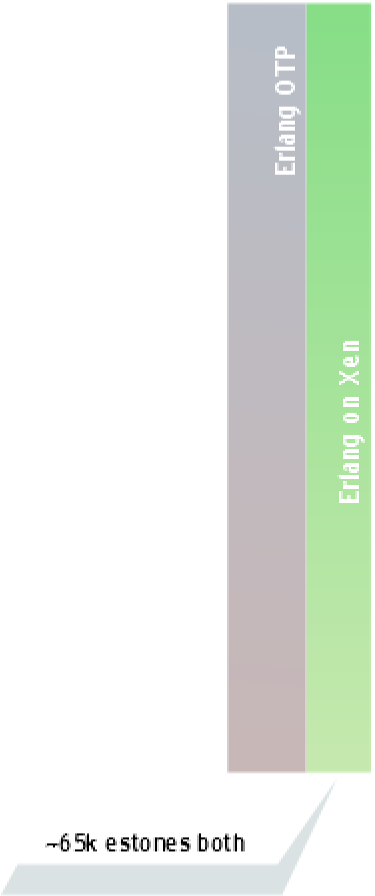
Preliminary Results

Startup Latency



~100ms from _start to init:start()

Performance



~65k estones both

Startup Latency

- 80% - page table setup
  - the spot to optimize – delay page table setup
- 15% - network driver initialization
  - speedier initialization unlikely
- 5% - Erlang-related initialization (hashing atom and export table, etc)
  - further optimization not justified

Emulator Tests

6 test suites done

- 🟩 lists_SUITE
- 🟩 tuple_SUITE
- 🟩 list_bif_SUITE
- 🟥 binary_SUITE (5 skipped, 3 failed)
- 🟩 bs_bincomp_SUITE (1 skipped)
- 🟩 bs_bit_binaries_SUITE (1 skipped)

13 test suites to go

- bs_construct_SUITE
- bs_match_bin_SUITE
- bs_match_int_SUITE
- bs_match_misc_SUITE
- bs_match_tail_SUITE
- bs_utf_SUITE
- big_SUITE
- exception_SUITE
- float_SUITE
- fun_SUITE
- guard_SUITE
- num_bif_SUITE
- ref_SUITE

🟩 All Ok

🟥 Partially failed

Under the Hood

- No code transformation – preloaded modules are ready to go (almost)
- Dynamic instruction specialization – derived from frequency analysis
- No external library dependencies (almost)
- No TCP/IP stack and no block device drivers
- 32-bit x86 only
- ~35K SLOC of C, 7K SLOC of Erlang - no code borrowed from Erlang/OTP

The Vision

“A super-elastic computing fabric for Erlang”:

A shared Erlang-application infrastructure capable of provisioning computing nodes in real time, after the client application receives a request; a platform that may scale a client application to 1000s of computing nodes within a second and remain profitable charging clients 1/10th of today’s rates.

Maxim Kharchenko
maxim.kharchenko@gmail.com