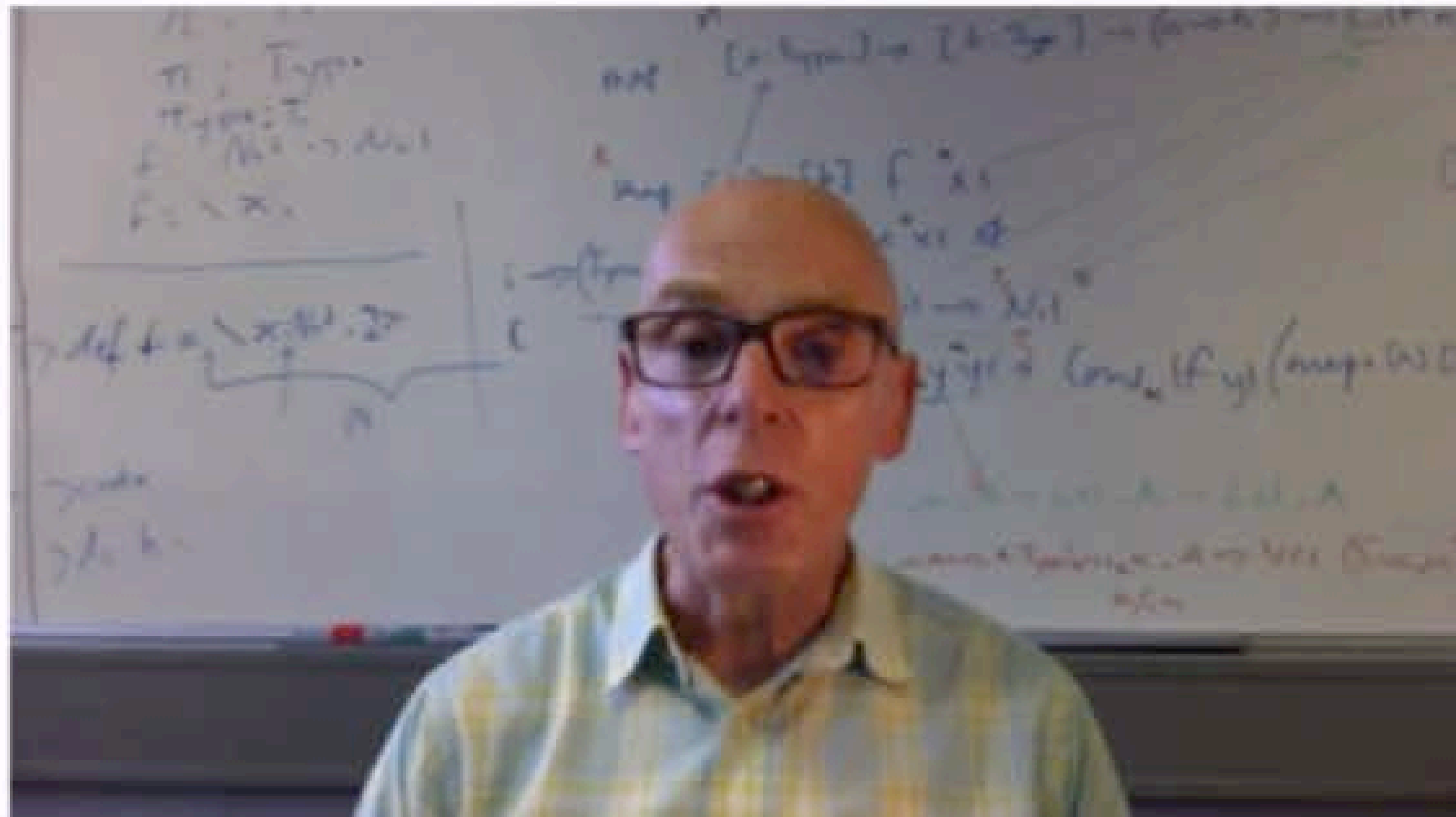
University of Kent

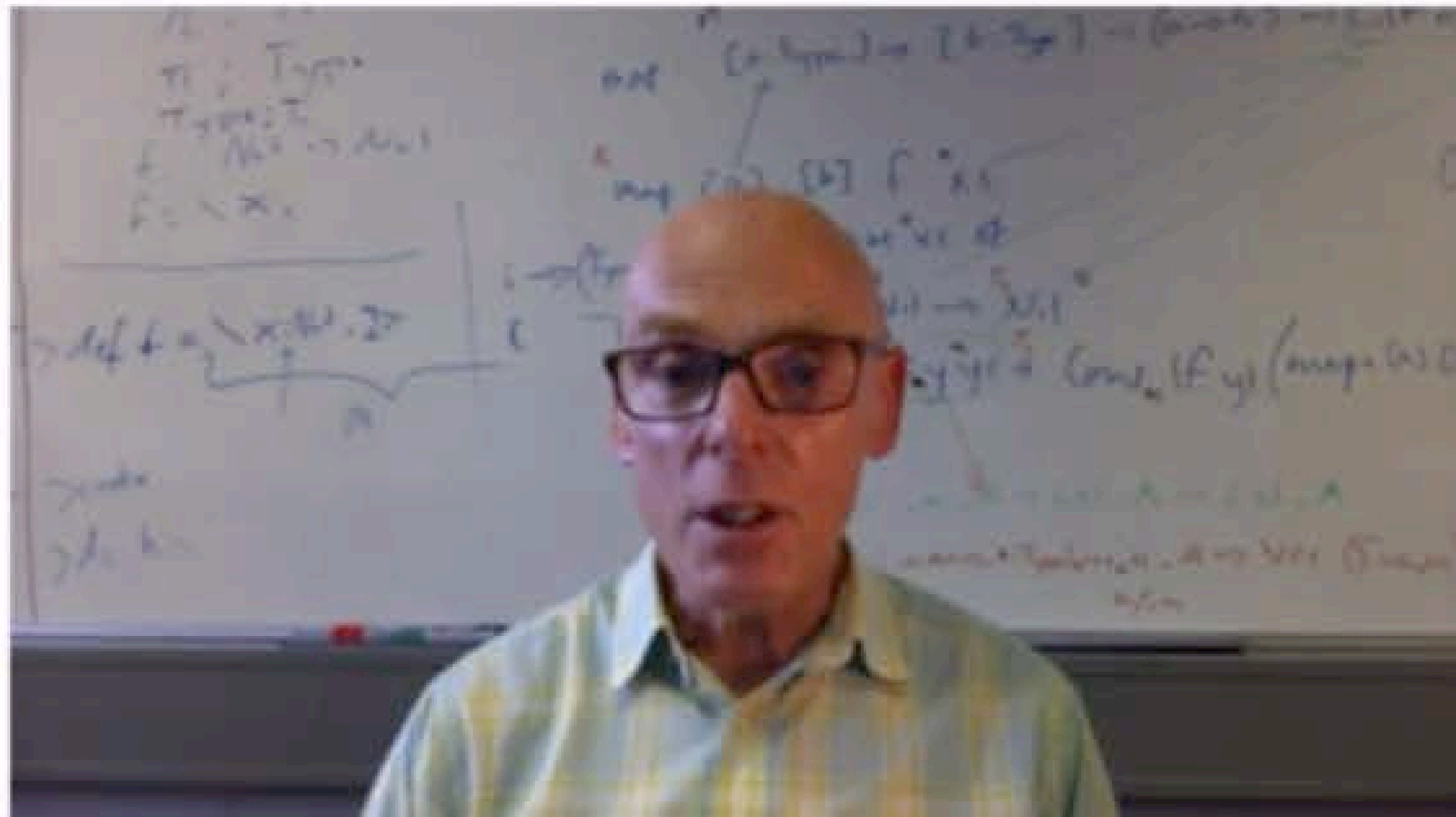
What can go wrong?



Concurrency in practice

On a single processing element, concurrent processes timeshare, mediated by a scheduler.

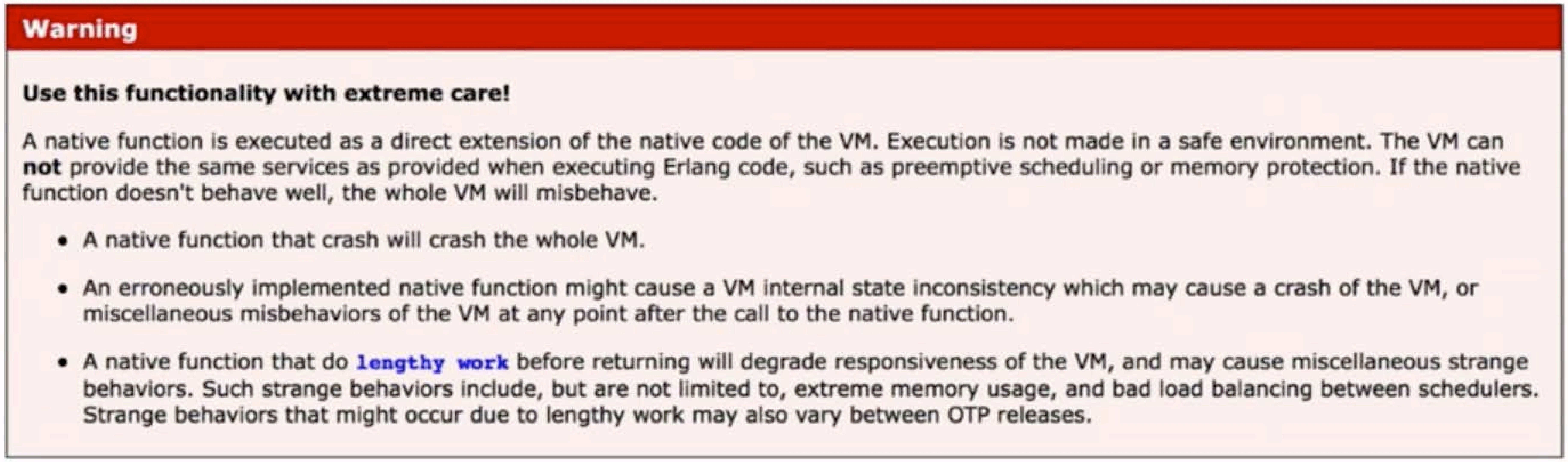
On a multicore processor, there is, potentially, concurrent activity on each core, with each scheduled separately.



Keeping things fair

The scheduler ensures that all runnable processes get to run: count reduction steps.

Built-in functions (BIF) or native implemented functions (NIF) could prevent this …



Warning

Use this functionality with extreme care!

A native function is executed as a direct extension of the native code of the VM. Execution is not made in a safe environment. The VM can **not** provide the same services as provided when executing Erlang code, such as preemptive scheduling or memory protection. If the native function doesn't behave well, the whole VM will misbehave.

- A native function that crash will crash the whole VM.
- An erroneously implemented native function might cause a VM internal state inconsistency which may cause a crash of the VM, or miscellaneous misbehaviors of the VM at any point after the call to the native function.
- A native function that do `lengthy work` before returning will degrade responsiveness of the VM, and may cause miscellaneous strange behaviors. Such strange behaviors include, but are not limited to, extreme memory usage, and bad load balancing between schedulers. Strange behaviors that might occur due to lengthy work may also vary between OTP releases.

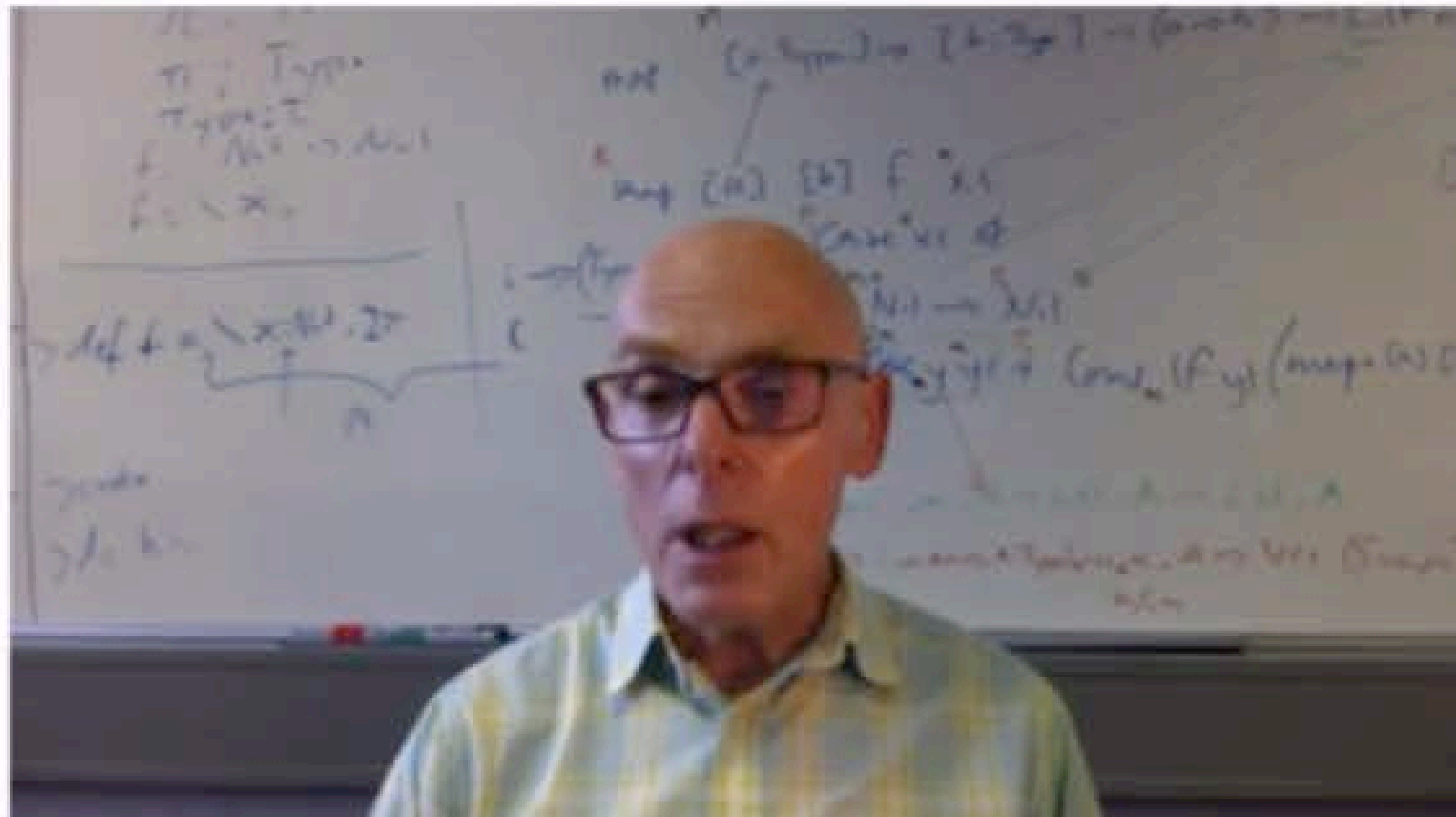
Warning from http://erlang.org/doc/man/erl_nif.html

Deadlock

When we discussed scheduling we said

> "A process is *waiting* if it is in a `receive` statement and waiting for a message to arrive in the mailbox …"

A system is *deadlocked* if no process is runnable: all are waiting in `receive` statements for a message to be delivered.



Dining Philosophers

http://soarlab.org/people/

Race conditions

No guarantees about ordering (except point to point).

Move from a purely sequential runtime to a concurrent one.

http://www.javacreed.com/wp-content/uploads/2014/11/What-is-Race-Condition-and-How-to-Prevent-It.png

Filling the mailbox

Are you processing all the incoming messages in a `receive`?

If not, then they will accumulate in the mailbox …

http://www.tapsmart.com/wp-cotent/uploads/2014/10/Full-open-mailbox.jpg

Sequentiality?

Are you getting all the concurrency you can from your system …

… or is it too sequential?

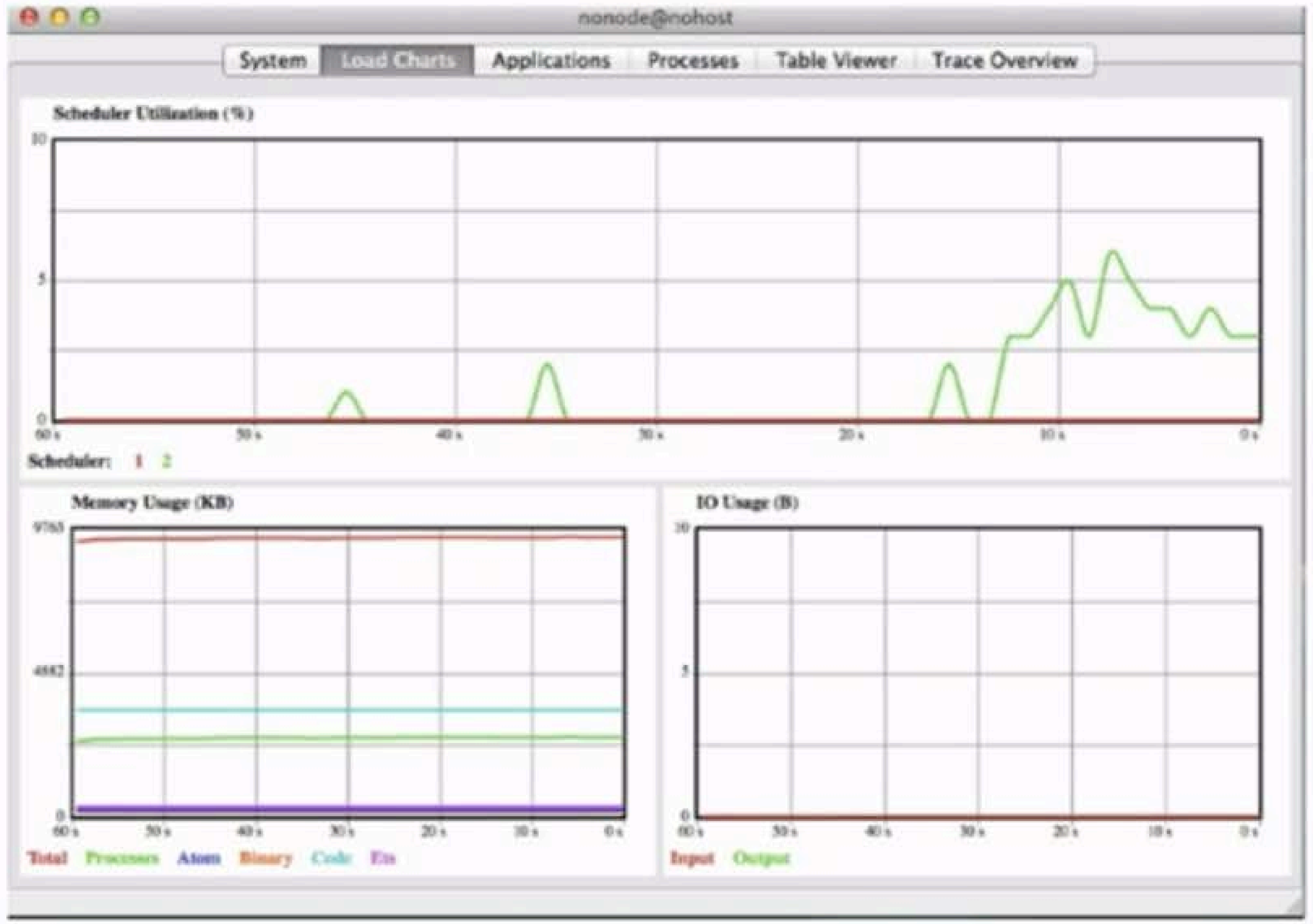
http://img10.deviantart.net/81c8/i/2006/146/e/d/pitching_sequence_by_gr8koolaid.jpg

Erlang Observer

Graphical tool for observing the characteristics of Erlang systems.

Observer displays system information, application supervisor trees, process information, ETS tables, Mnesia tables.

Contains a front end for Erlang tracing.



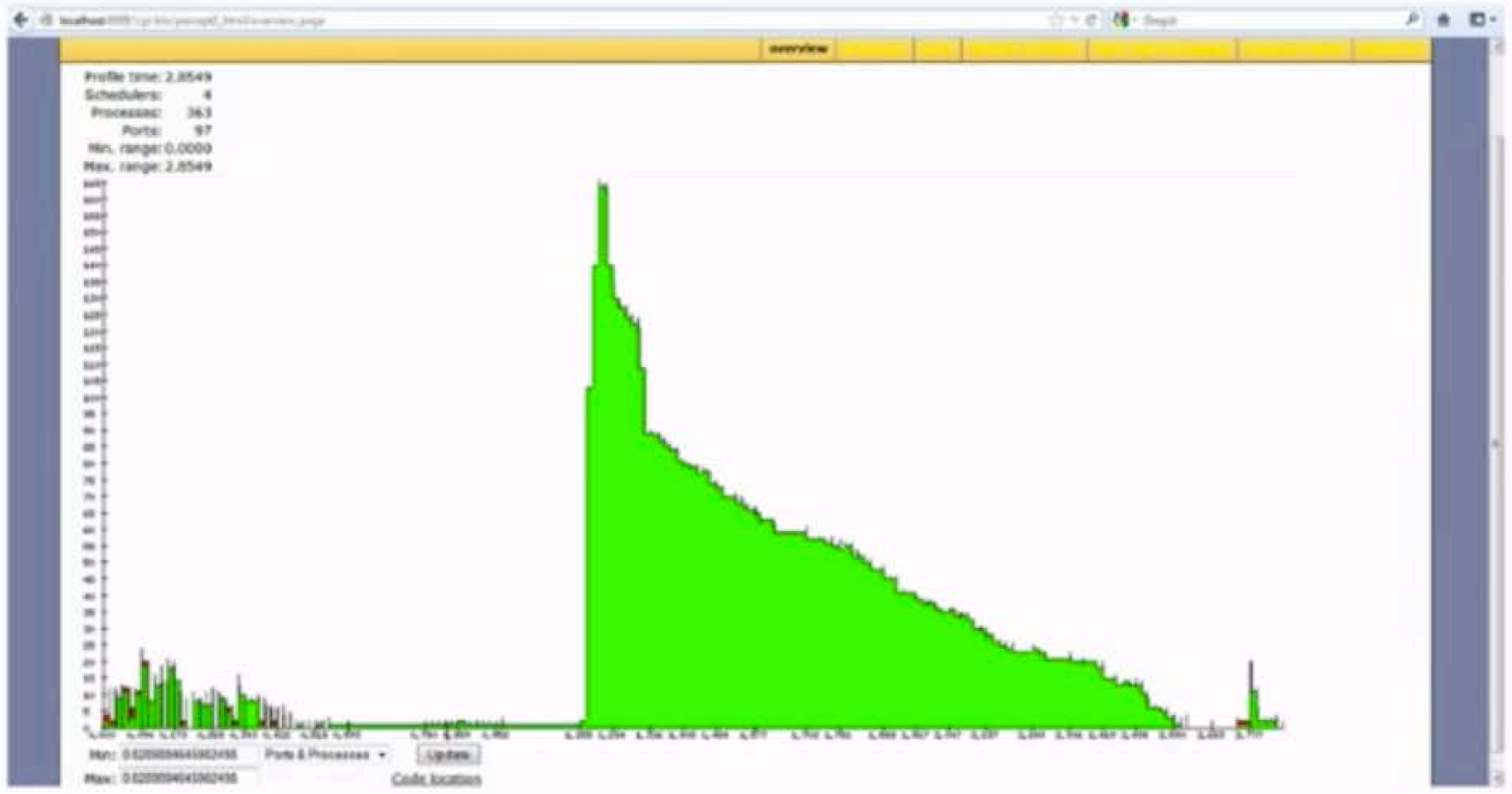
http://benjamintan.io/images/observer_2.png

http://erlang.org/doc/apps/observer/observer_ug.html

Percept2

Concurrent profiling tool for Erlang that enhances Percept

Active processes, runnable or running, process trees, code, …



```
Profile time: 2.8549
 Schedulers:      4
  Processes:    363
      Ports:     97
 Min. range: 0.0000
 Max. range: 2.8549
```

https://github.com/RefactoringTools/percept2

Conqueror and QuickCheck + Pulse

Concuerror is a tool for systematically testing Erlang programs.

It is used to detect errors that only occur on few, specific schedules of a program or verify the absence of such errors.

http://concuerror.com

QuickCheck provides random testing of Erlang systems.

With PULSE it supports repeatable schedules, so that it is possible to "shrink" and replay for race condition examples.

DOI: 10.1145/1596550.1596574

University of Kent