

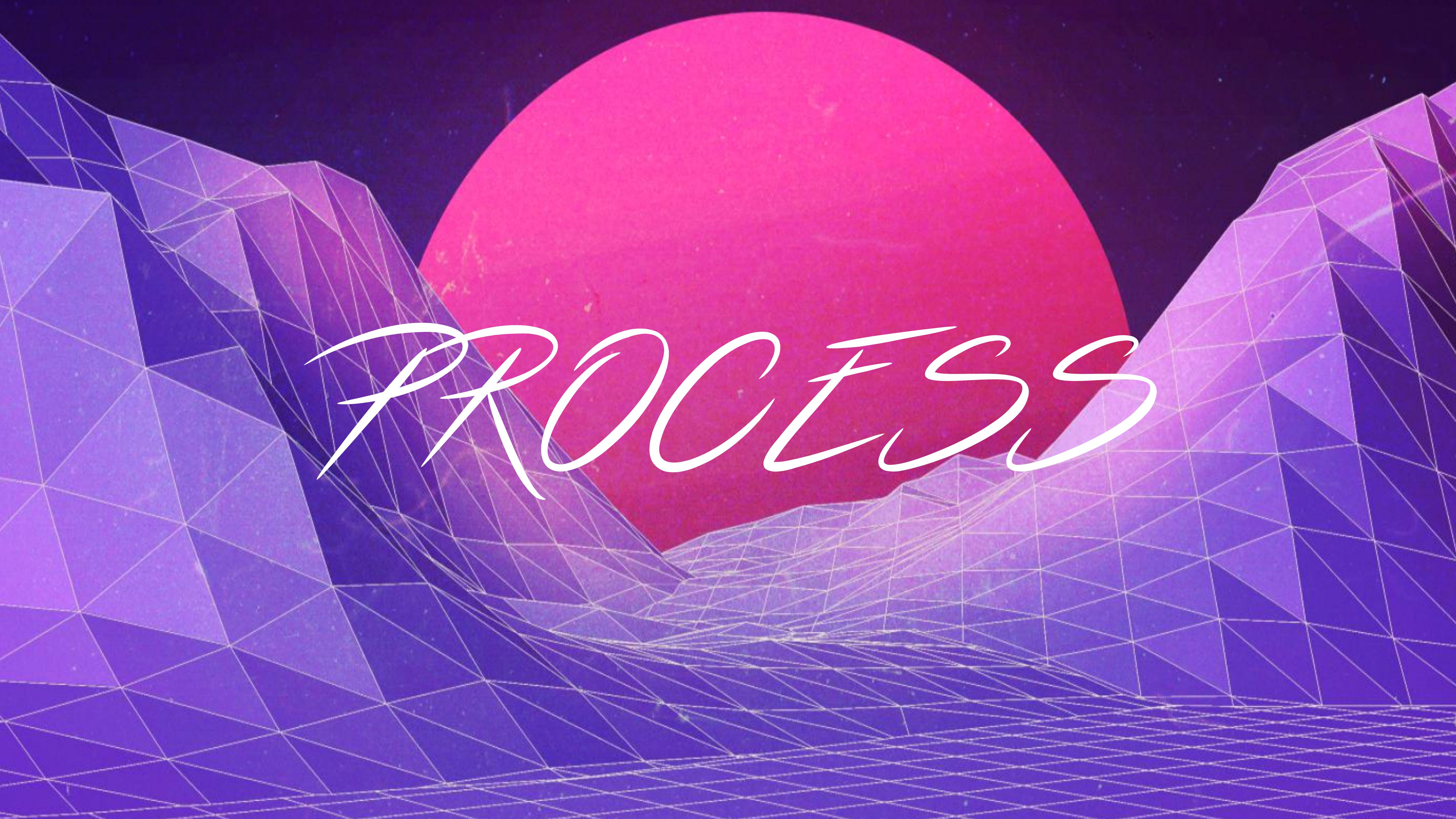
THAT ONE,  
FOR NOW

@bgmarx





# PHOENIX REQUEST

The background features a large, solid red circle centered in the frame. It is surrounded by several translucent, semi-transparent purple shapes that have a low-poly, geometric, and slightly faceted appearance. These shapes overlap each other and the red circle, creating a sense of depth and layering. The overall aesthetic is modern and minimalist.

*PROCESS*

*SCHEDULERS*



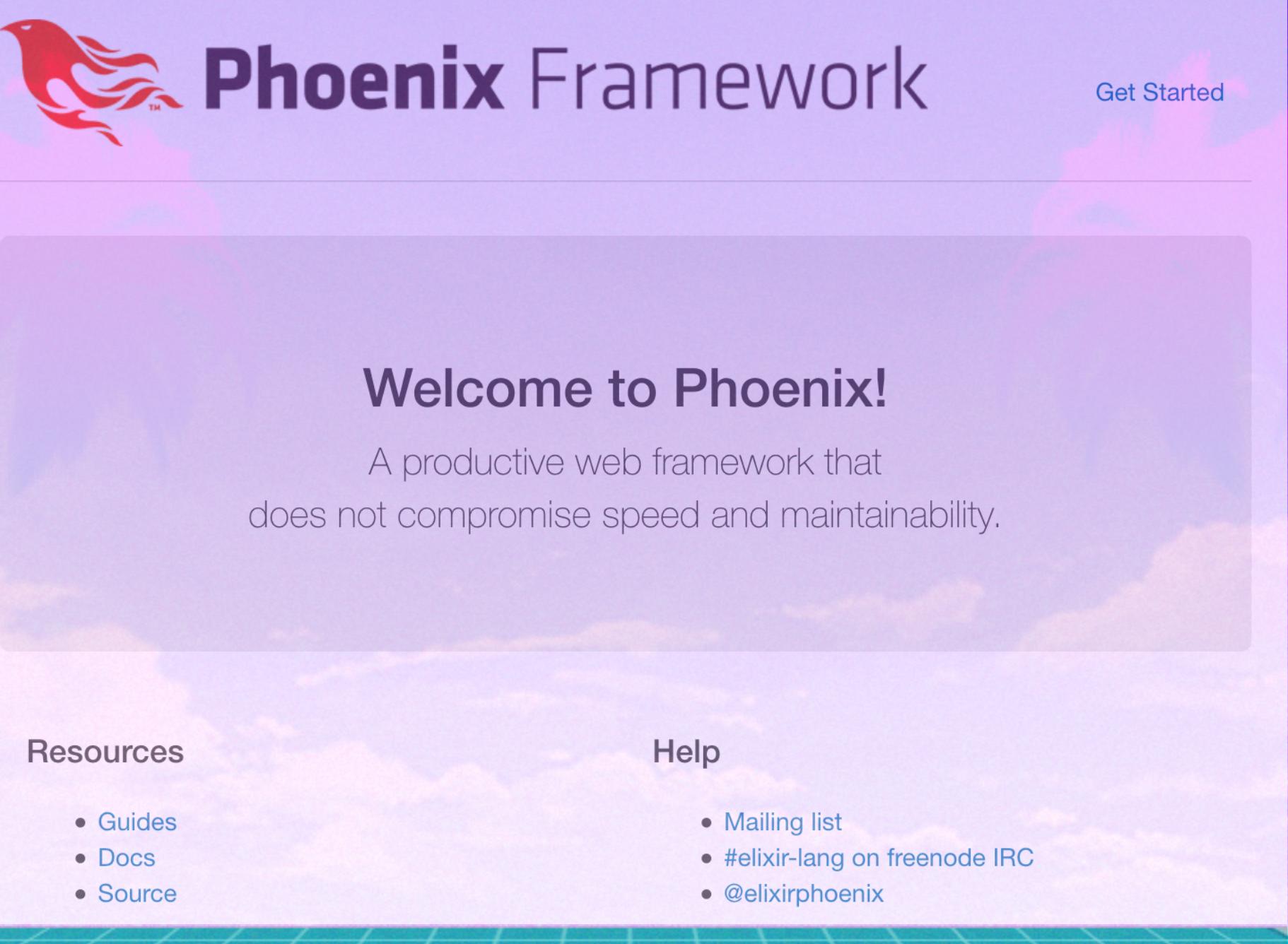
*SINGLE*

*SCHEDULER*

MULTIPLY  
SCHEDULERS  
(SNP)



# PHOENIX REQUEST



The image shows the official Phoenix Framework website landing page. The background features a tropical scene with palm trees and a grid pattern at the bottom. The header includes the Phoenix logo (a red bird icon) and the text "Phoenix Framework". A "Get Started" button is located in the top right corner. The main content area has a large title "Welcome to Phoenix!" followed by a subtitle: "A productive web framework that does not compromise speed and maintainability." Below this are two sections: "Resources" (with links to Guides, Docs, and Source) and "Help" (with links to a Mailing list, #elixir-lang on freenode IRC, and @elixirphoenix on Twitter).

# Phoenix Framework

Get Started

## Welcome to Phoenix!

A productive web framework that does not compromise speed and maintainability.

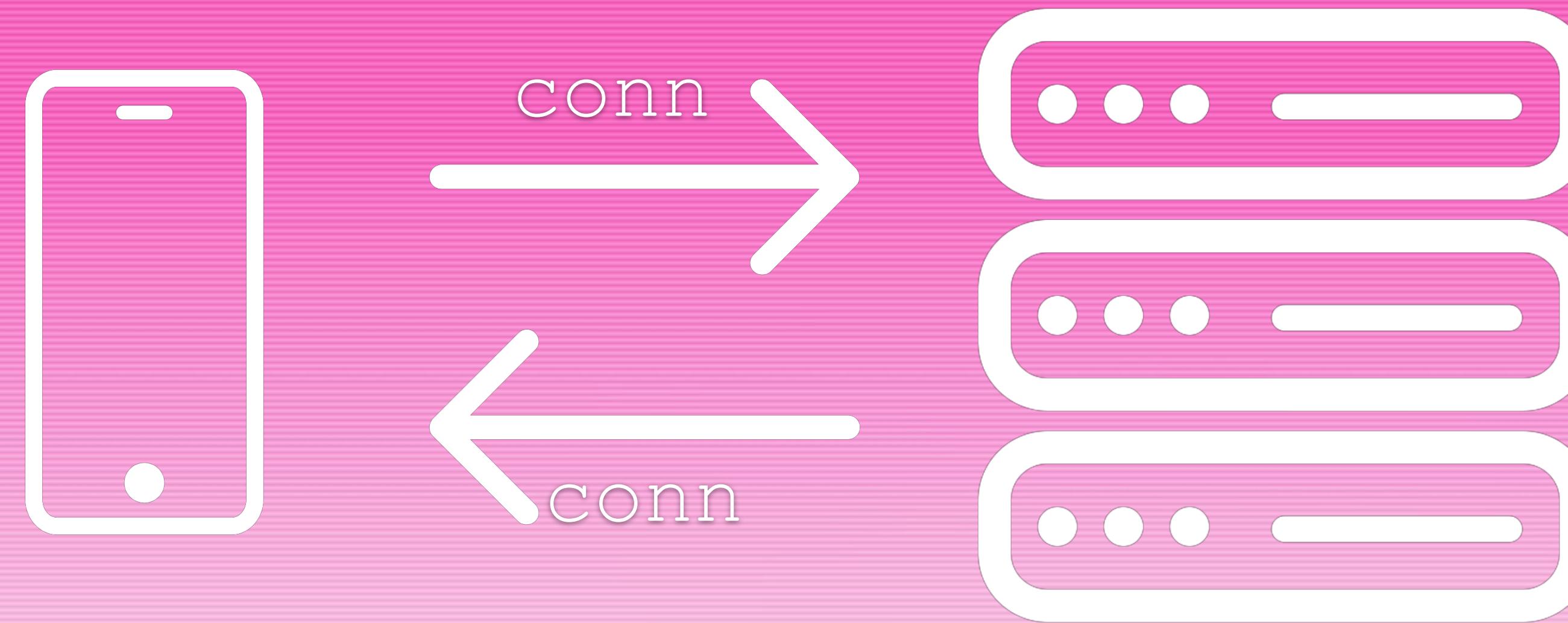
### Resources

- [Guides](#)
- [Docs](#)
- [Source](#)

### Help

- [Mailing list](#)
- [#elixir-lang on freenode IRC](#)
- [@elixirphoenix](#)

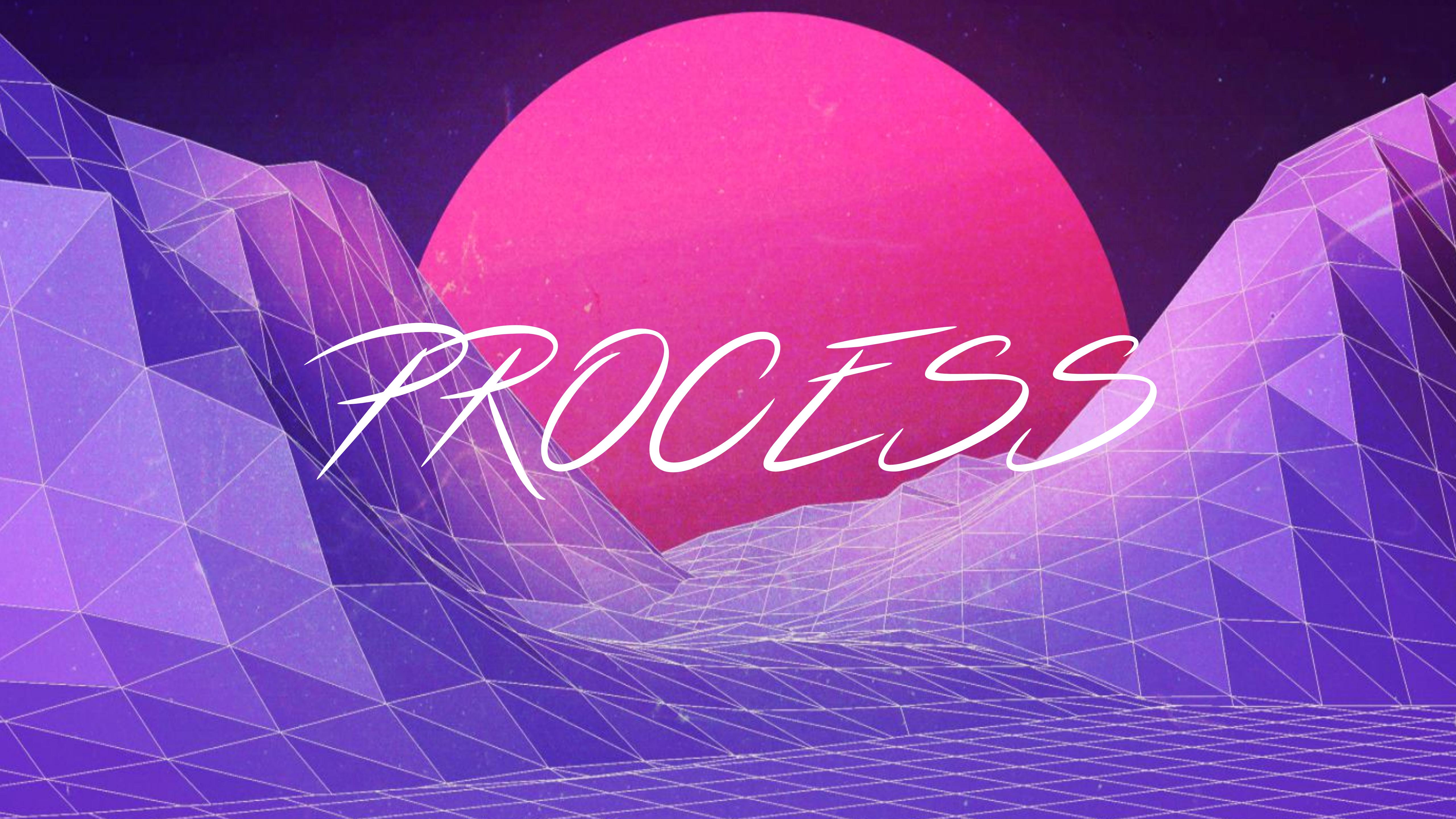
transformations



endpoint  
plugs  
router  
pipeline  
controller  
view  
template



# PHOENIX REQUEST

The background features a large, solid red circle centered in the frame. This circle is partially obscured by several translucent, semi-transparent purple shapes. These shapes include a large, irregular polygon on the left and a smaller, more rounded shape on the right. Both shapes have a subtle, glowing effect and are composed of many thin, white lines forming a mesh-like pattern.

*PROCESS*

# ~~LIGHTWEIGHT~~

## System statistics / limit

Atoms:	22458 / 1048576 (2 % used)
Processes:	248 / 262144 (0 % used)
Ports:	8 / 65536 (0 % used)
ETS:	58 / 8192 (1 % used)
Distribution buffer busy limit:	1048576

# ISOLATED

In simple relativity theory the concept of simultaneity just does not exist. The point is that in reality objects do not share state, I believe its not a good idea to model what cannot exist in reality in software

...

Everything that can be achieved with sharing and locks can also be achieved with pure message passing and no locks. The [sic] is the Erlang way.

EXECUTES  
CODE

# FOUR BLOCKS OF MEMORY

MAILBOX

HEAP

PCB

STACK

# MATZ BOX

Pid	Name or Initial Func	Reds	Memory	MsgQ	Current Function
<0.256.0>	Elixir.Plug.Supervisor	1	7104	0	gen_server:loop/7
<0.241.0>	ranch_sup	1	7104	0	gen_server:loop/7
<0.221.0>	hackney_sup	1	7104	0	gen_server:loop/7
<0.203.0>	Elixir.Logger.Supervisor	1	7104	0	gen_server:loop/7
<0.380.0>	erlang:apply/2	2581	8756	0	timer:sleep/1
<0.145.0>	Elixir.Hex.State	1	8844	0	gen_server:loop/7
<0.202.0>	application_master:start_it/4	1	8888	0	application_master:loop_it/4
<0.394.0>	observer_port_wx:init/1	2	8904	0	wx_object:loop/6
<0.247.0>	cowboy_clock	265	8932	0	gen_server:loop/7

*PROCESS  
CONTROL BOARD*

```
Uint32 rcount; /* suspend count */
int schedule_count; /* Times left to reschedule a low prio
process */

Uint reds; /* No of reductions for this process */
Eterm group_leader; /* Pid in charge (can be boxed) */
Uint flags; /* Trap exit, etc (no trace flags anymore) */
Eterm fvalue; /* Exit & Throw value (failure reason) */
Uint freason; /* Reason for detected failure */
Eterm ftrace; /* Latest exception stack trace dump */
```

# PROCESS STATES

FREE

WAITING

RUNNABLE

RUNNING

EXITING

GARBING

SUSPENDED

garbing computer science



All

Images

Videos

News

Shopping

More

Settings

Tools

About 120,000 results (0.45 seconds)

Did you mean: **garbage** computer science

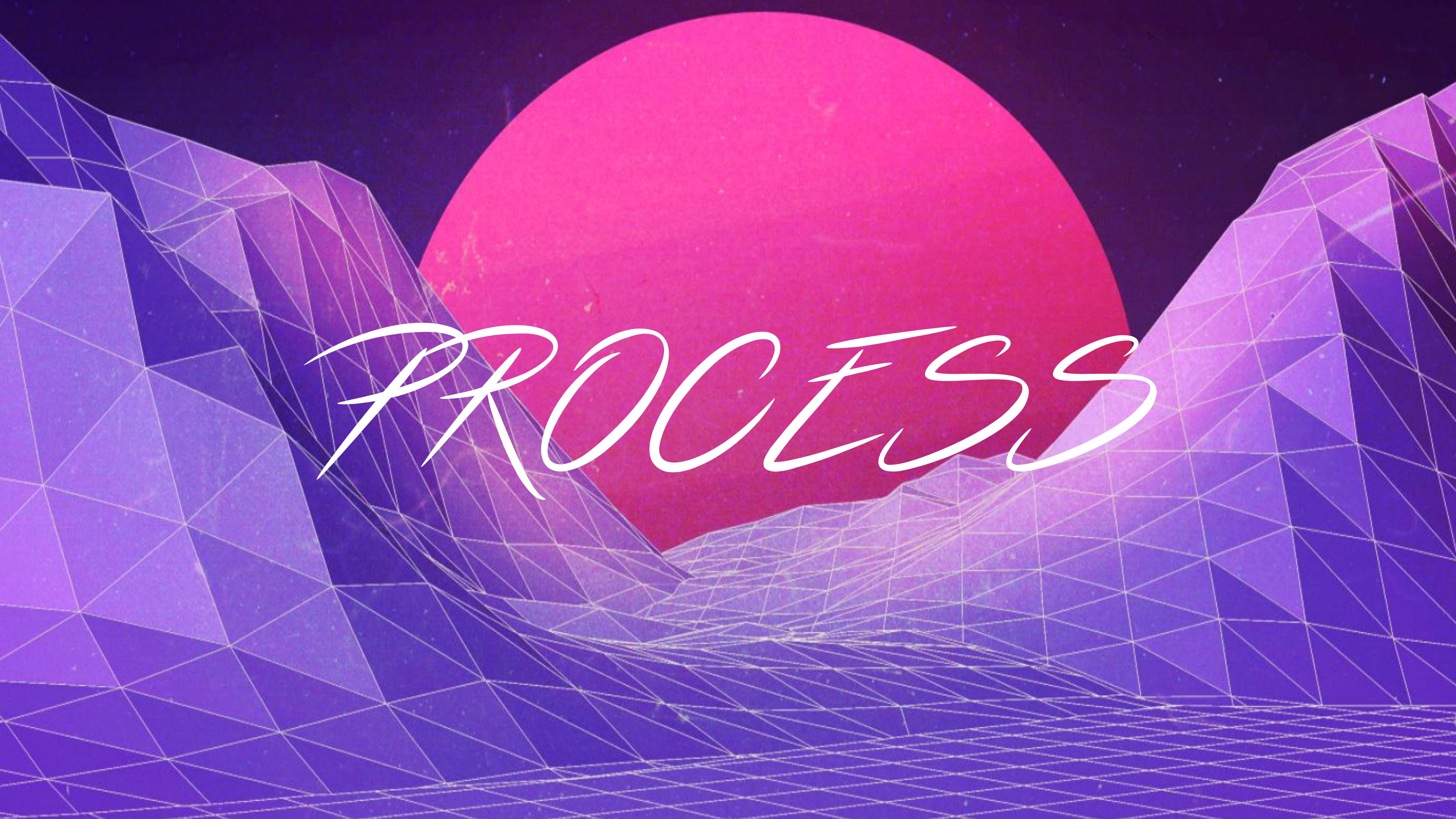
# "NORMAL" STATES

WAITING

RUNNABLE

RUNNING

```
iex(5)> :erlang.process_info self(), :status  
{:status, :running}
```

The background features a large, solid red circle centered in the frame. This circle is surrounded by several translucent, semi-transparent purple shapes that resemble stylized, rounded mountain peaks or hills. These purple shapes have a low-poly, wireframe-like texture with visible white lines forming their edges and vertices. The overall composition is minimalist and modern.

*PROCESS*

*SCHEDULERS*



*SINGLE*

*SCHEDULER*

```
iex(1)> :erlang.system_info:(smp_support)
```

```
true
```

```
iex(2)> :erlang.system_info :.schedulers_online
```

```
4
```

```
~> iex --erl "+S l"
```

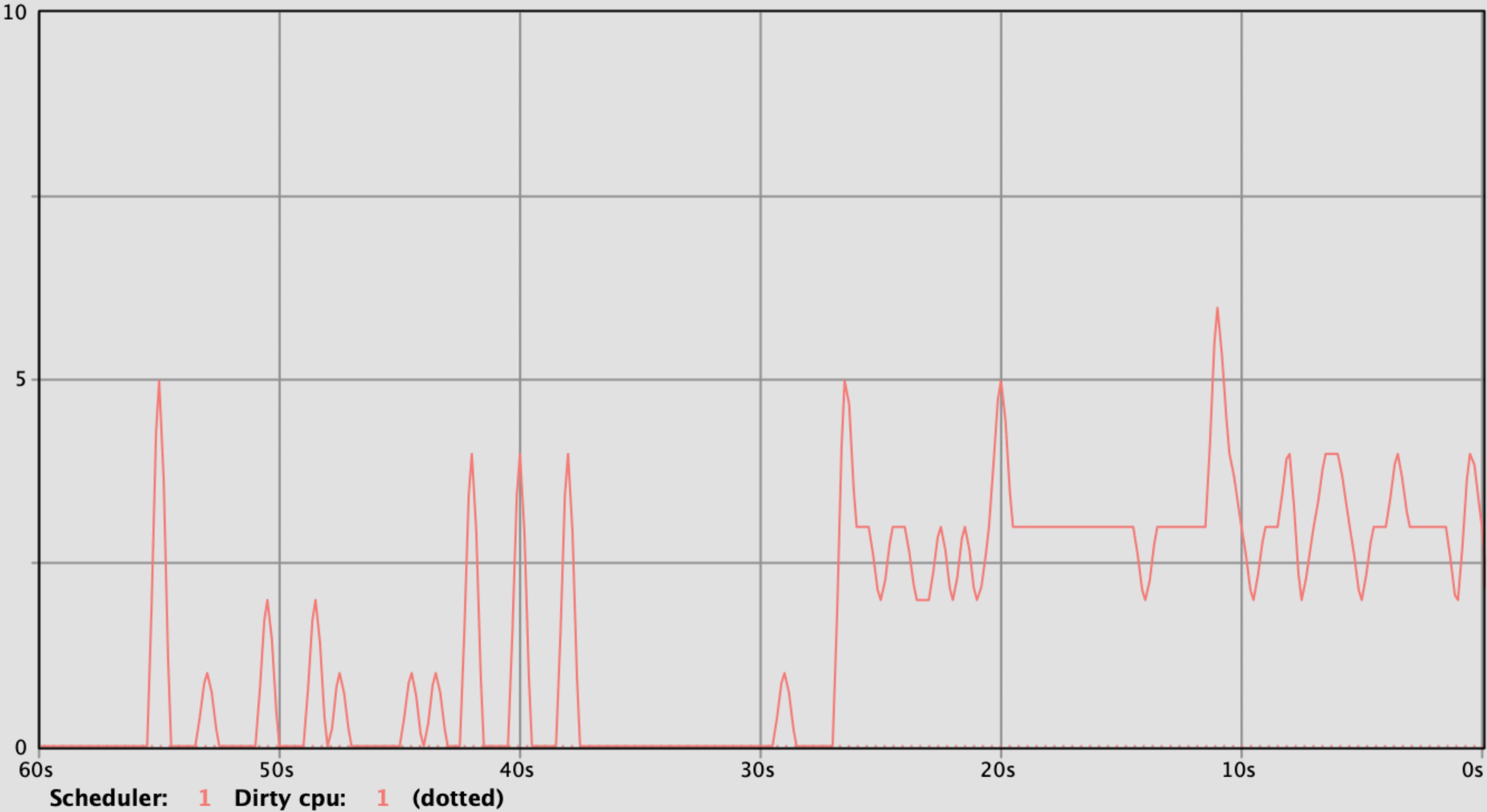
```
Erlang/OTP 21 [erts-10.0] [source] [64-bit] [smp:1:1] [ds:1:1:10] [async-threads:1] [hipe]
```

```
Interactive Elixir (1.7.3) - press Ctrl+C to exit (type h() ENTER for help)
```

```
iex(l)> :erlang.system_info :.schedulers_online
```

```
|
```

## Scheduler Utilization (%)



```
iex(25)> Process.sleep(10_000)
[info] GET /
[debug] Processing with SimpleWeb.PageController.index/2
Parameters: %{}
Pipelines: [:browser]
[info] Sent 200 in 150µs
```



*PREEMPTIVE\**



*REDUCTIONS*



4000

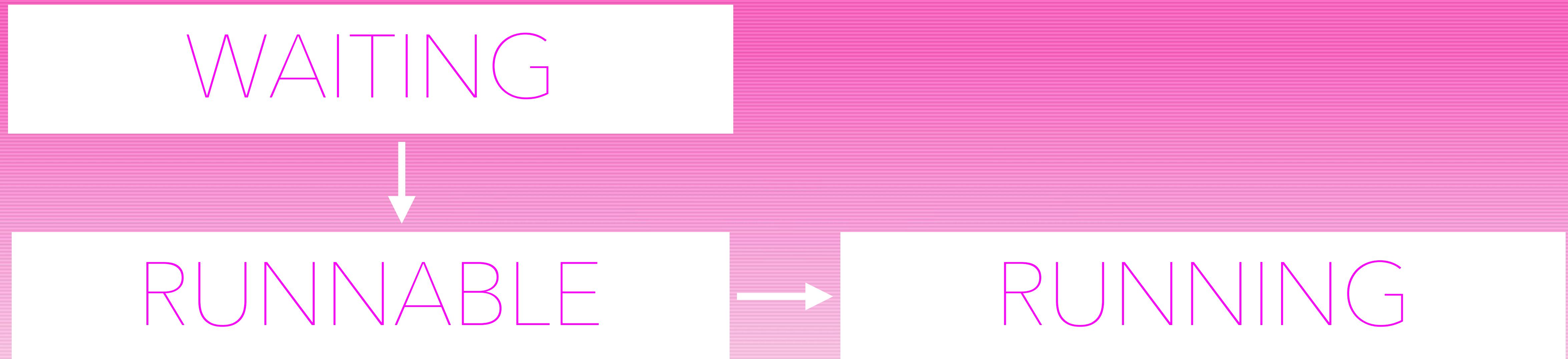
REDUCTIONS

```
#DEFINE CONTEXT_REDs 4000 /* SWAP PROCESS OUT AFTER THIS NUMBER */
```



[HTTPS://GITHUB.COM/ERLANG/OTP/BLOB/MASTER/ERTS/EMULATOR/BEAM/ERL\\_VM.H#L39](https://github.com/erlang/otp/blob/master/erts/emulator/beam/erl_vm.h#L39)

# RUN QUEUE



SCHEDULER



RUN QUEUE

TASK 1

TASK 2

TASK N



iex(11)> :erlang.statistics(:run\_queue)

0

SCHEDULER



RUN QUEUE

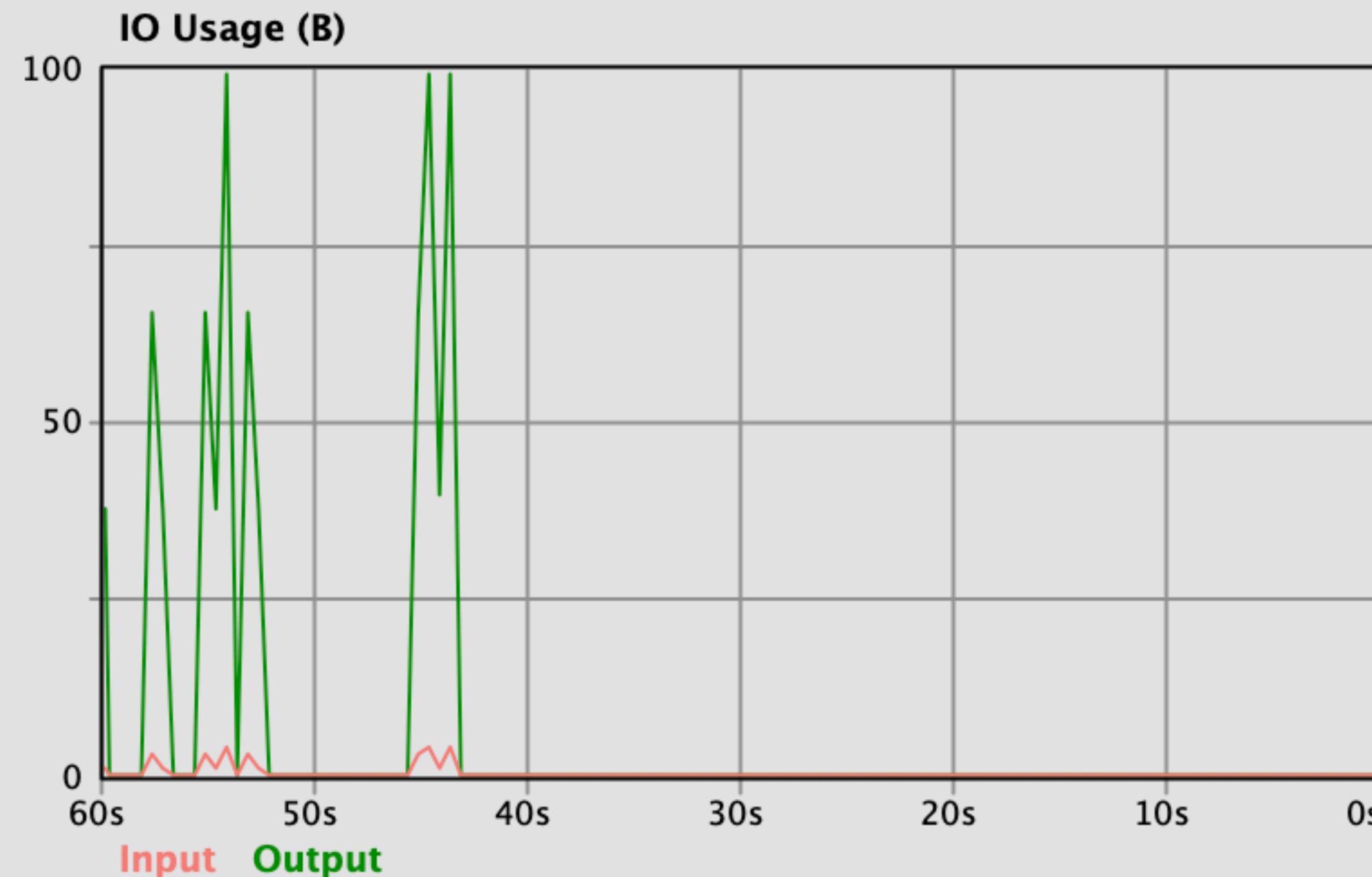
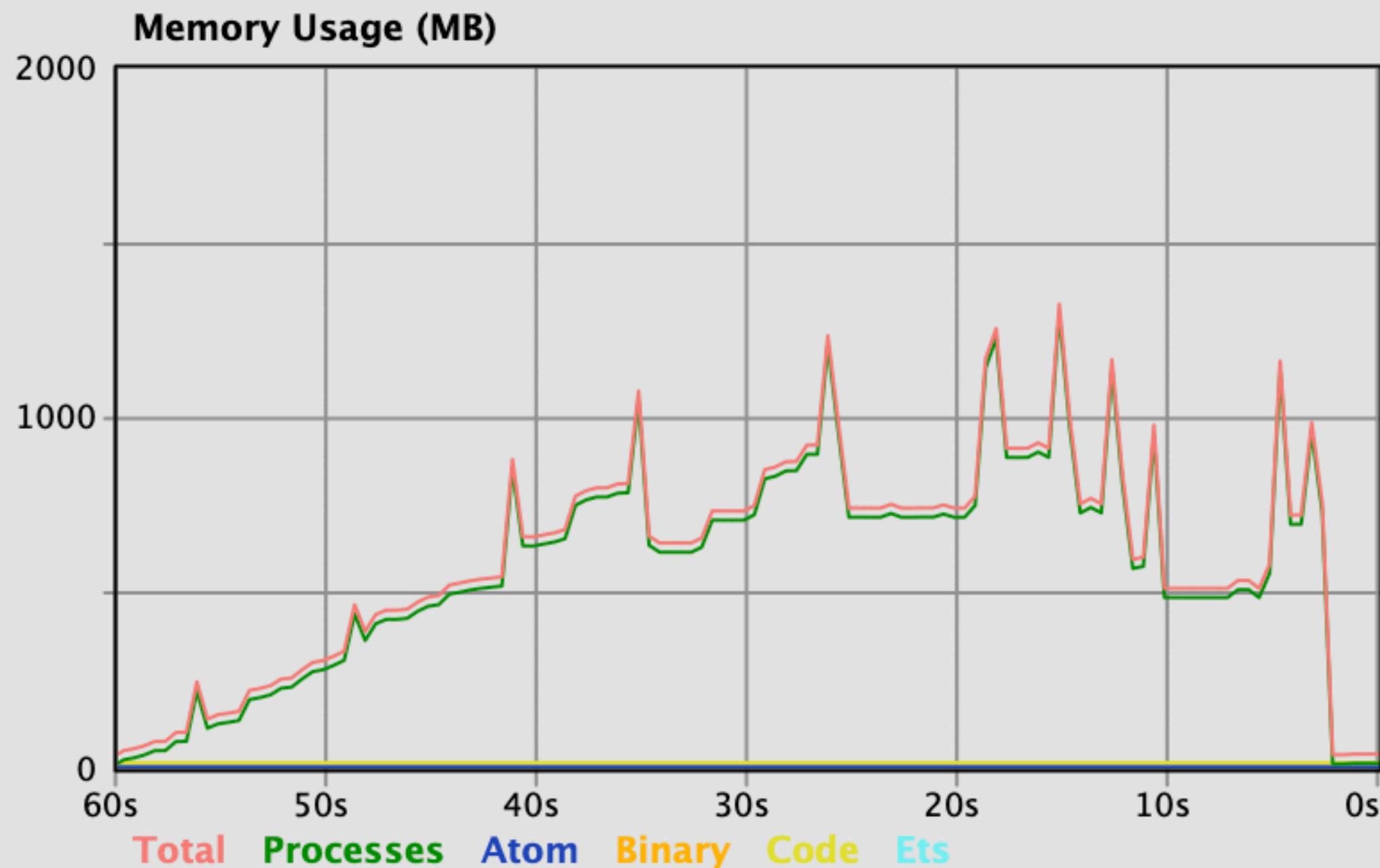
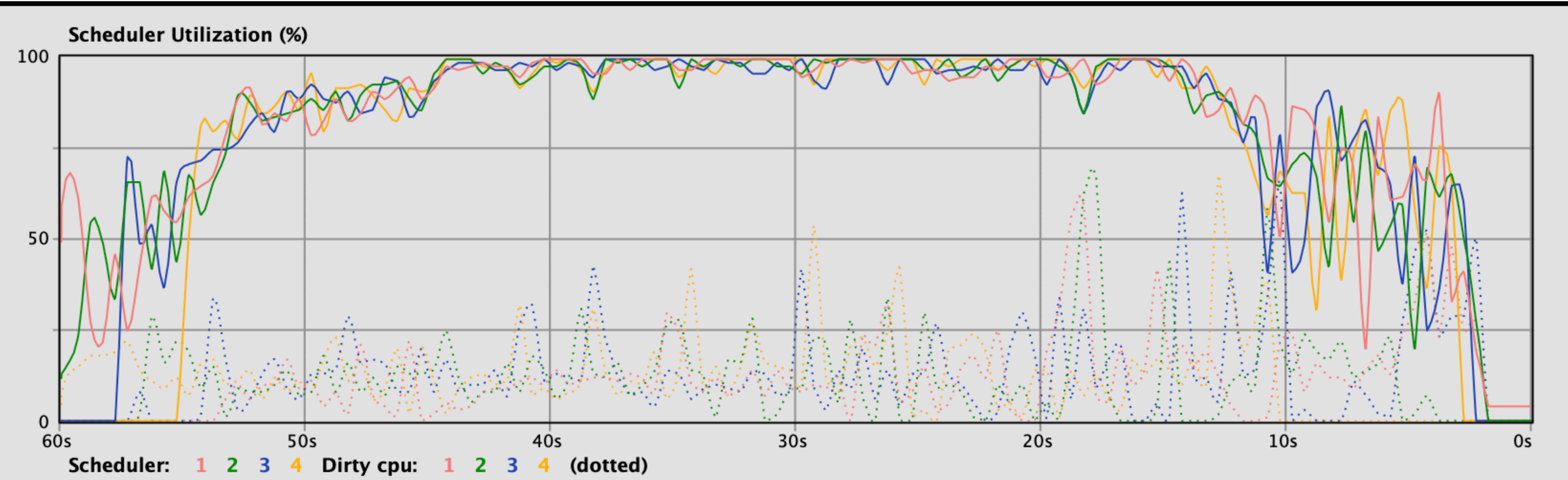
TASK 1

TASK 2

TASK N



MULTIPLY  
SCHEDULERS  
(SNP)



# RUN QUEUE

SCHEDULER



MAX PRIORITY

HIGH PRIORITY

NORMAL & LOW

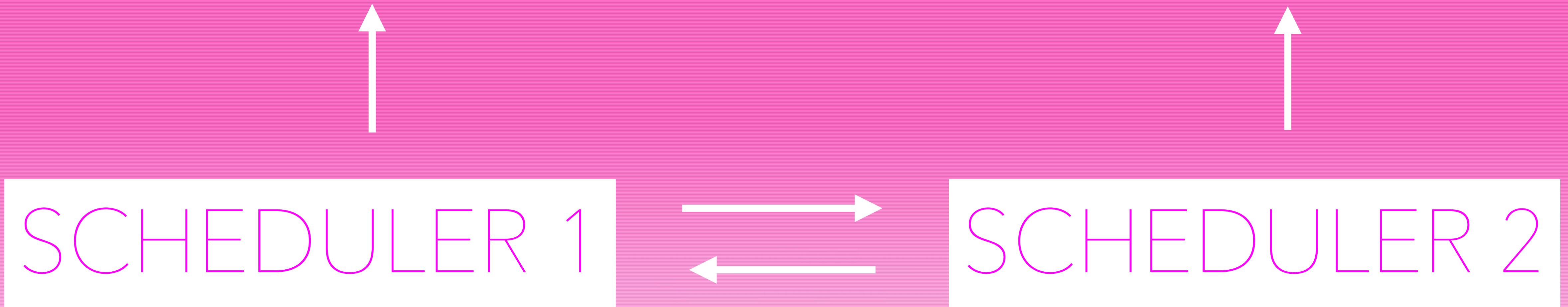
*LOAD  
BALANCING*



*THIEF*  
*STEALING*



# MIGRATION LOGIC



# Migration Logic



FINELY  
TUNED



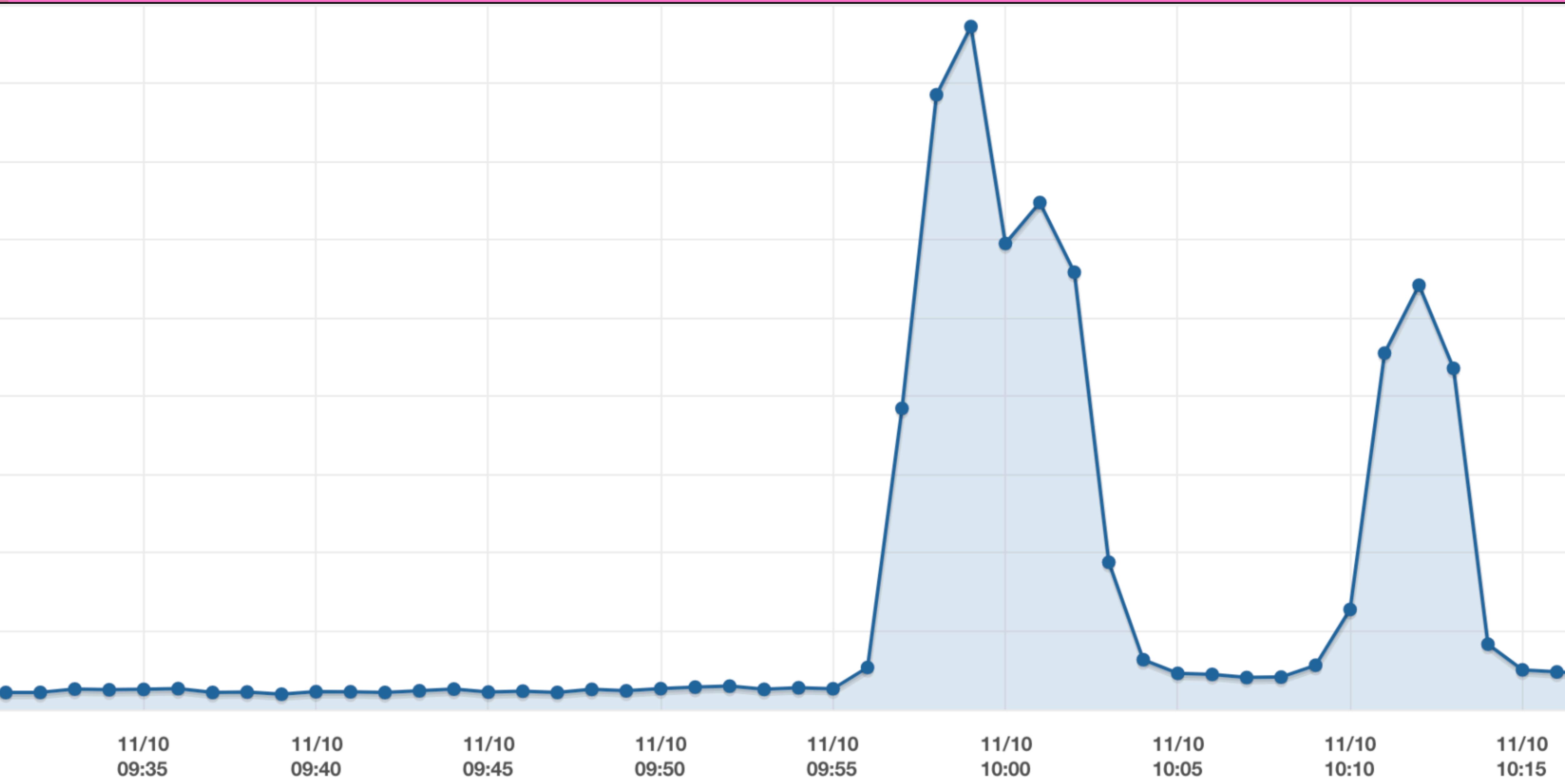
# SCHEDULER FLAGS

elixir --erl "+sbt ns" #etc

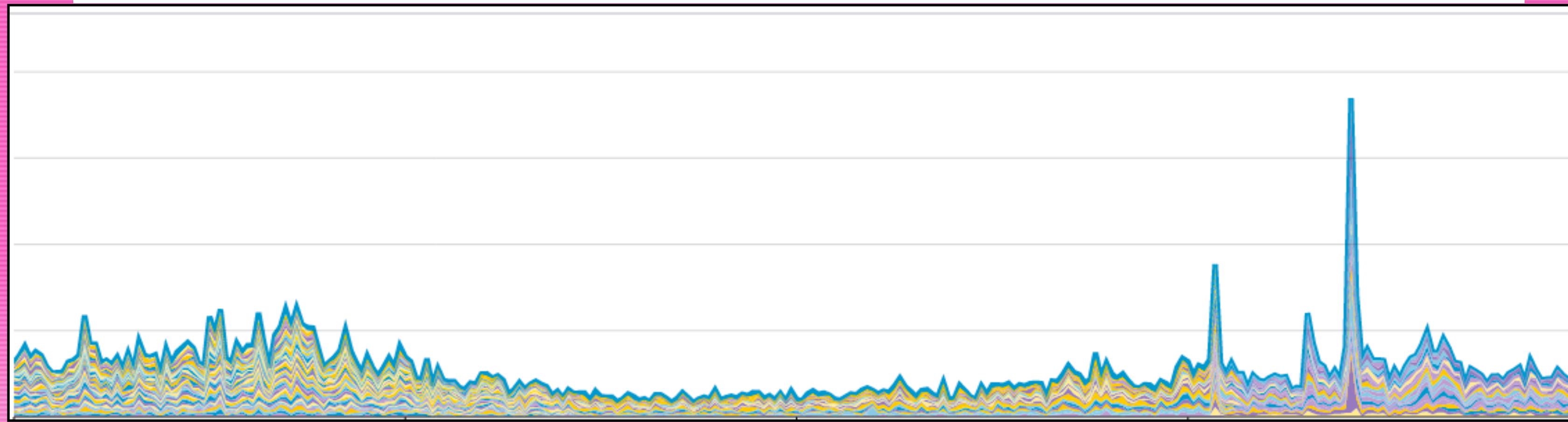
# MOCHI SCHEDULERS



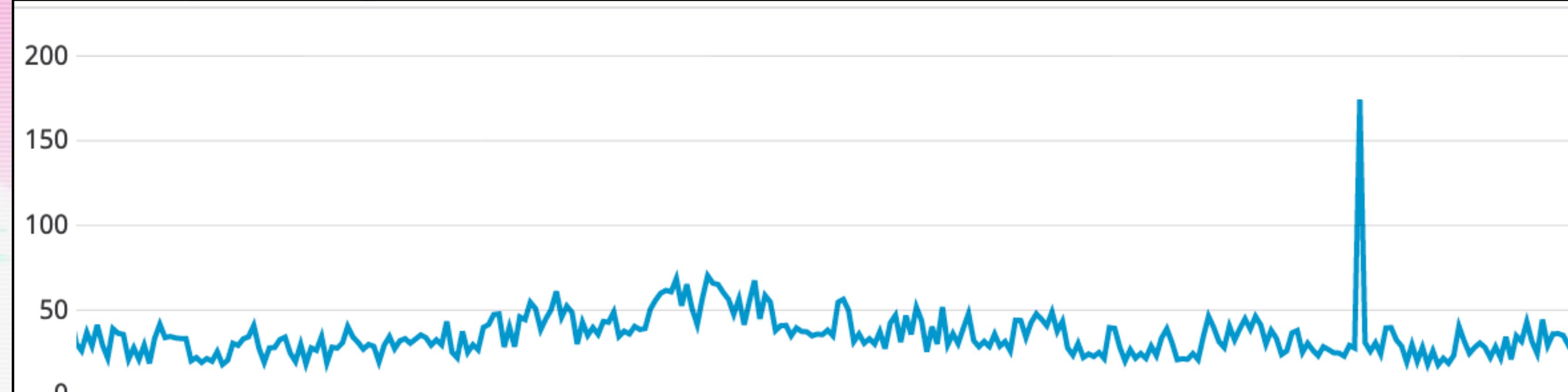
HONKAS  
THTS  
HEZ  
PFDI



# RUN QUEUE



# LATENCY



REQUEST

SCHEDULER ASSIGNED

ADDED TO RUN QUEUE

TRANSFORMATIONS

RESPONSE

# IMAGE CREDITS

<https://wallpaper-gallery.net/single/vaporwave-background-hd-14.html>

<http://www.desktopimages.org/preview/770453/1920/1080/>

[https://www.wallpapervortex.com/  
wallpaper-52083\\_l\\_miscellaneous\\_digital\\_art\\_vaporwave\\_image\\_glitch.html](https://www.wallpapervortex.com/wallpaper-52083_l_miscellaneous_digital_art_vaporwave_image_glitch.html)

<https://www.deviantart.com/epix0r/art/R-o-a-d-W-a-y-1920-x-1080-HD-629299503>

<https://wallpaper-gallery.net/single/glitch-wallpaper-hd-11.html>

# RESOURCES

The BEAM Book - <https://github.com/happi/theBeamBook>

<https://jlouisramblings.blogspot.com/2013/01/how-erlang-does-scheduling.html>

<http://www.erlang-factory.com/upload/presentations/105/KennethLundin-ErlangFactory2009London-AboutErlangOTPandMulti-coreperformanceinparticular.pdf>

<https://hamidreza-s.github.io/erlang/scheduling/real-time/preemptive/migration/2016/02/09/erlang-scheduler-details.html>

DZIEKUJE

@bgmarx