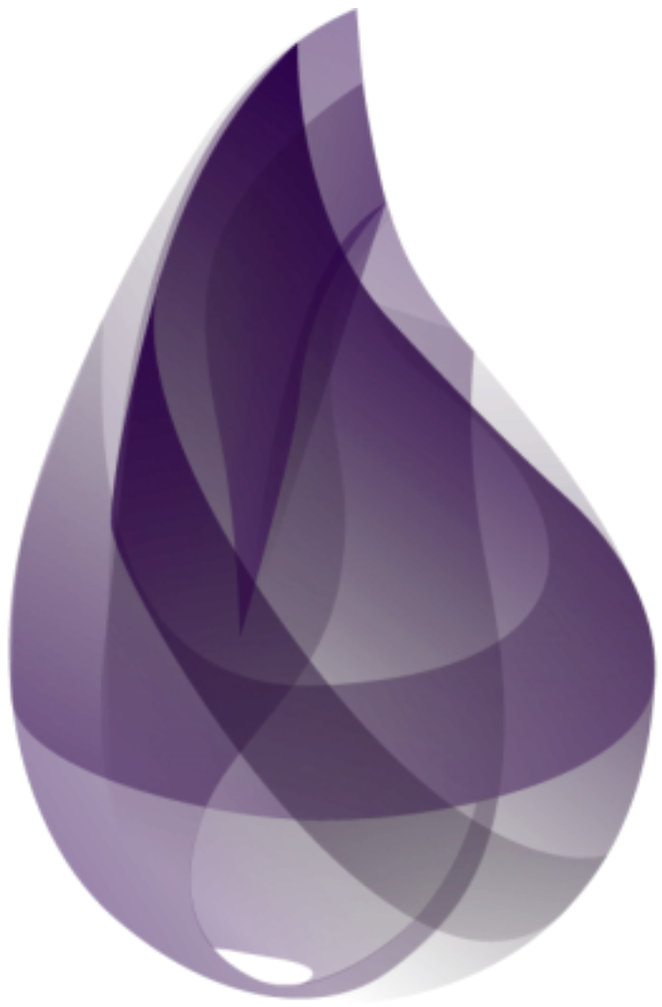


elixir

KLXRB June

elixir

- Runs on Erlang VM
- Syntax is borrowed from Ruby
- Functional
- Concurrent
- Distributed

Erlang VM

Transforming

```
milad@milad  ~/tmp/log ]
09:30:26 ] > cat cli/TH/broadcast.log | grep "Flushing"  | cut -d"[" -f4  | cut -d "]" -f1 | perl -pi -e '$a+=$_; $_=$a."\n"'
```

```
File.open("cli/TH/broadcast.log") |> grep "Flushing" |> cut  "[", 4  |> cut  "]", 4  |> sum total
```

“Elixir is a journey that you must dare to think differently.”

–Milad Rastian

Think Differently
(match operator)

```
root@000ab772f5a5:/# iex
Erlang/OTP 17 [erts-6.2] [source] [64-bit] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]

Interactive Elixir (1.0.2) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> a = 1
1
iex(2)> a + 3
4
iex(3)> 1 = a
1
iex(4)> 2 = a
** (MatchError) no match of right hand side value: 1

iex(4)>
```

Think Differently (mutable vs immutable data)

```
[ milad@milad  ~/sandbox/ruby ]
[ 09:55:08 ] > pry
[1] pry(main)> my_list = []
=> []
[2] pry(main)> my_list << "1"
=> ["1"]
[3] pry(main)> my_list
=> ["1"]
[4] pry(main)> my_list << "2"
=> ["1", "2"]
[5] pry(main)> my_list
=> ["1", "2"]
[6] pry(main)>
```

```
root@000ab772f5a5:/# iex
Erlang/OTP 17 [erts-6.2] [source] [64-bit]

Interactive Elixir (1.0.2) - press Ctrl+C
iex(1)> my_list = []
[]
iex(2)> my_list = [ 1 | my_list ]
[1]
iex(3)> my_list = [ 2 | my_list ]
[2, 1]
iex(4)> my_list = [ 3 | my_list ]
[3, 2, 1]
iex(5)> my_list
[3, 2, 1]
iex(6)>
```

Think Differently (pattern matching)

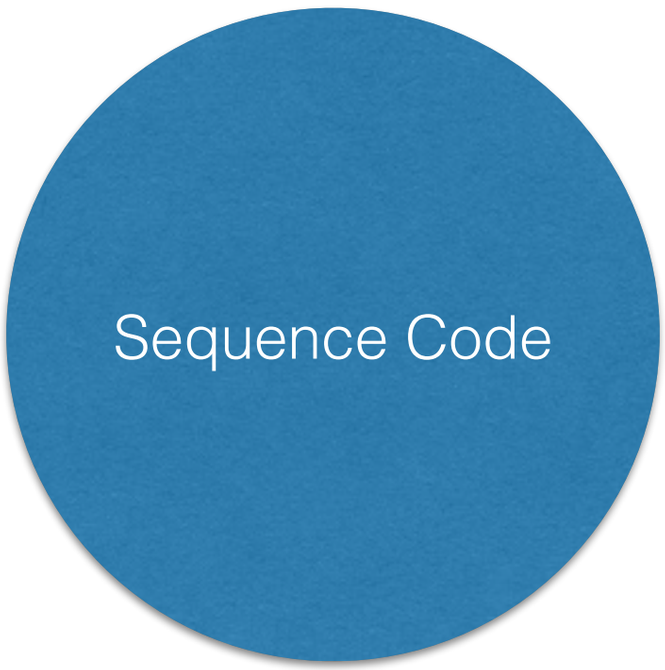
```elixir
handle_open = fn
  {:ok, file} -> "Read data: #{IO.read(file, :line)}"
  {_, error}  -> "Error: #{:file.format_error(error)}"
end

handle_open.(File.open("access.log"))
#Error: no such file or directory

File.open("/var/log/access.log") |> handle_open.()
#Read data: May 31 11:40:20 "PUT /api/v1/ports/my HTTP/1.1" 200
```

Think Differently (meta programming)

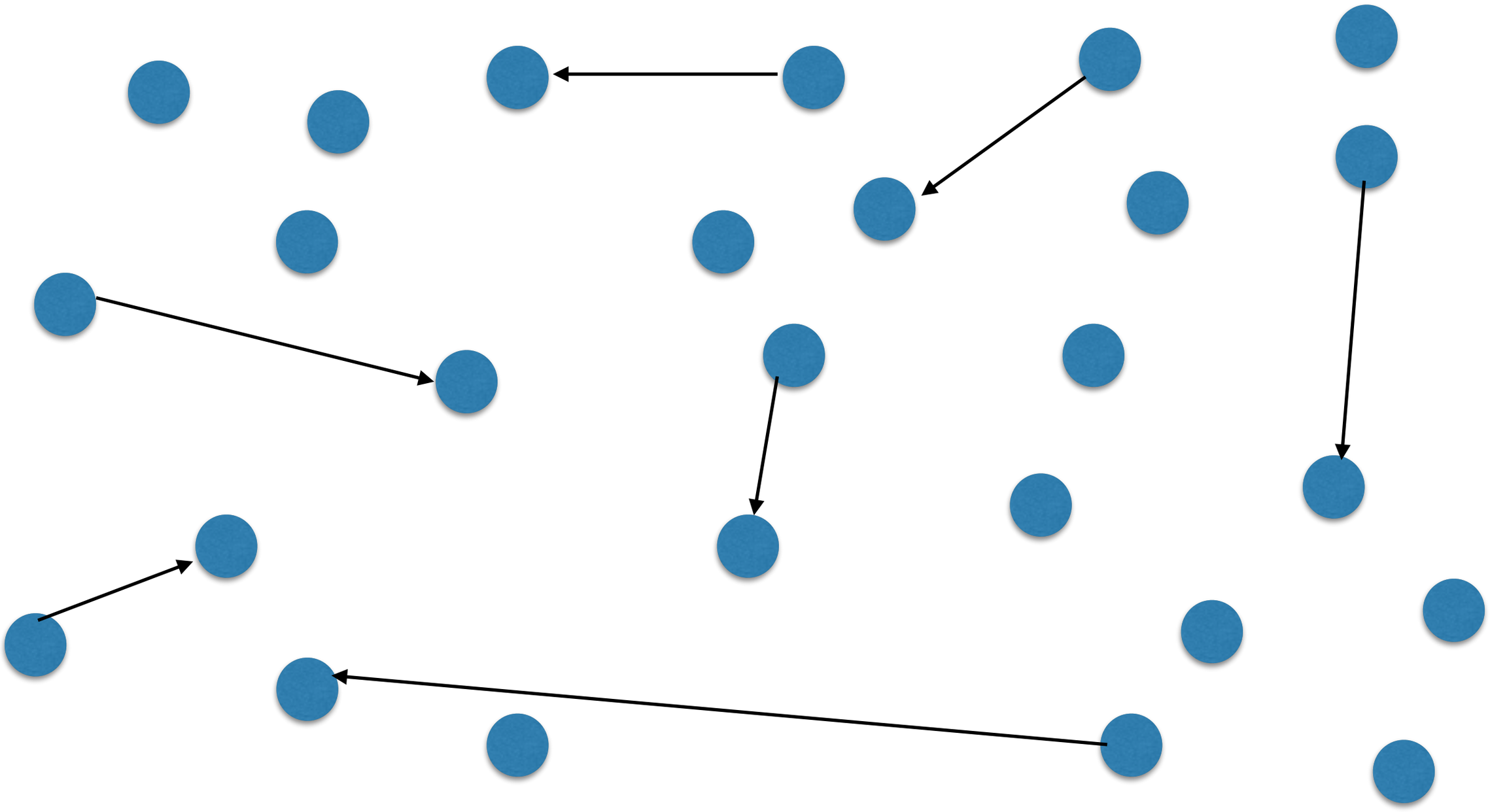
```elixir
defmodule Unless do
  def fun_unless(clause, expression) do
    if(!clause, do: expression)
  end

  defmacro macro_unless(clause, expression) do
    quote do
      if(!unquote(clause), do: unquote(expression))
    end
  end
end
```

```
iex(15)> require Unless
nil
iex(16)> Unless.macro_unless true, IO.puts "this should never be printed"
nil
iex(17)> Unless.fun_unless true, IO.puts "this should never be printed"
this should never be printed
nil
```

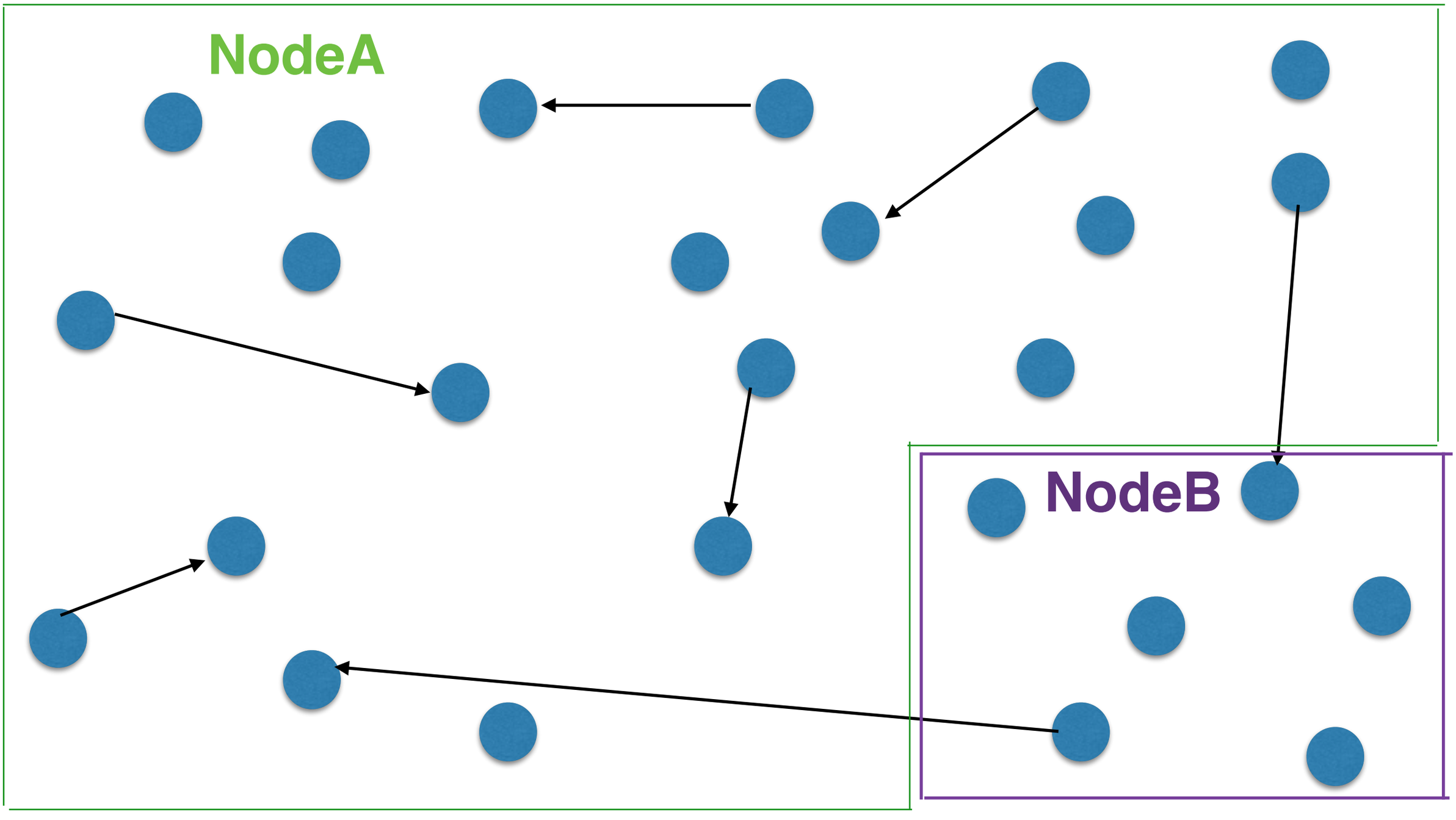
Think Differently
(process / concurrency / distribution)



Think Differently
(process / concurrency / distribution)



Think Differently
(process / concurrency / distribution)



Fibonacci

Live Coding!

Where to go from here