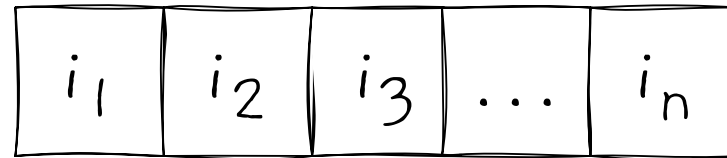


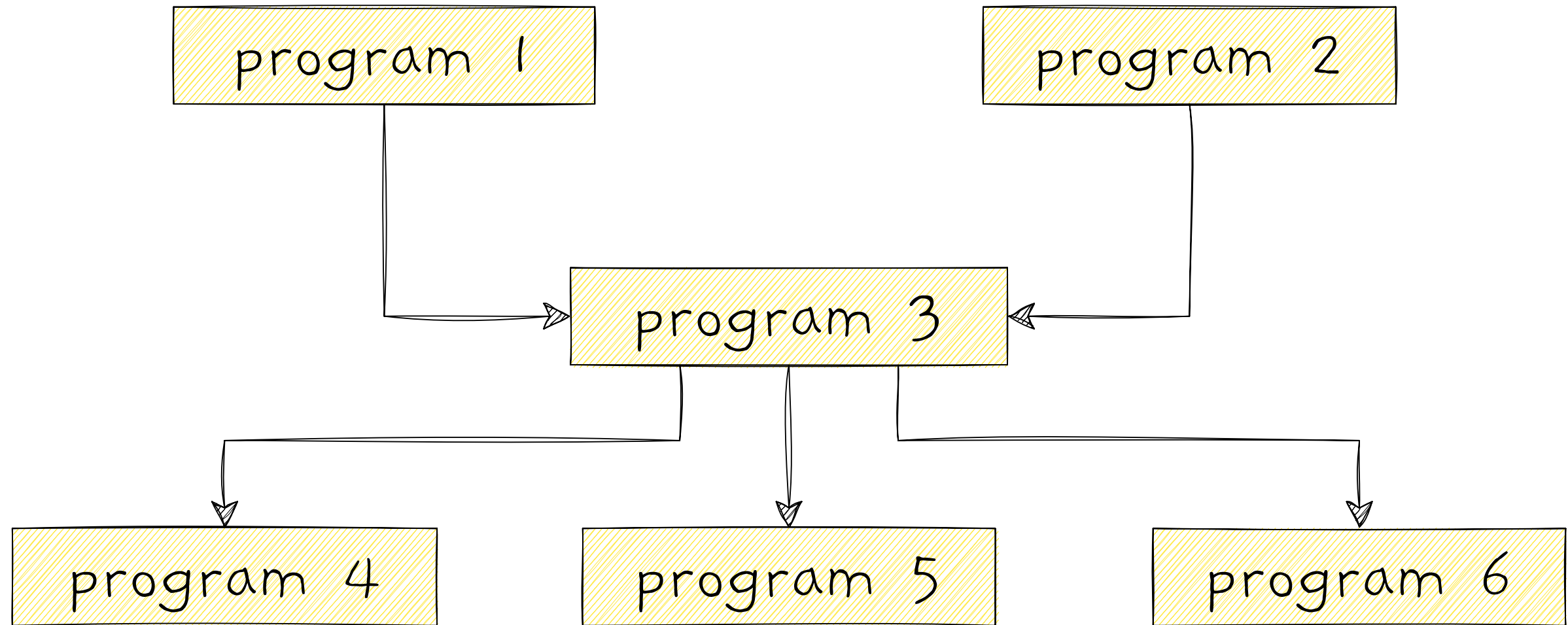
Concurrent Elixir

program



## **concurrency**

multiple sequential programs  
parts of a greater whole



system

web server

proxy

load balancer

database server

message queue

home automation

system

runs for a long time

does multiple things at once

non-binary success

high availability

essential for any software system



BEAM concurrency is designed for high availability

## **process**

sequential program

runtime execution context

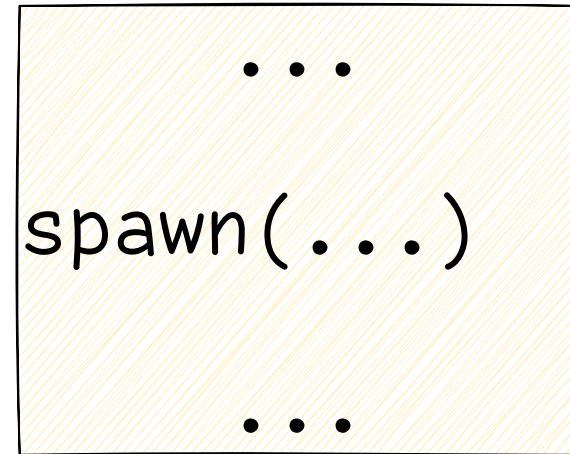
not an OS process (nor thread)

```
# process A
```

```
foo(...)  
bar(...)
```

```
spawn(fn →  
      # process B  
      ...  
end)
```

process a



process b



```
pid = spawn(fn → ... end)  
my_pid = self()
```

```
...
```

```
send(pid, some_message)
```

```
receive do  
  message →  
  do_something_with(message)  
end
```

```
n = 10
caller_pid = self()

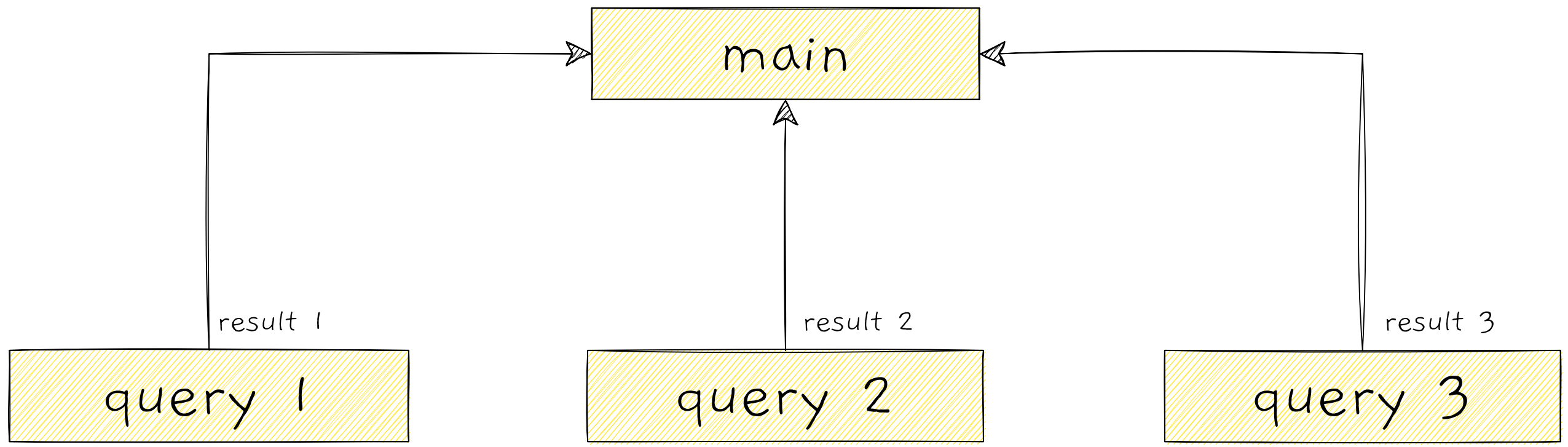
spawn(fn →
  x = fib(n)
  send(caller_pid, {:fib, x})
end)

receive do
  {:fib, x} →
    IO.inspect(x)
end
```

```
db_queries = [...]  
caller_pid = self()
```

```
Enum.each(  
  db_queries,  
  fn query →  
    spawn(fn →  
      result = run_query(query)  
      send(caller_pid, {:result, result})  
    end)  
  end  
)
```





```
Enum.map(  
  db_queries,  
  fn _query →  
    receive do  
      {:result, result} → result  
    end  
  end  
)
```

The bartender asks what they want.

Two threads walk into a bar.

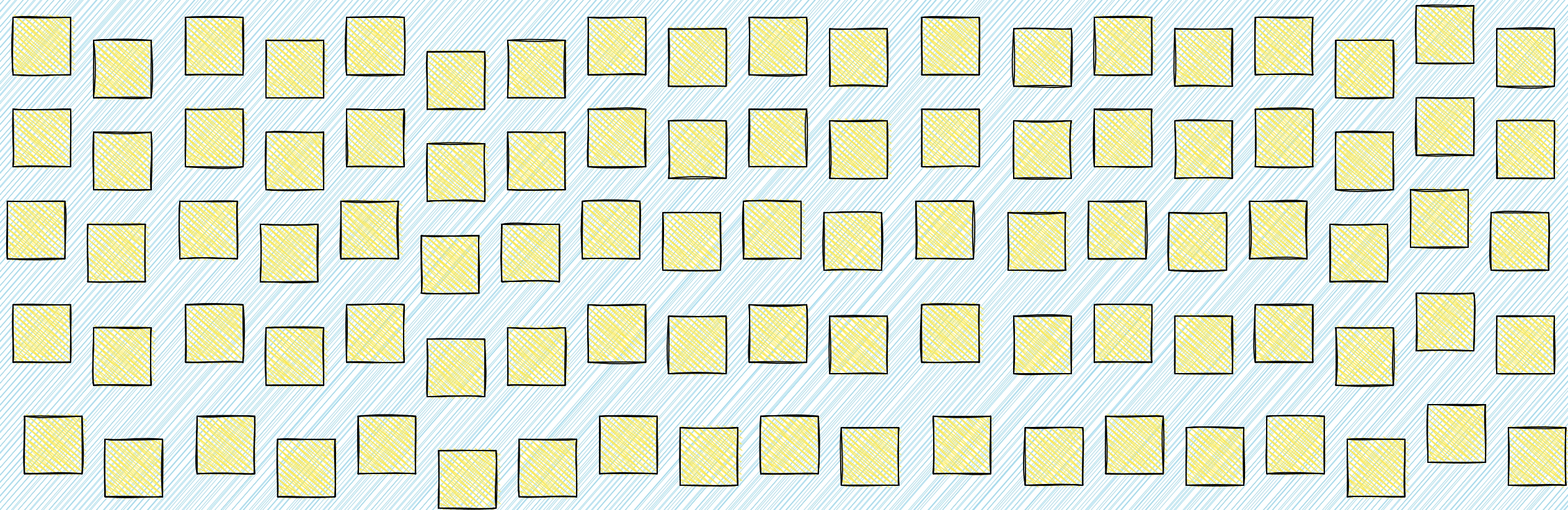
```
init_mem = :erlang.memory(:total)
```

```
Enum.each(  
  1..100_000,  
  fn _ → spawn(fn → Process.sleep(:infinity) end) end  
)
```

```
round(:erlang.memory(:total) - init_mem) / 1_000_000)
```

```
# 264 MB
```

# BEAM



scheduler

scheduler

scheduler

scheduler

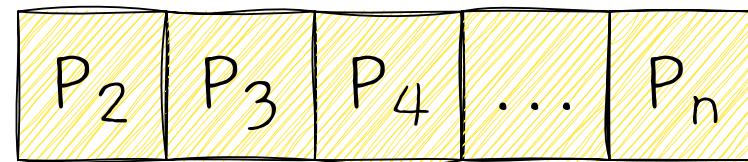
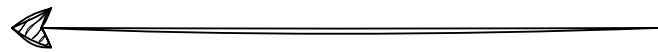
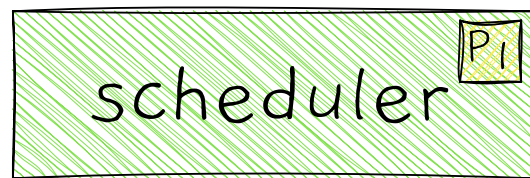
CPU

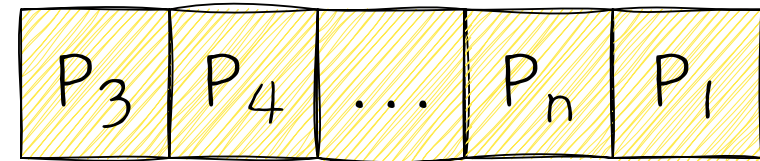
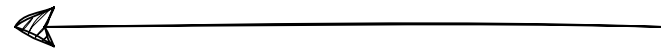
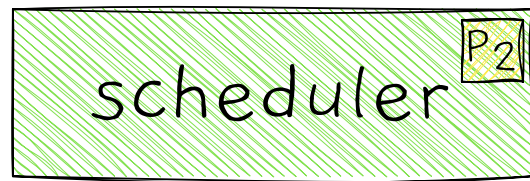
CPU

CPU

CPU

demo







## context switching

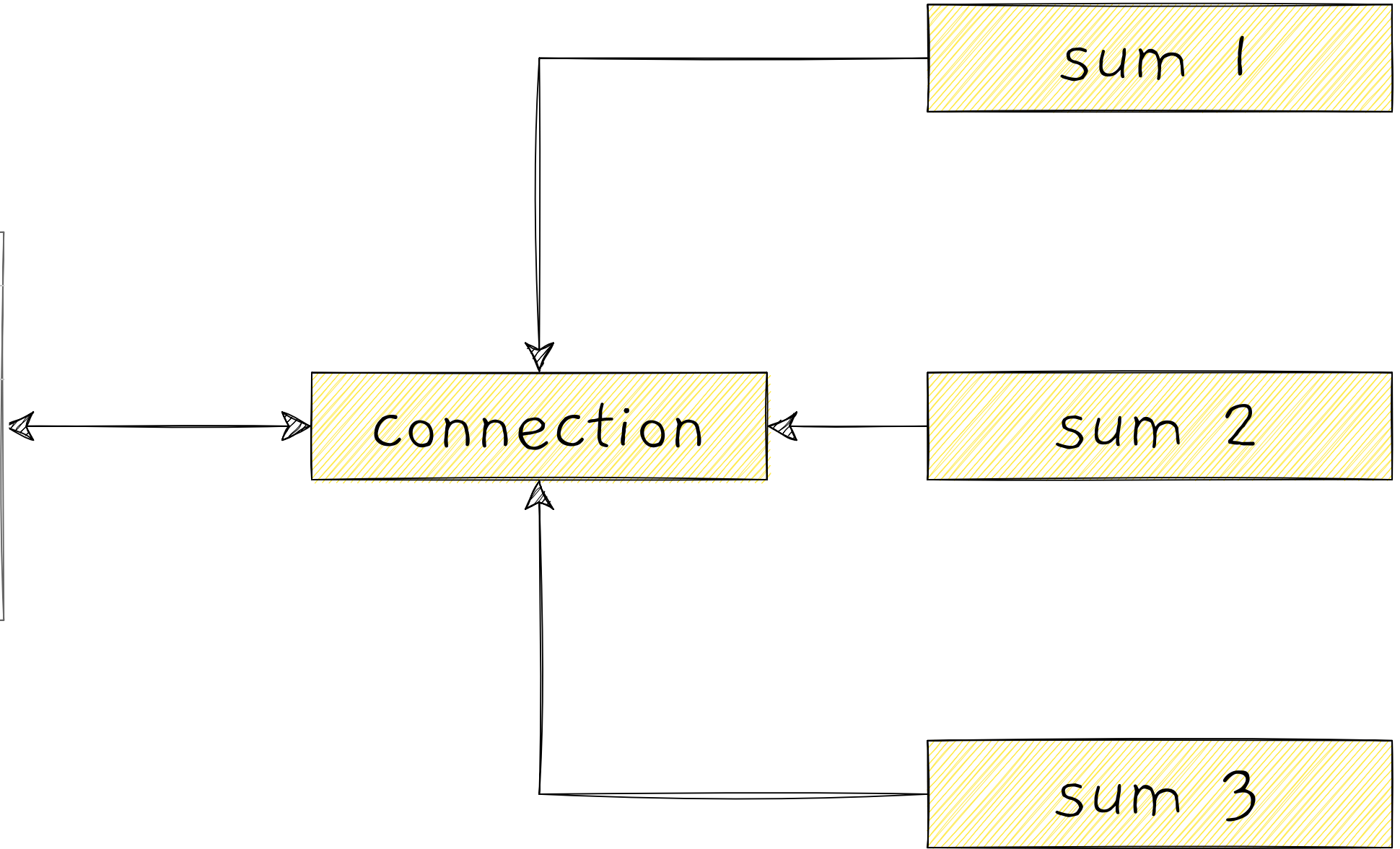
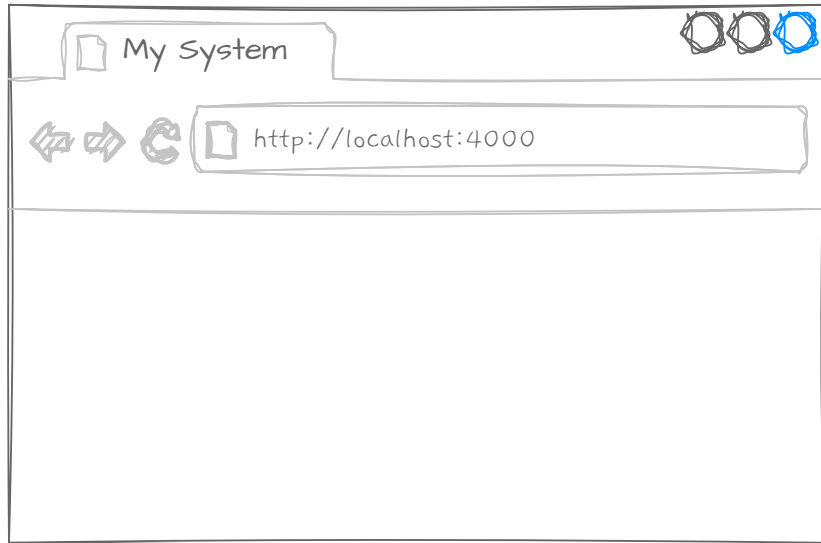
preemptive

typically every 1ms or less

=> fair distribution of CPU

stable progress of the entire system

demo



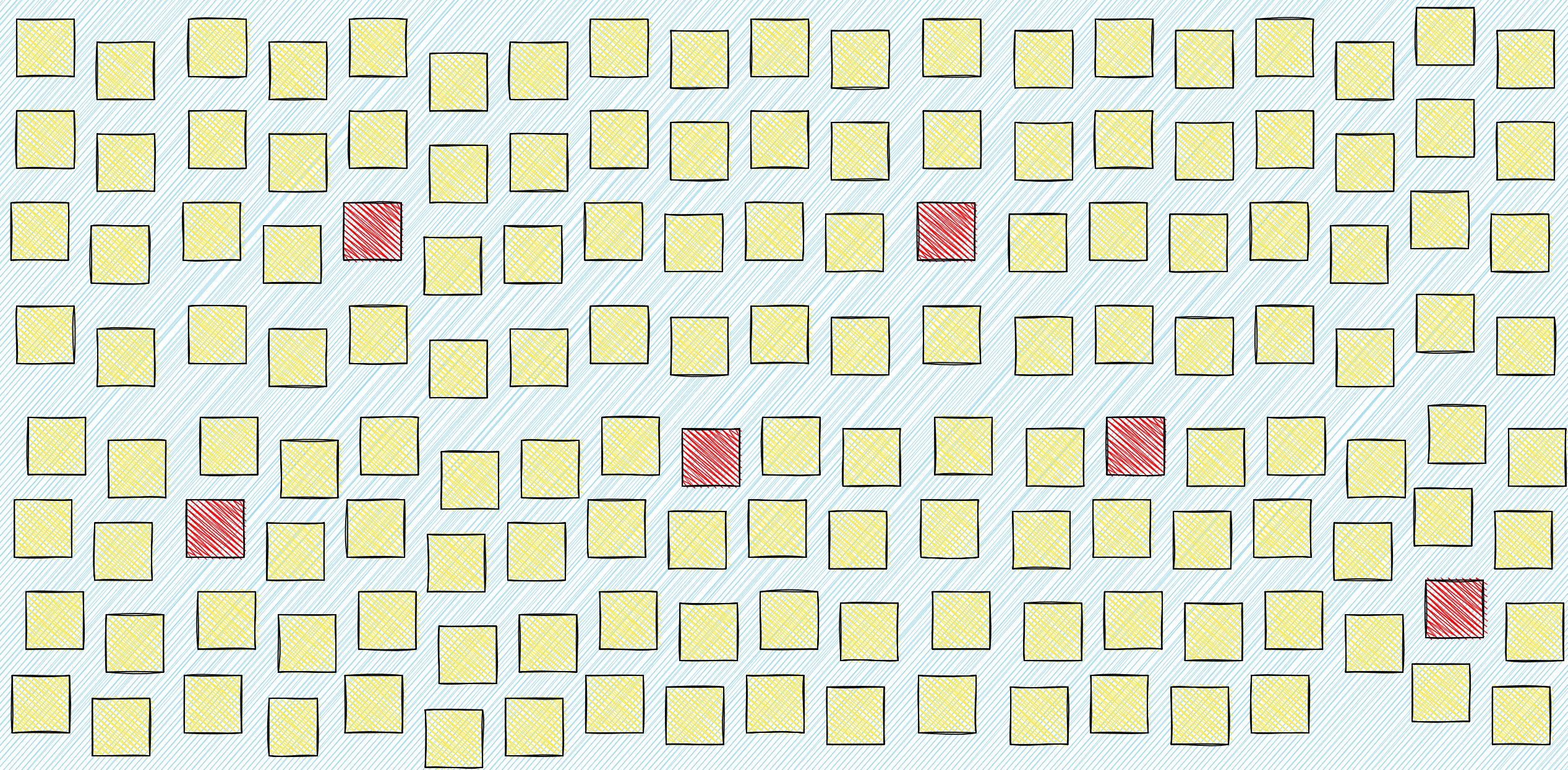
## fault tolerance

provide service in the presence of errors  
self-heal as soon as possible

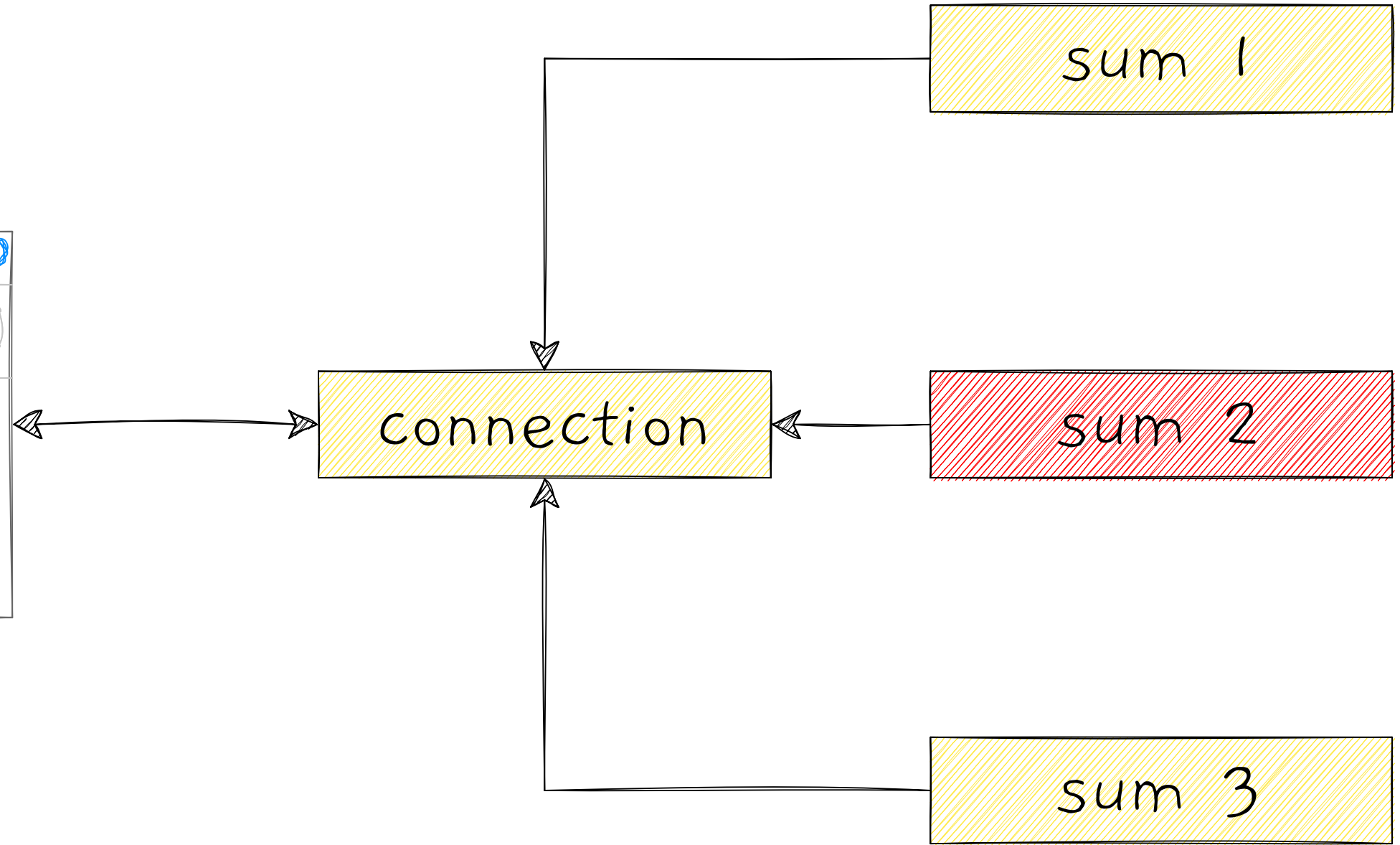
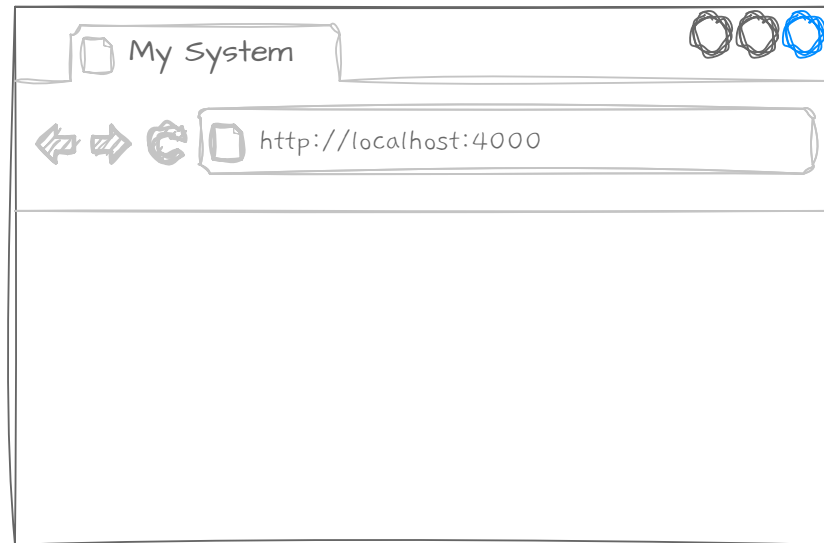
## process crash

caused by an unhandled exception  
only the affected process goes down  
other processes can be notified

# BEAM



demo







understanding production behaviour

demo

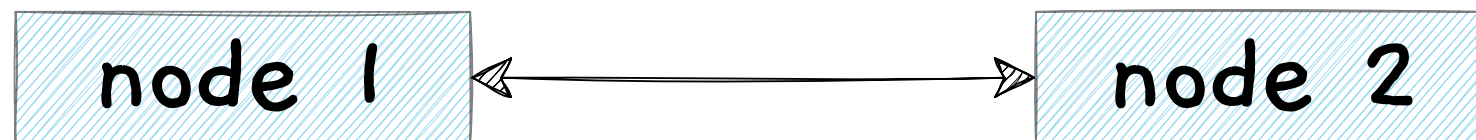
*distributed Elixir*

```
terminal   
$ iex --name node1@127.0.0.1  
iex(node1@127.0.0.1)1>
```

```
terminal   
$ iex --name node2@127.0.0.1  
iex(node2@127.0.0.1)1>
```

```
# on node2
```

```
Node.connect(":node1@127.0.0.1")
```



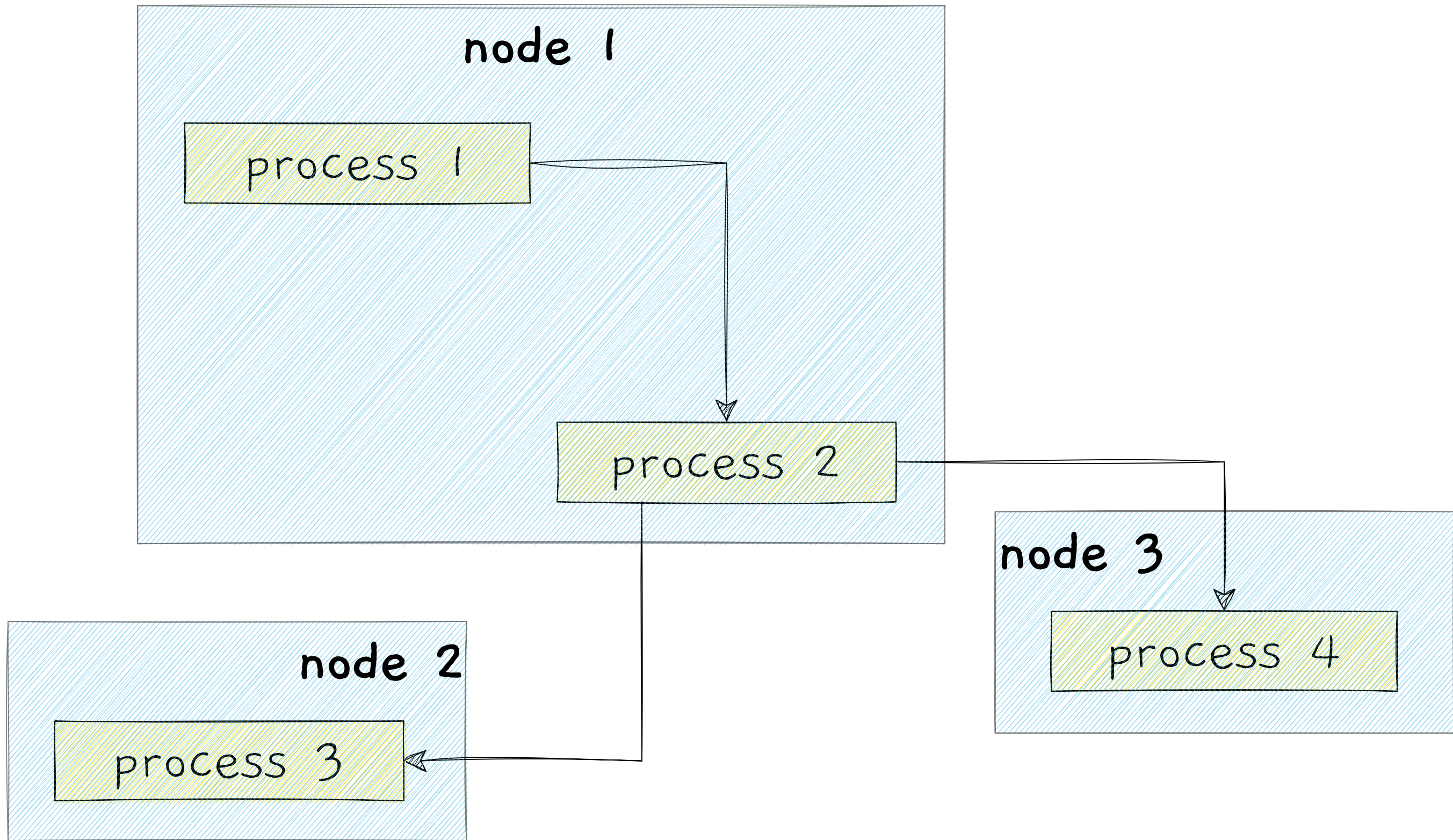
```
# on node2
Node.spawn(
  "node1@127.0.0.1",
  fn →
    IO.puts("Hello from #{node()}")
  end
)
```

```
# Hello from node1@127.0.0.1
```

location transparency

pid may point to a remote process



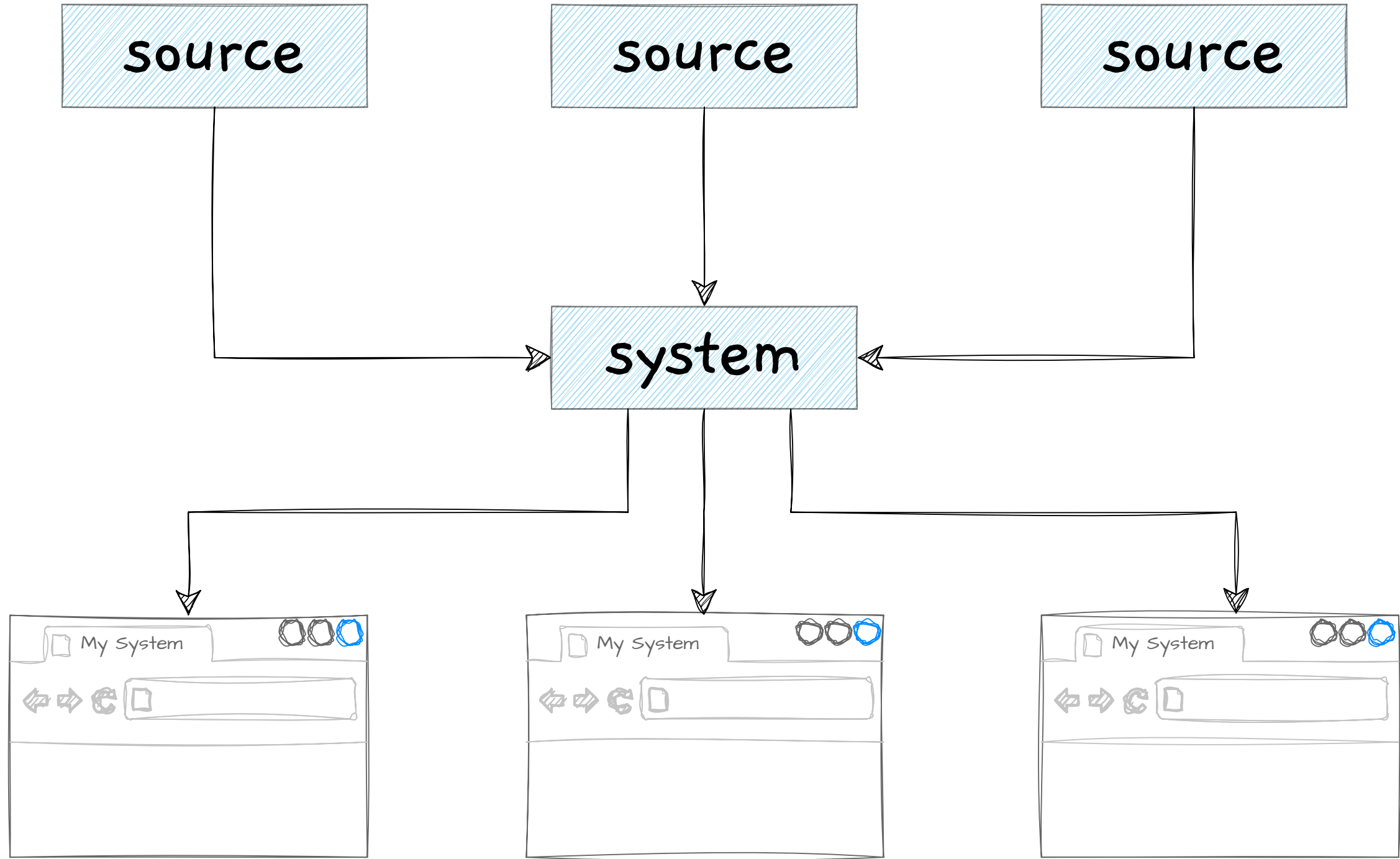


concurrency == distributed programming

demo

lightweight processes  
message-passing concurrency  
preemptive scheduling  
process isolation  
termination notifications  
remote shell  
distribution

case study



## ingestion steps

consume bytes

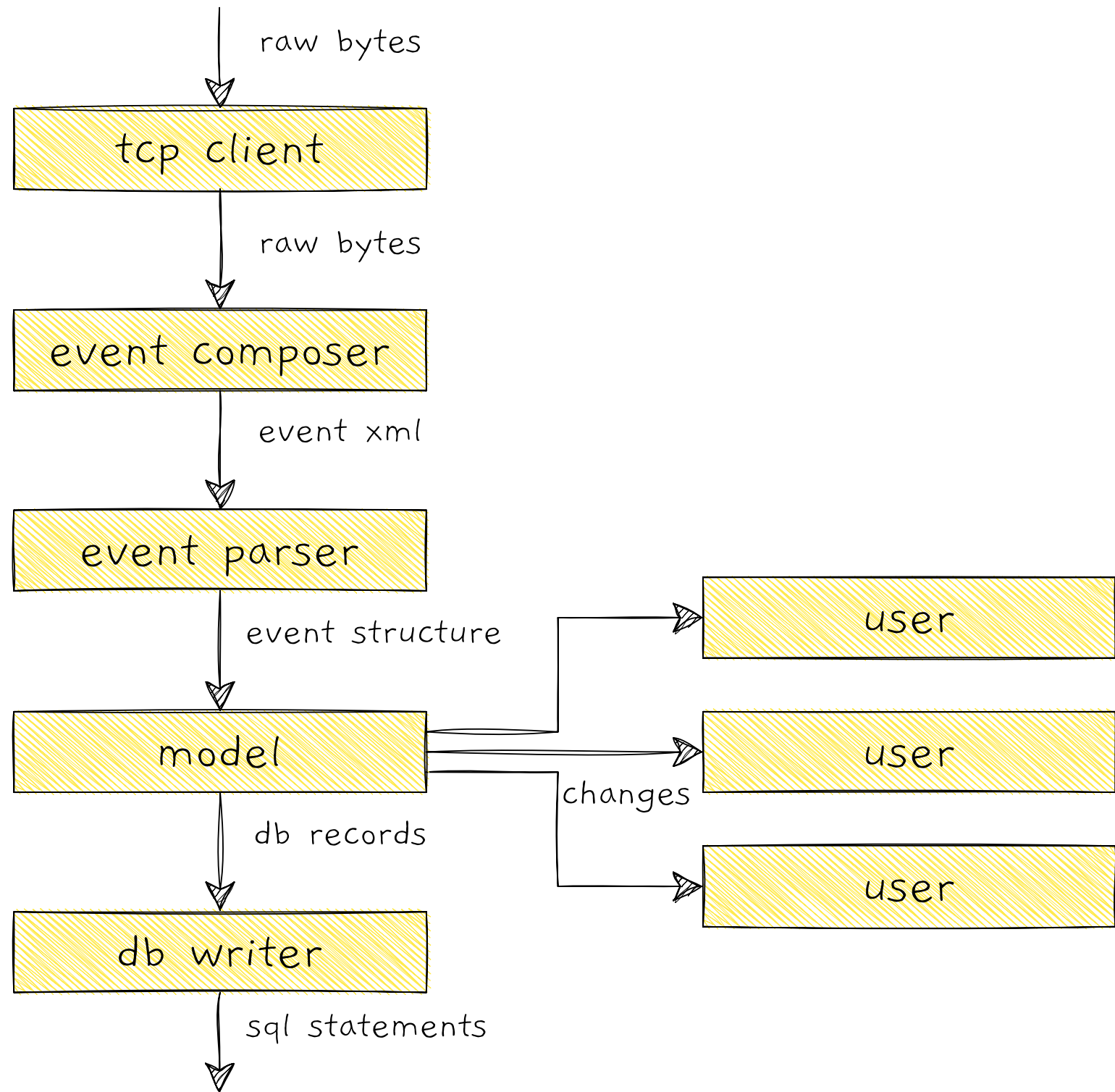
compose event xml

parse event xml

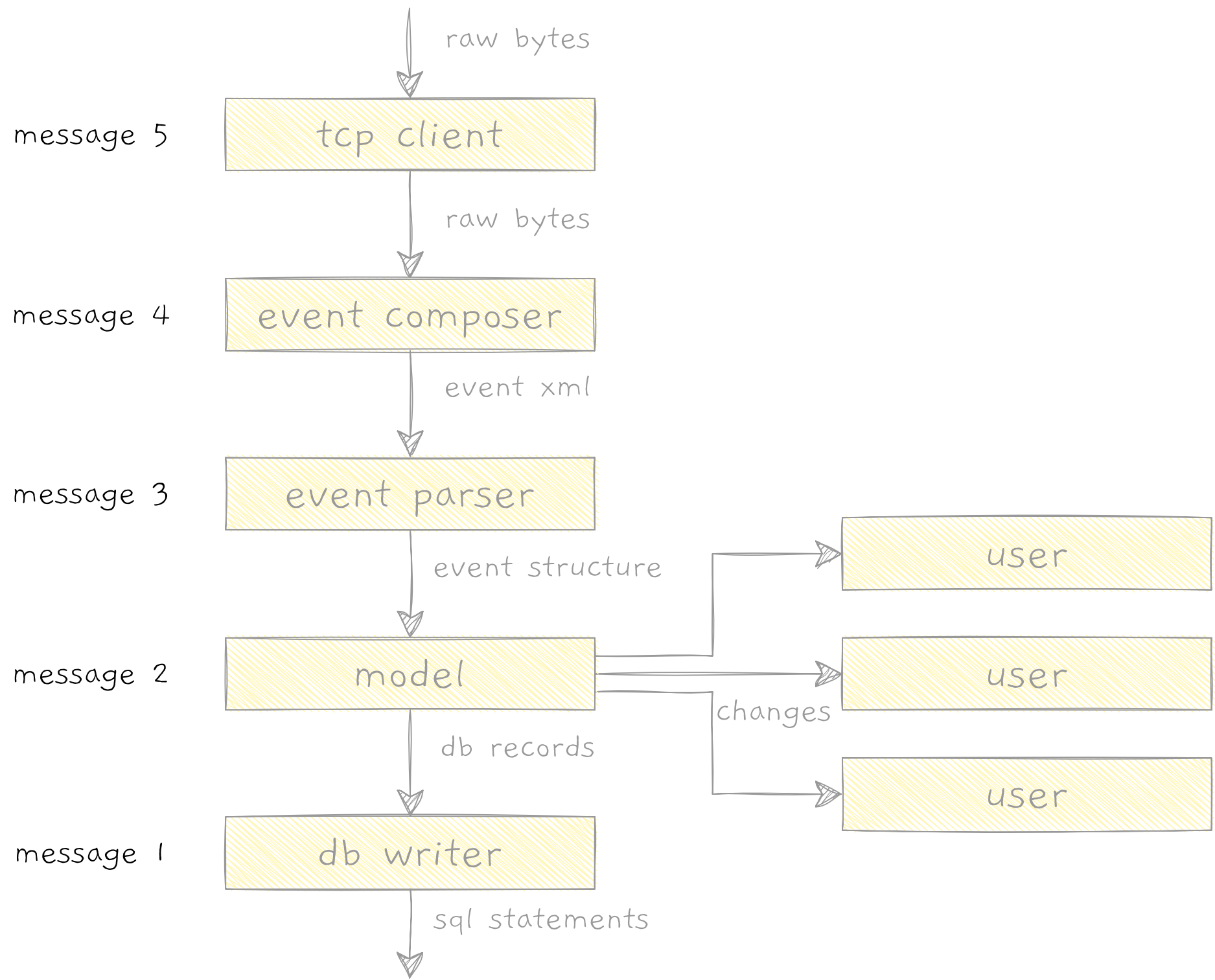
apply event to the model

notify connected users

store to database

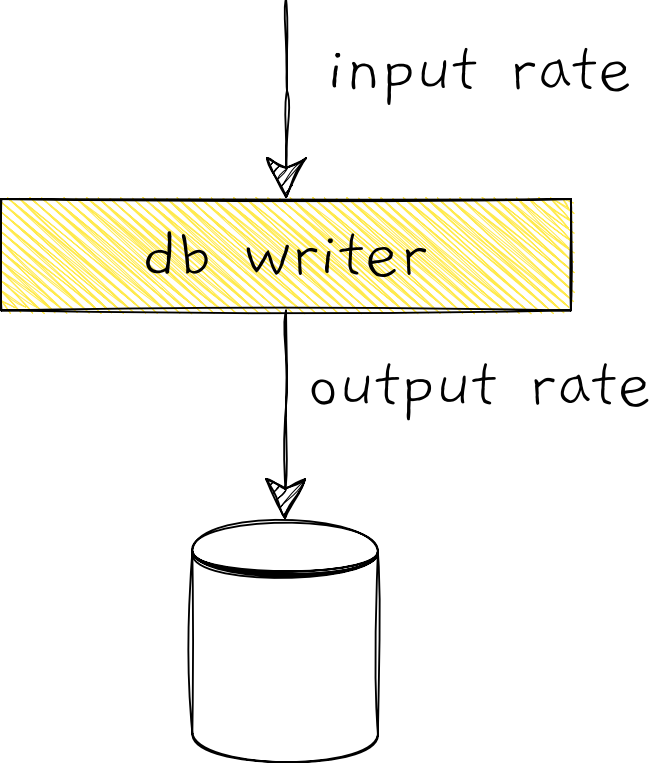






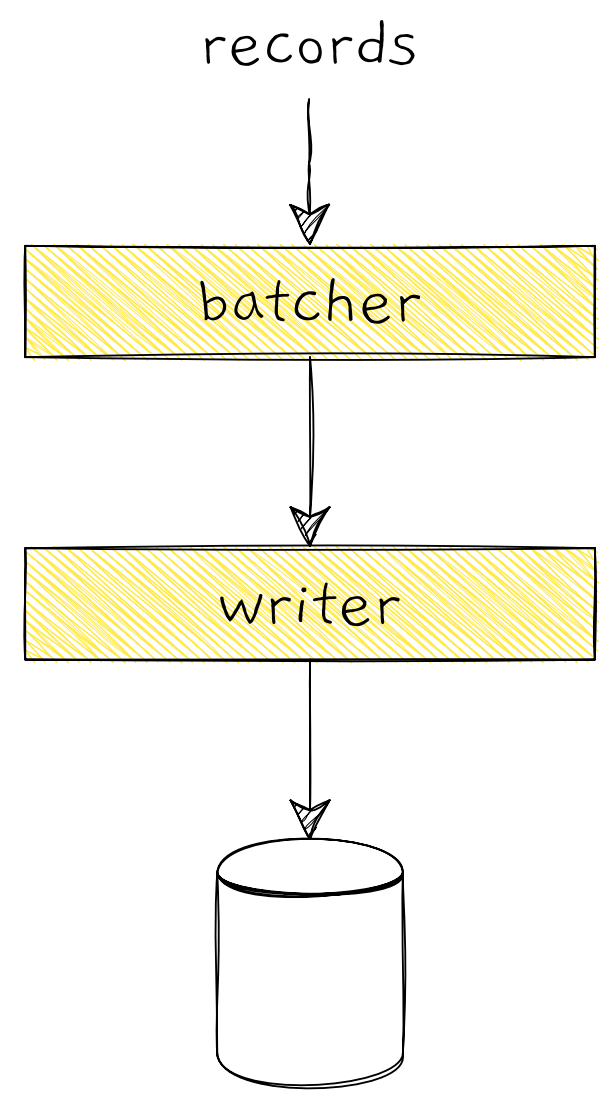
db writing challenge

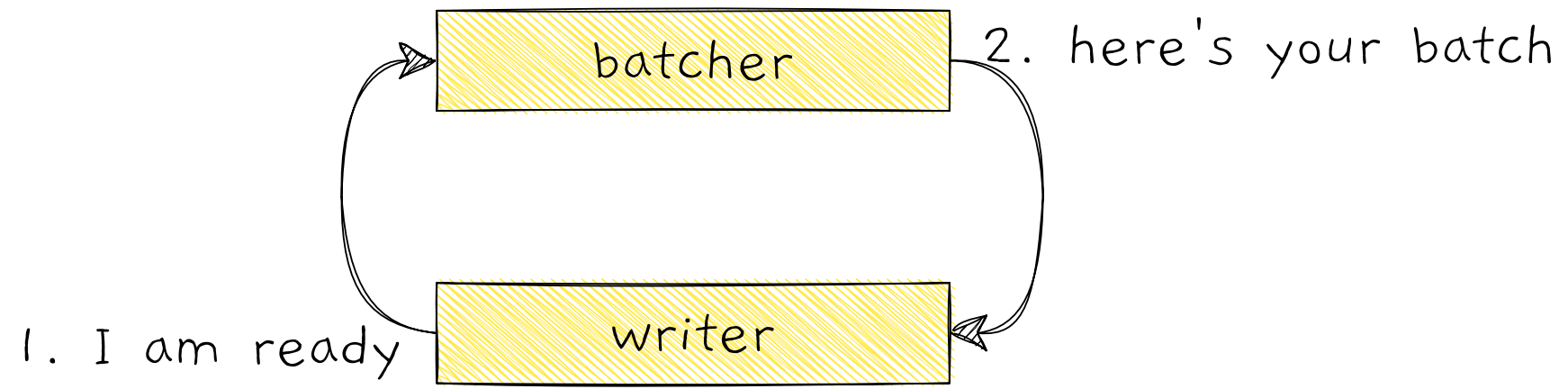
frequent db insertions



**batching**

inserting n rows at once



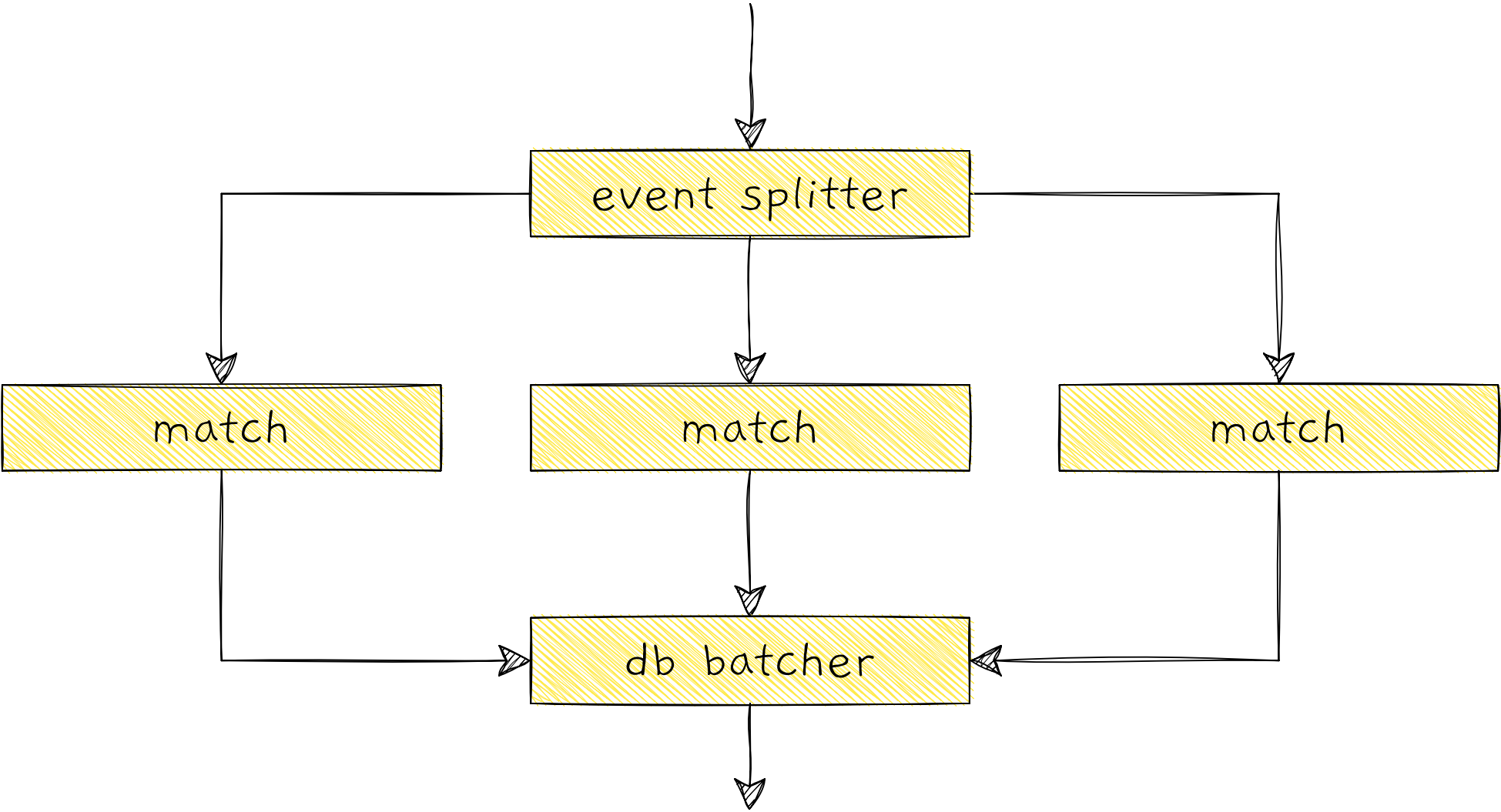


```
<event>
  <match id="1" ... >
    ...
  </match>

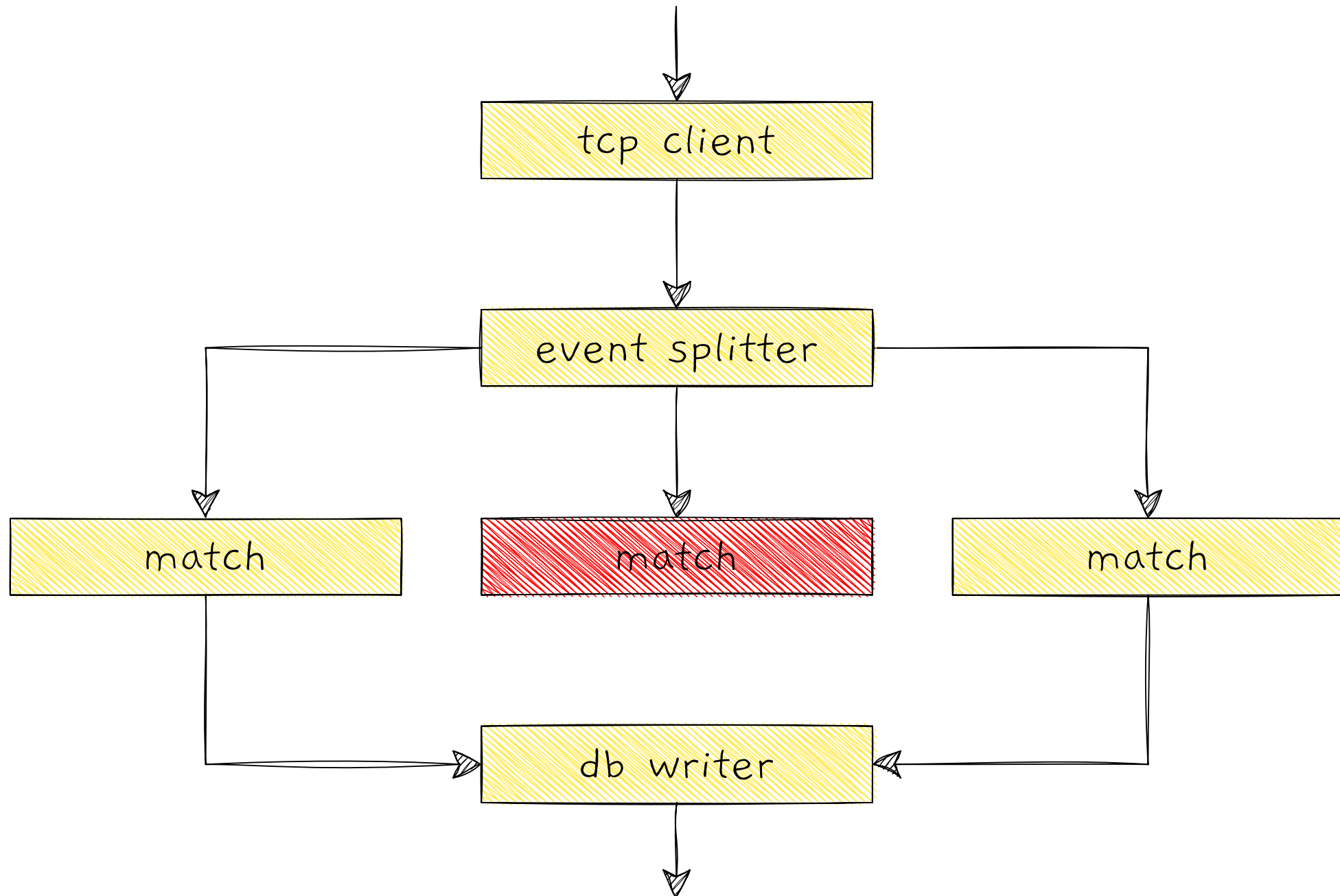
  <match id="2" ... >
    ...
  </match>

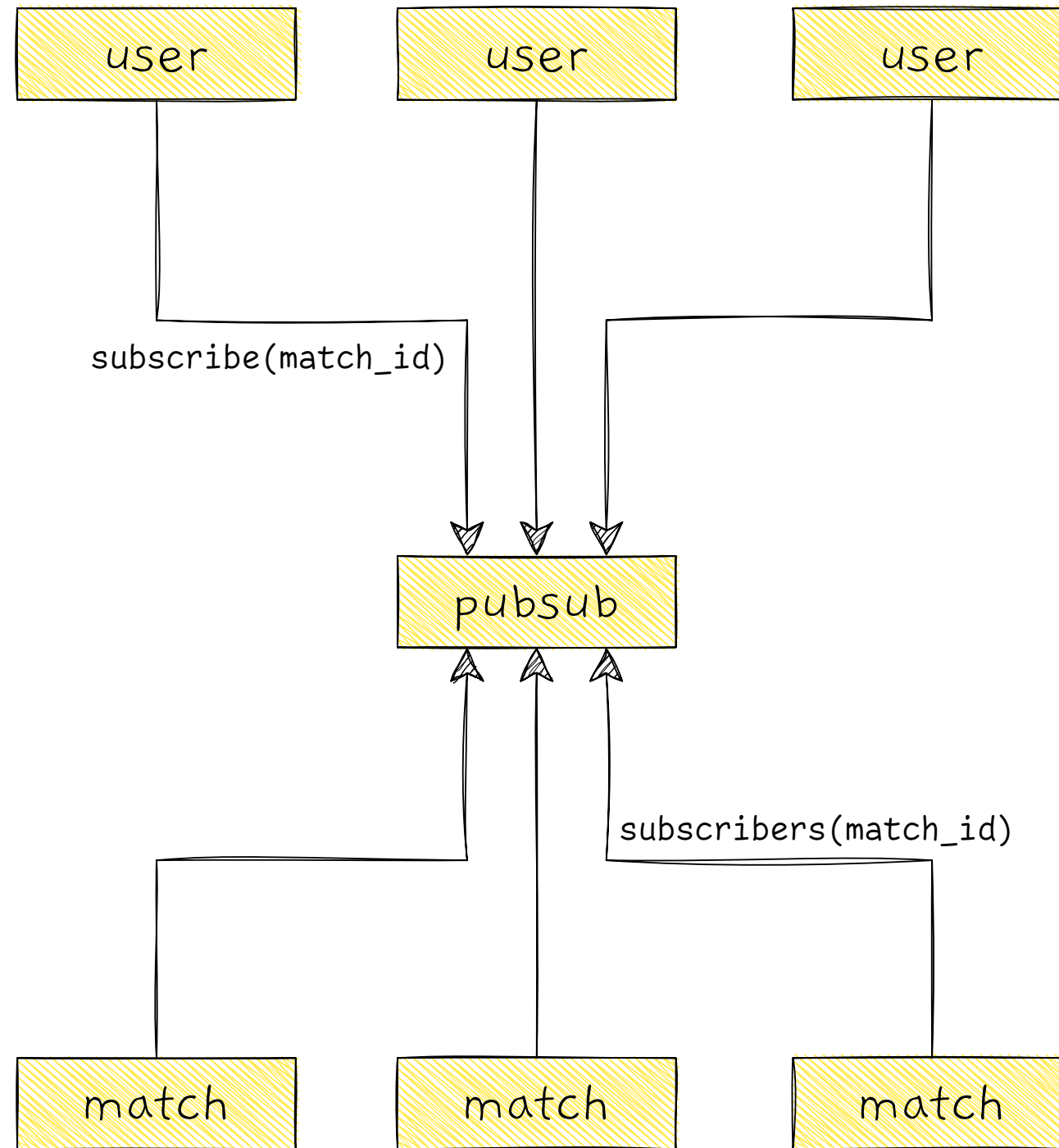
  ...

  <match id="n" ... >
    ...
  </match>
</event>
```









requirement	system A	system B
http server	Nginx	Erlang
request processing	Ruby on Rails	Erlang
long-running requests	Go	Erlang
server-wide state	Redis	Erlang
stored data	Redis and MongoDB	Erlang
background jobs	cron, bash, ruby	Erlang
crash recovery	upstart	Erlang

demo project source code

[https://github.com/sasa1977/soul\\_of\\_erlang\\_and\\_elixir](https://github.com/sasa1977/soul_of_erlang_and_elixir)

🦋 <https://bsky.app/profile/sasajuric.bsky.social>

in <https://www.linkedin.com/in/sasajuric/>

✂ <https://x.com/sasajuric>