

# The Erlang Stock Exchange



- Kenny Stone
- Trading Systems
- C/C++, Ruby, Erlang

# What is a stock exchange?

- Electronically match buyers and sellers
- “Matching engine”
- Stock exchange = matching engine for stocks

# Matching Engine

Buy 10 @ 100

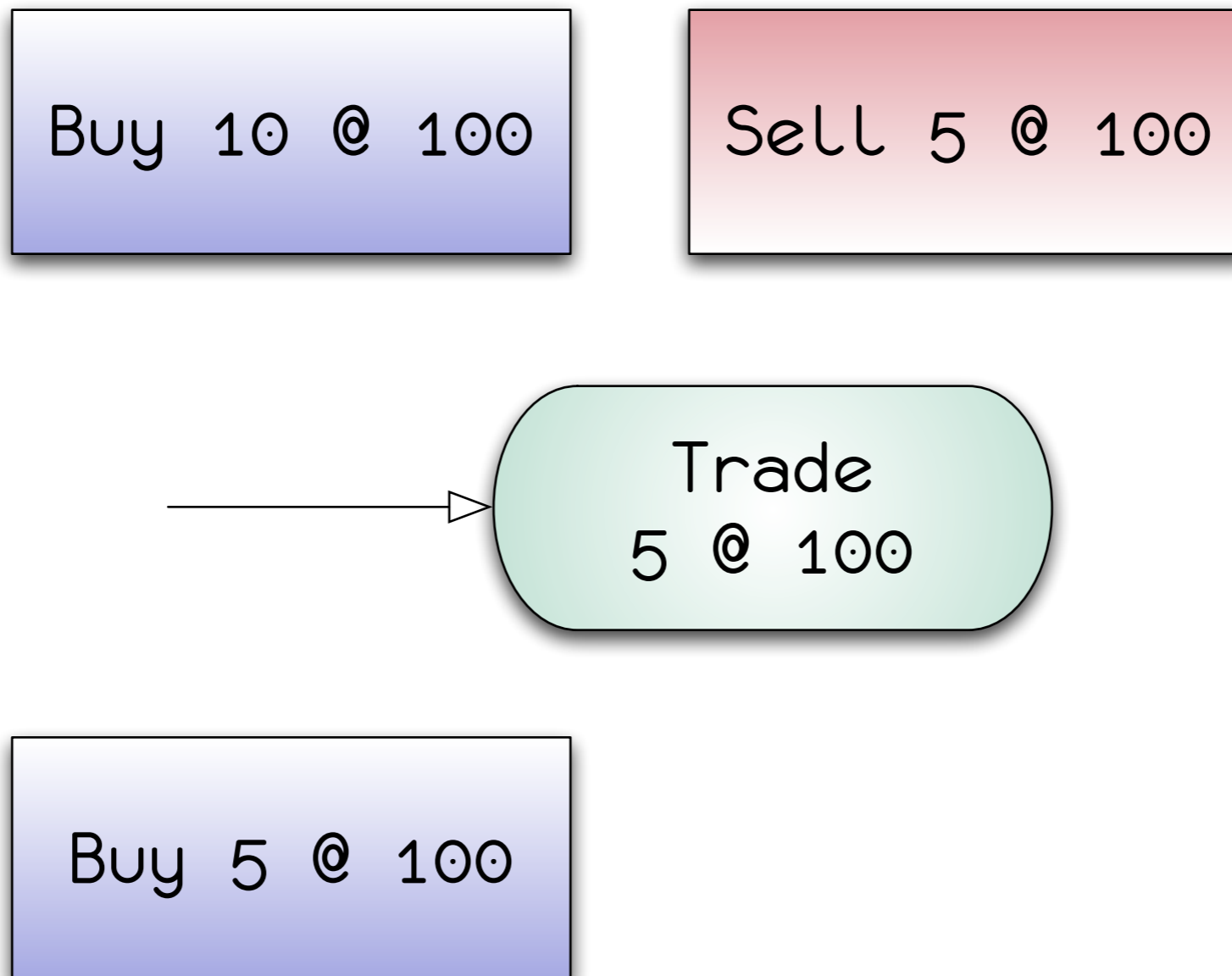
Sell 10 @ 100

Trade  
10 @ 100

```
graph TD; A[Buy 10 @ 100] --> C(Trade 10 @ 100); B[Sell 10 @ 100] --> C;
```

The diagram illustrates the matching process in a trading engine. It shows two orders at the top: a buy order for 10 units at a price of 100 (represented by a blue box) and a sell order for 10 units at a price of 100 (represented by a red box). Below these, a green rounded rectangle represents the resulting trade, which is 10 units at a price of 100. An arrow points from the left side of the trade box towards the left, indicating the flow of the trade execution.

# Matching Engine



# Keeping the Book

Buy 10 @ 110

Buy 20 @ 105

Buy 10 @ 100

# Keeping the Book

Buy 25 @ 110

Buy 10 @ 110

Sell 10 @ 110

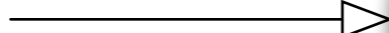
Buy 30 @ 105

Buy 20 @ 105

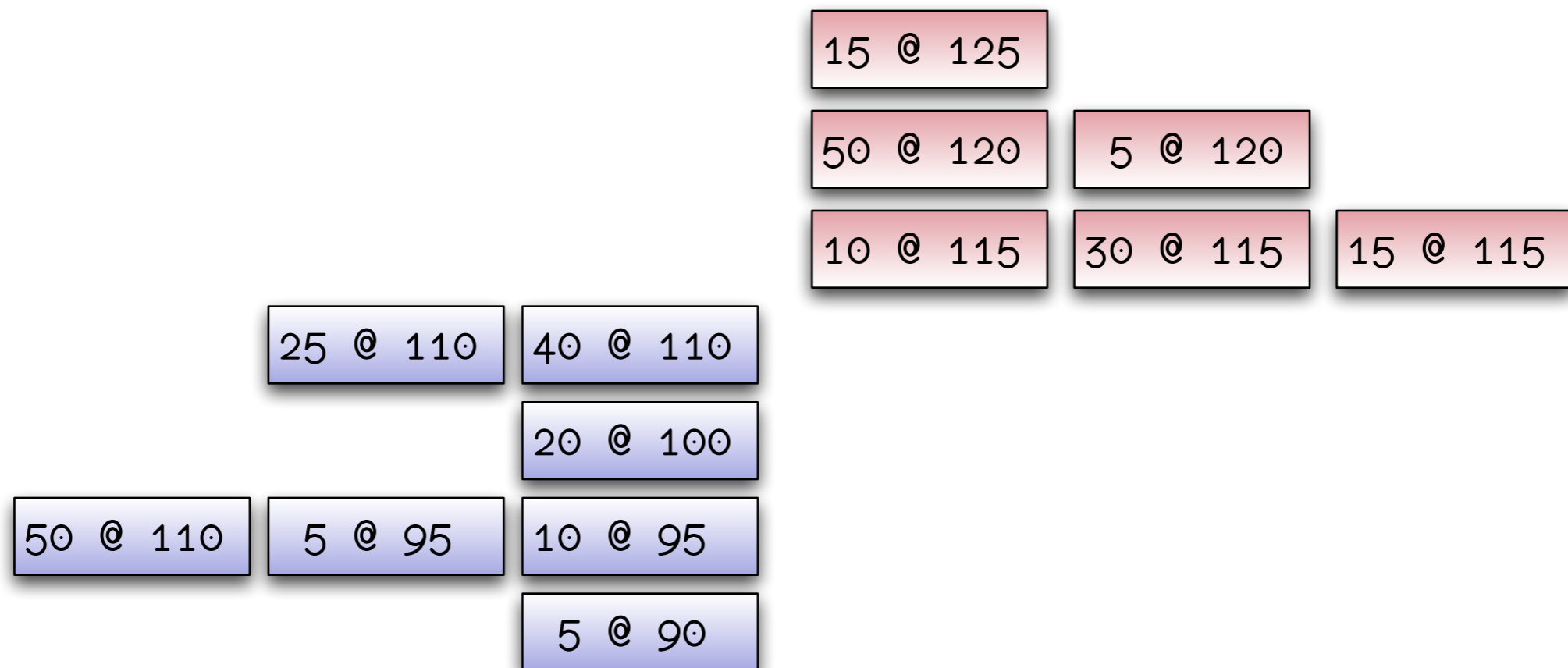
Buy 10 @ 100

105

Trade  
10 @ 110



# Keeping the Book



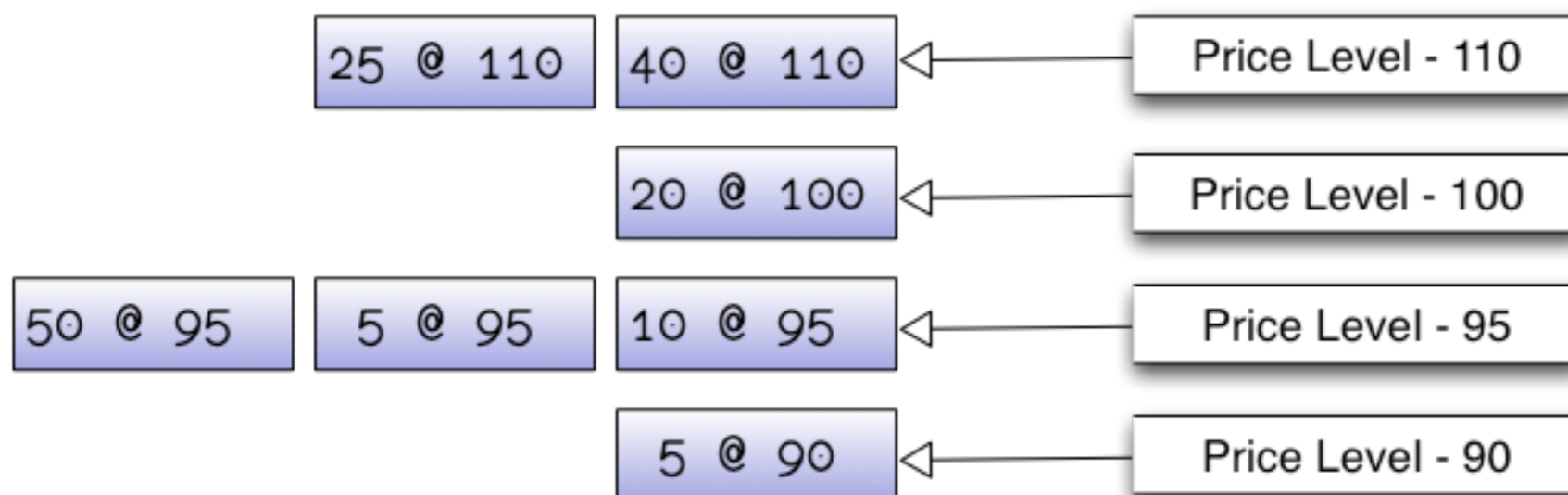


# Matching Engine

- Simple Price/Time Priority Algorithm
- Keep the book
- N actors
- Why Erlang?

# The Concurrent Engine

- Price Levels
- Insert and Match Operations can happen concurrently per price level
- Linked List of Price Levels



# The Concurrent Engine

NEW ORDER: 10 @ 95

50 @ 110

25 @ 110

40 @ 110

20 @ 100

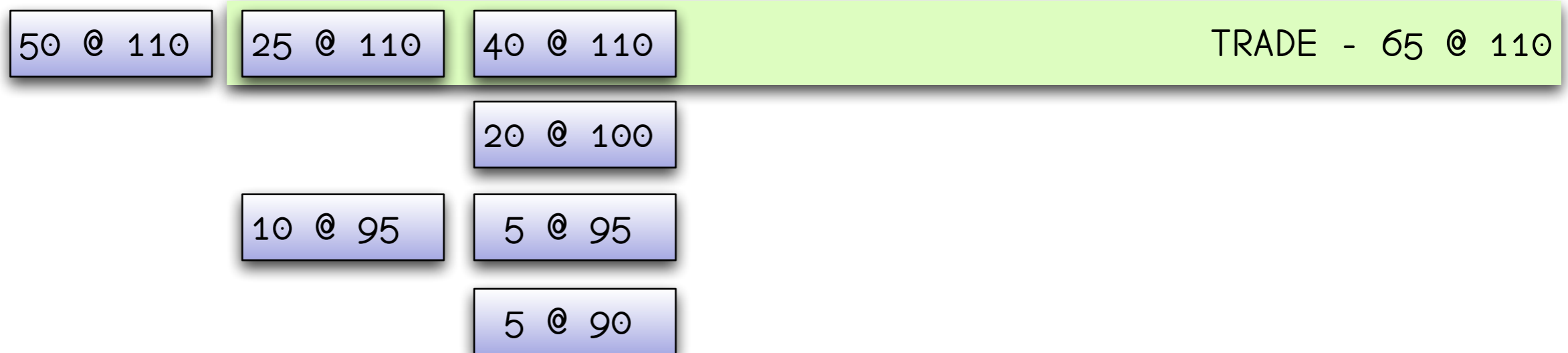
10 @ 95

5 @ 95

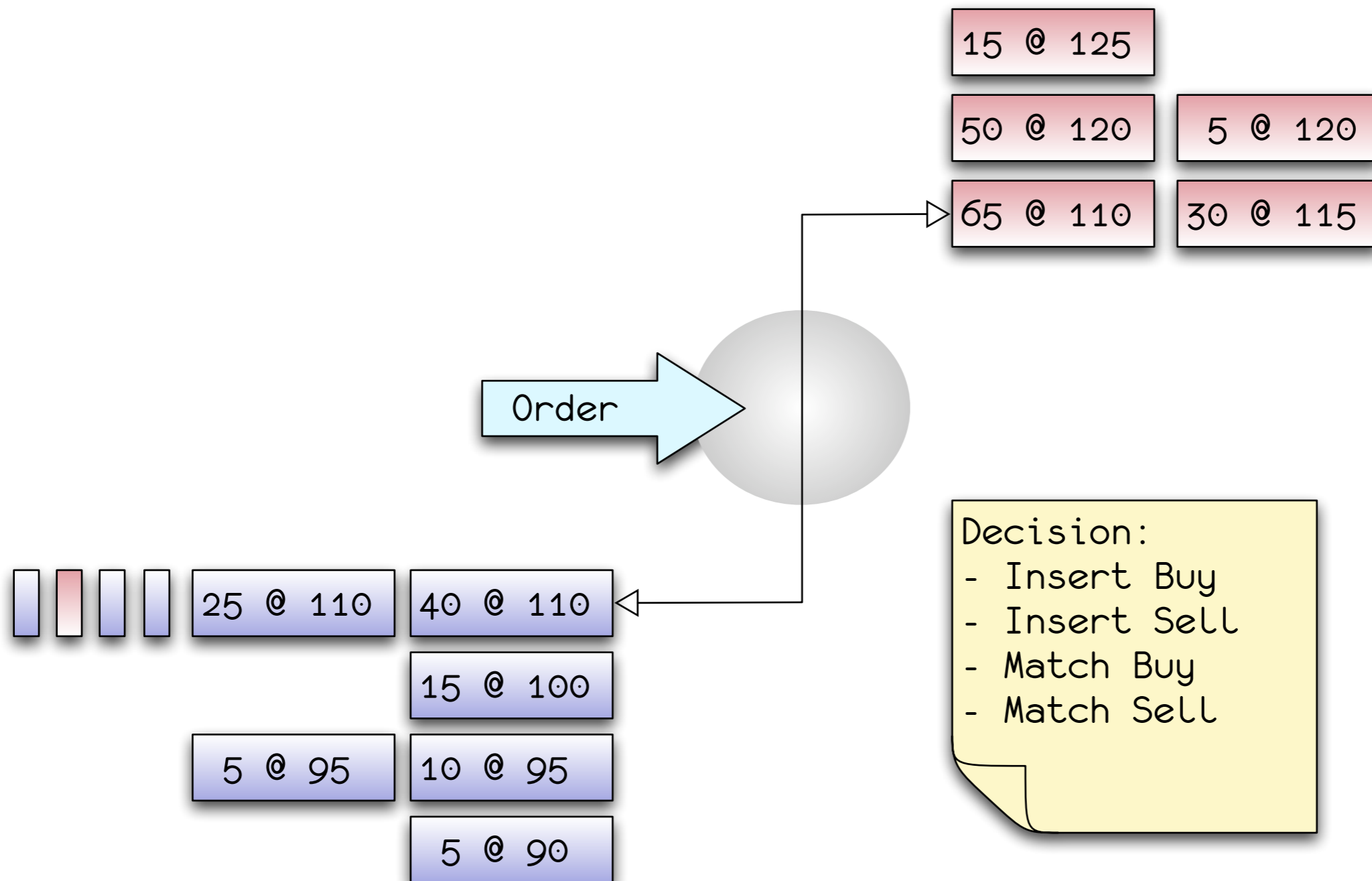
5 @ 90

# The Concurrent Engine

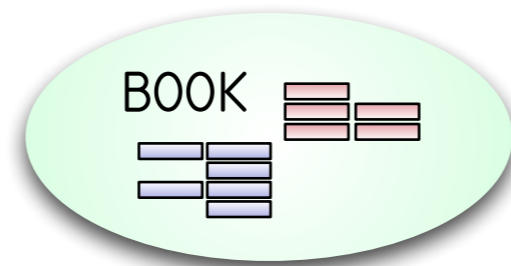
NEW ORDER: 65 @ 110



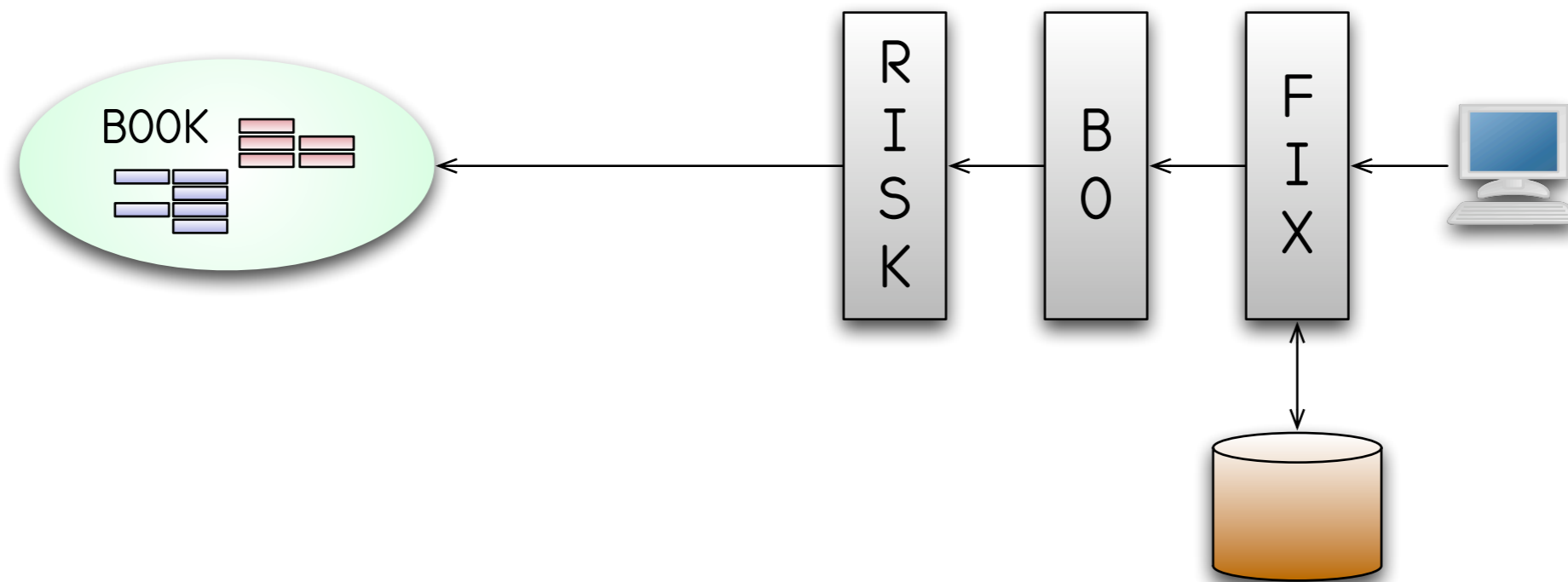
# The Concurrent Engine



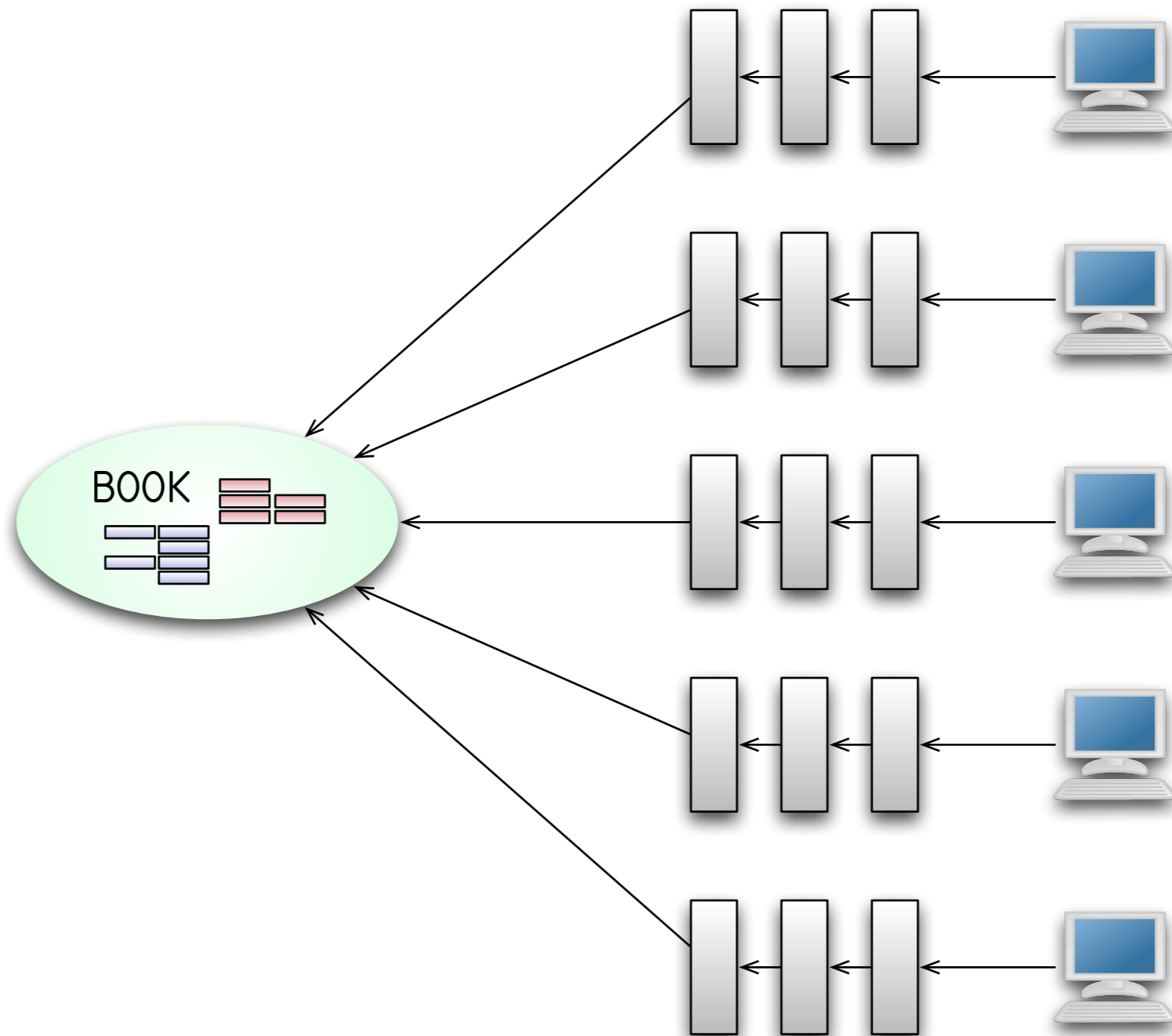
# The Concurrent Engine



# The Concurrent Engine

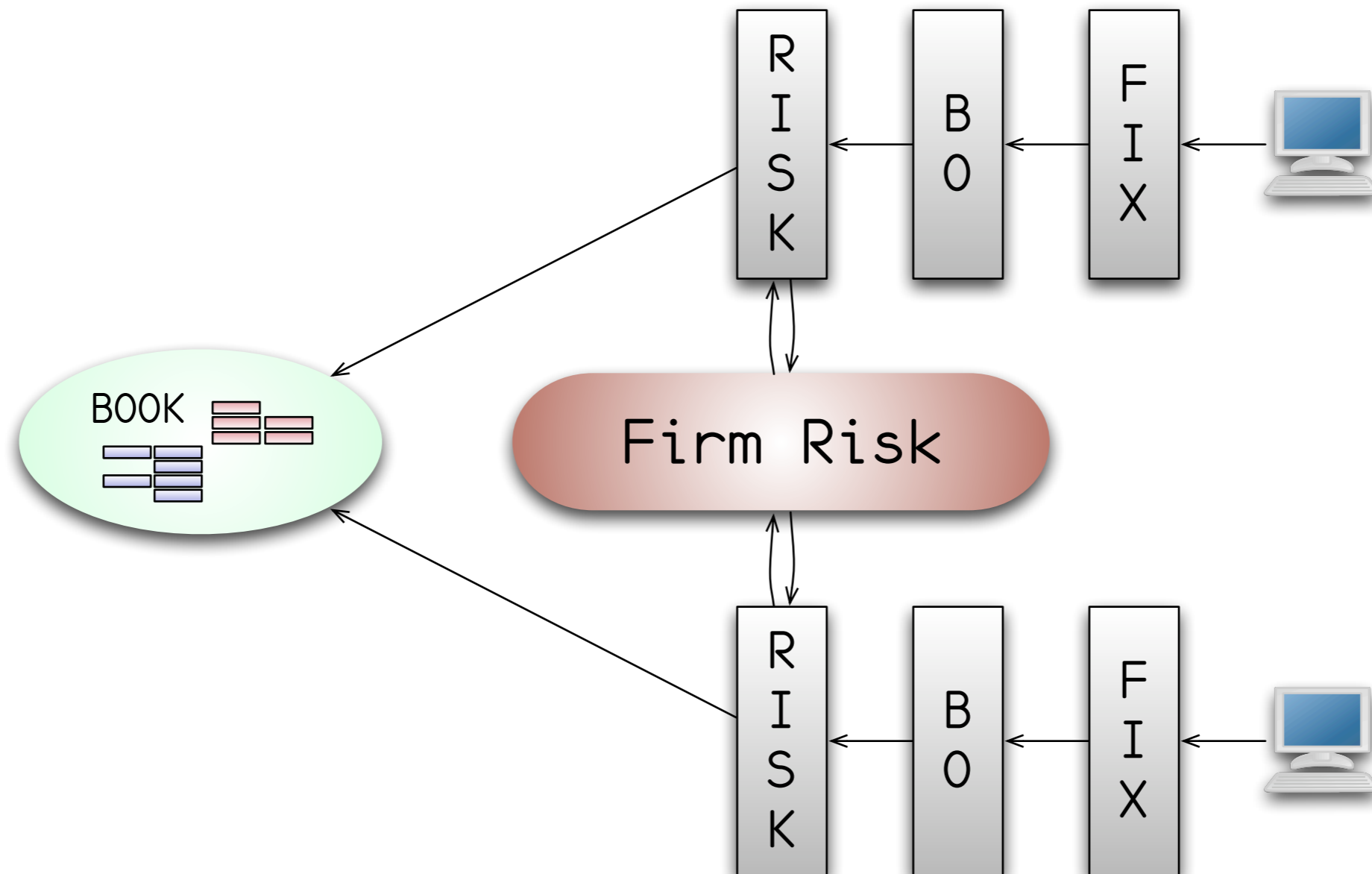


# The Concurrent Engine

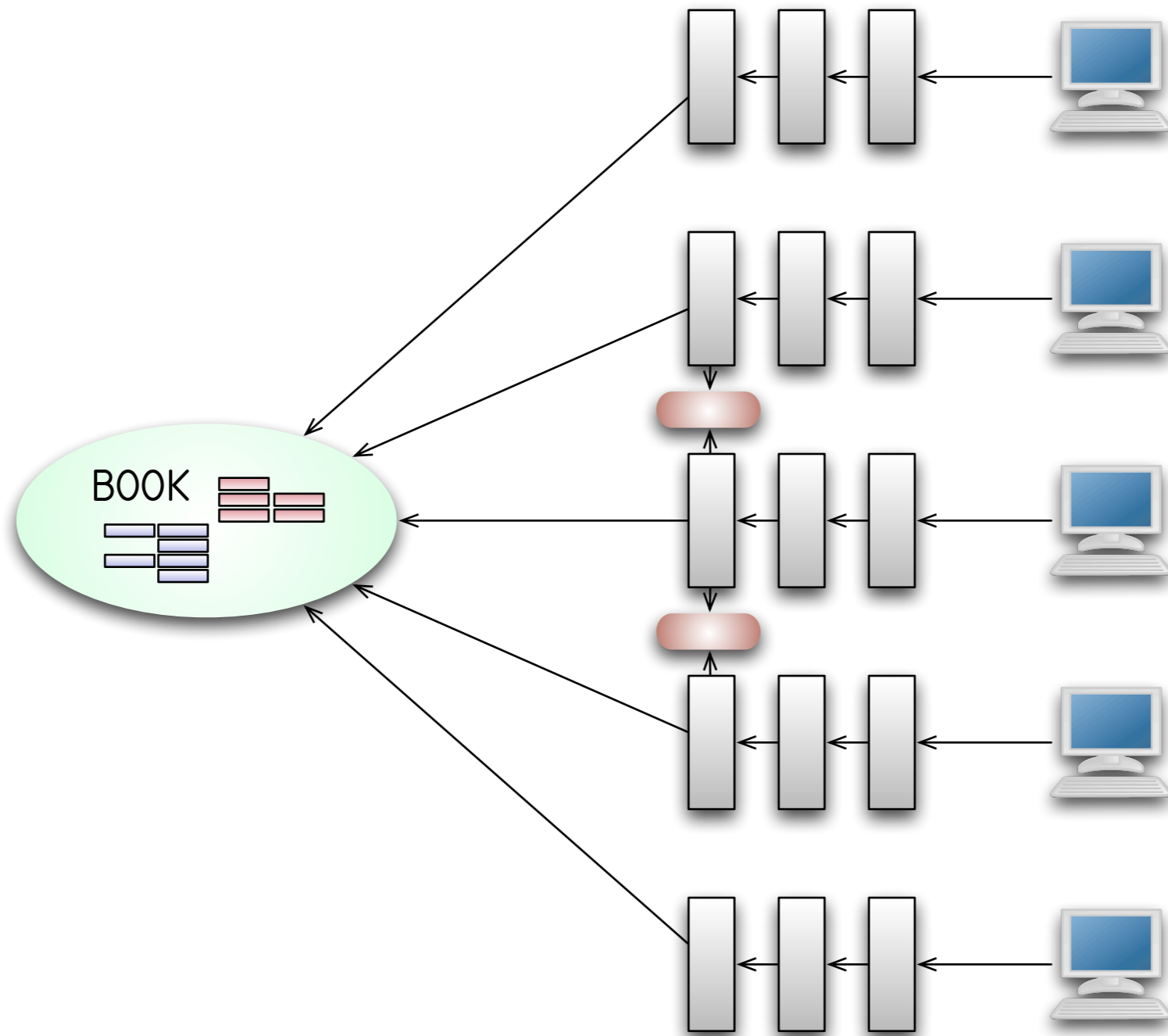




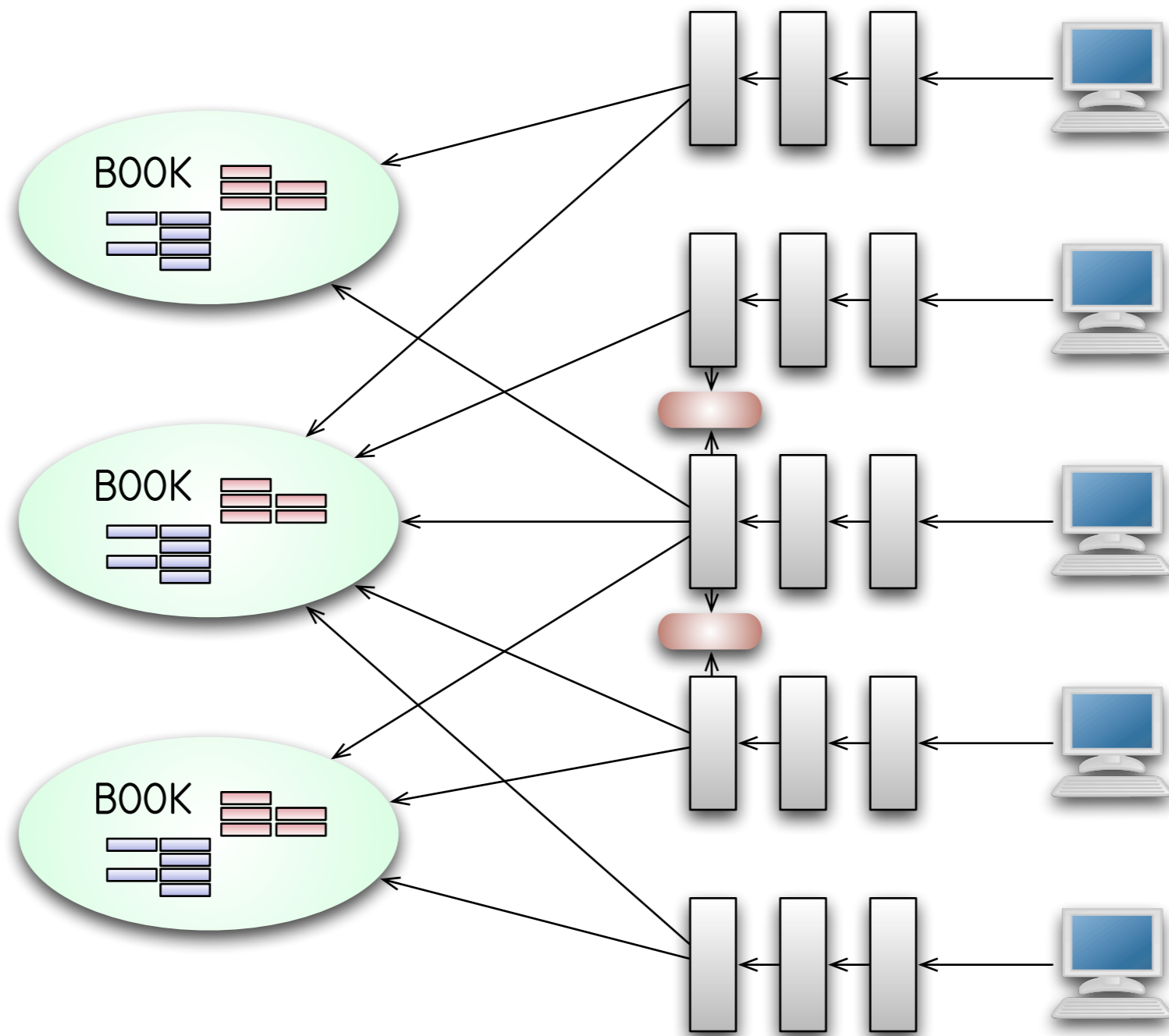
# The Concurrent Engine



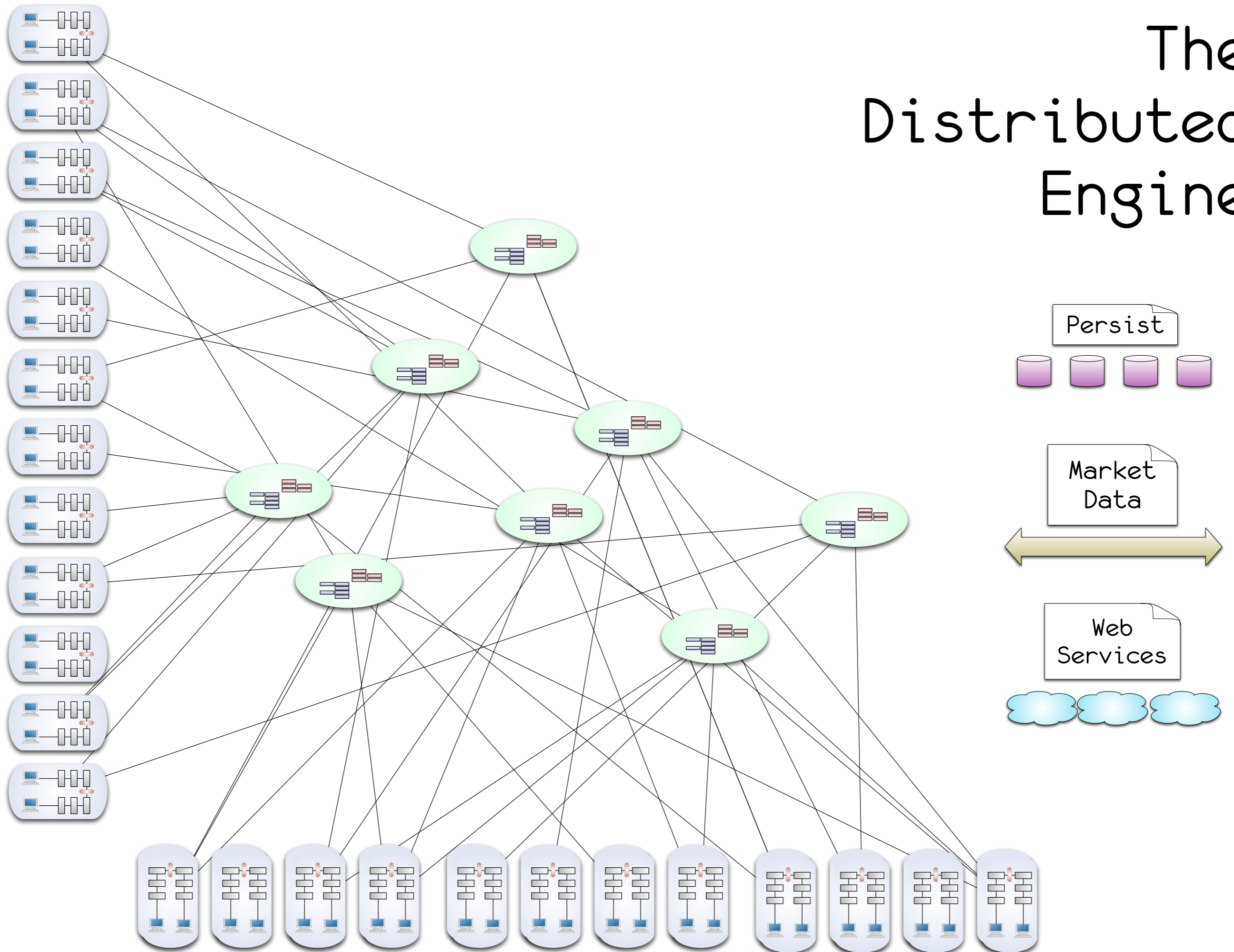
# The Distributed Engine



# The Distributed Engine



# The Distributed Engine

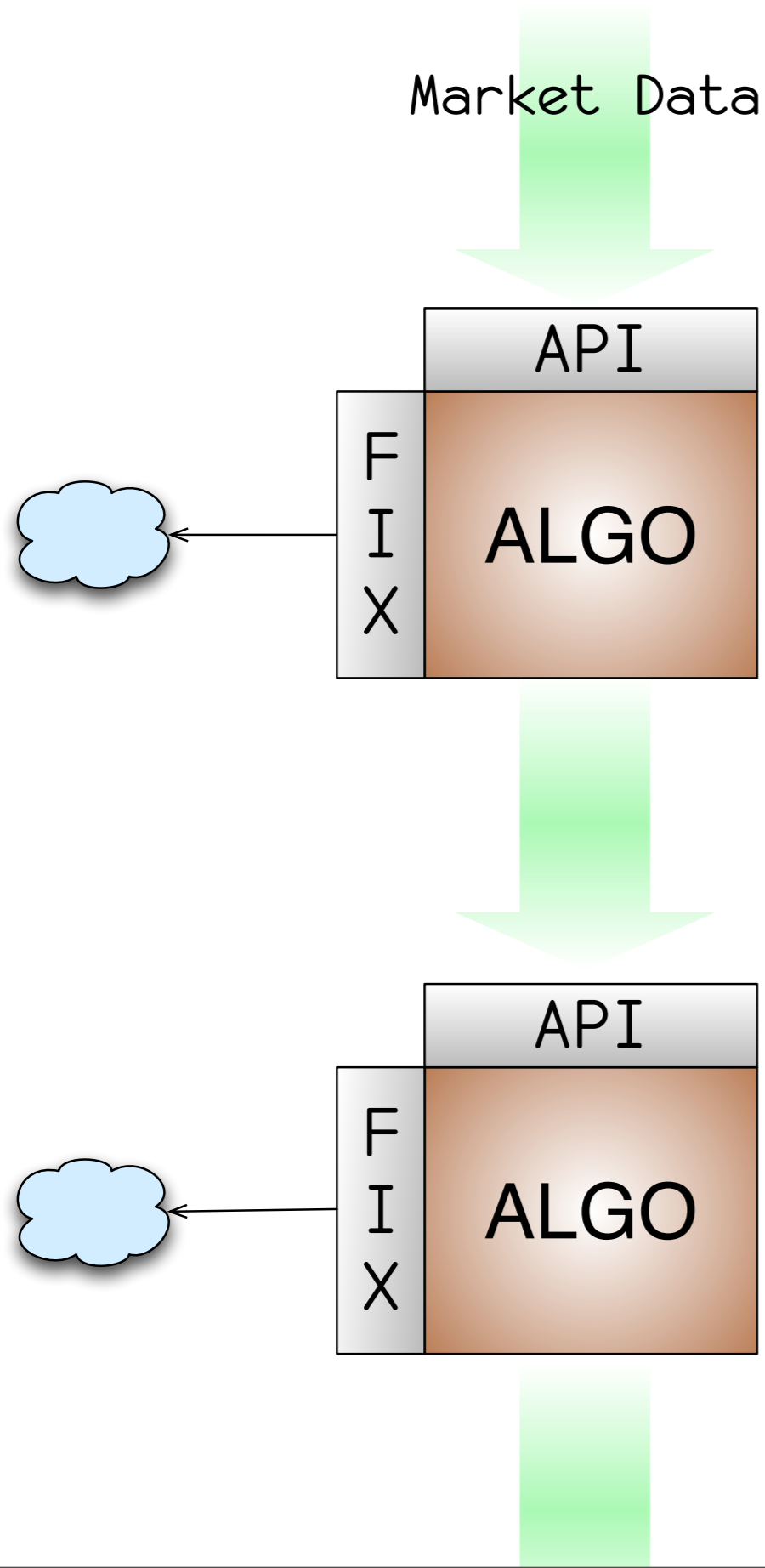


By using Erlang,  
I can build one and  
scale to many

# The Trading Industry

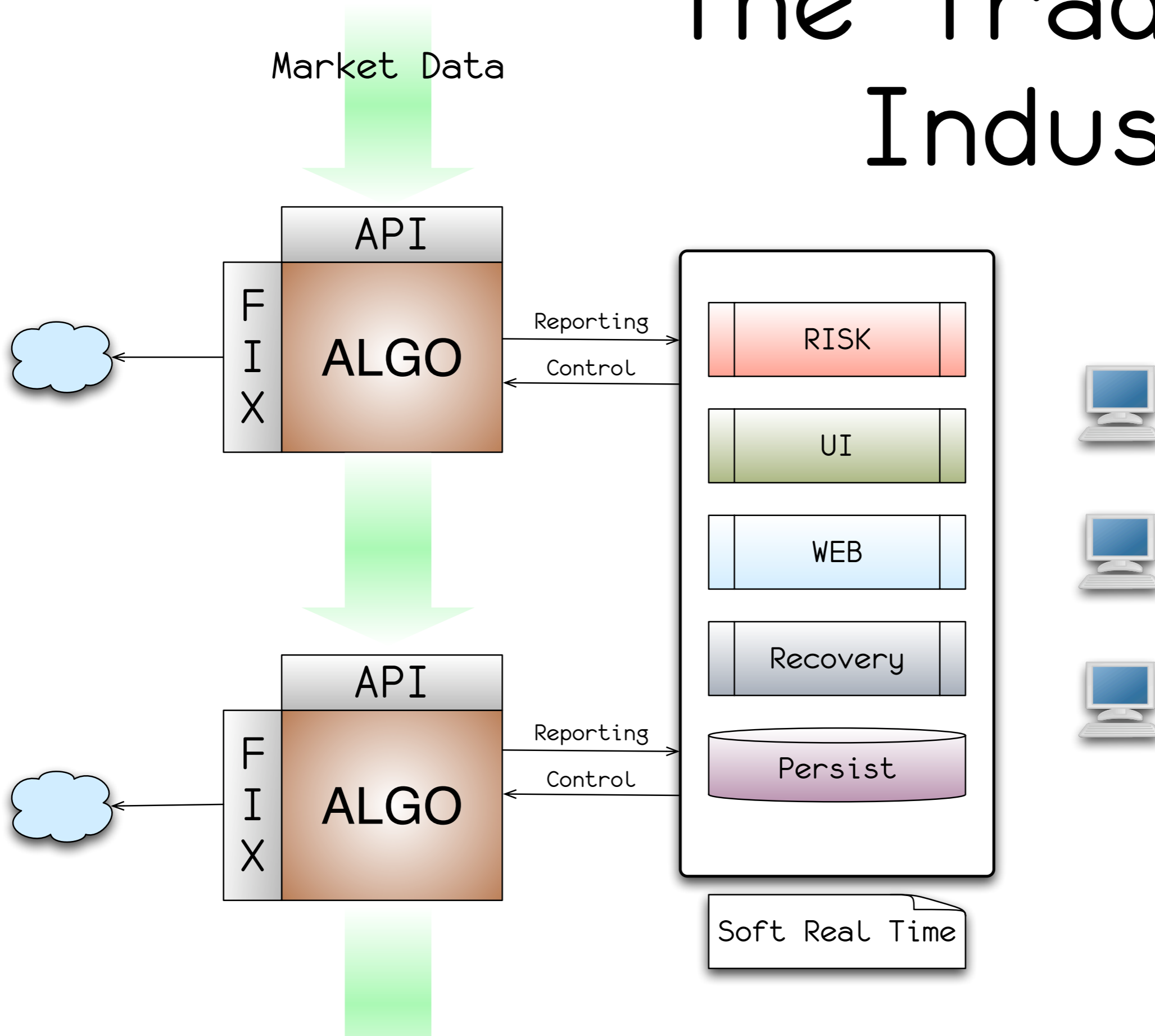
- High frequency, low latency
- $\mu$ Seconds matter
- The critical path cannot be compromised

# The Trading Industry



- Market Data Input, Orders Output
- Critical path can be very latency sensitive
- Need systems around Algo engine

# The Trading Industry





# The Trading Industry

- Challenges of Erlang
  - It's very different.
  - Customers aren't going to ask for it.
  - Developers don't know anything about it.
- Software is about choosing the right tools.
- A business is about execution.

# The Trading Industry

- What we're building:
  - Highly event driven systems
  - Distributed systems
  - Systems that scale
  - Systems that are fault tolerant



connamara  systems, LLC

# Thanks

Kenny Stone  
kstone@connamara.com

