The Road to Running Haskell at Facebook Scale

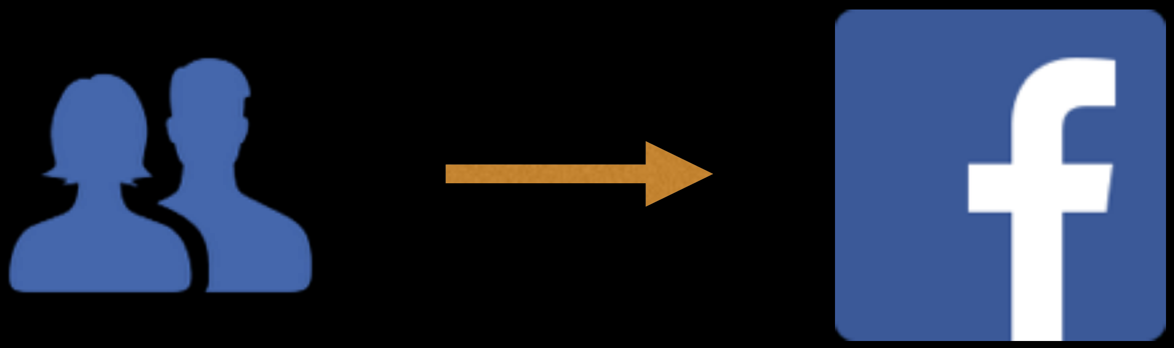
Jon Coens
Haskell Shepherd at Facebook

Haskell is ready for industry

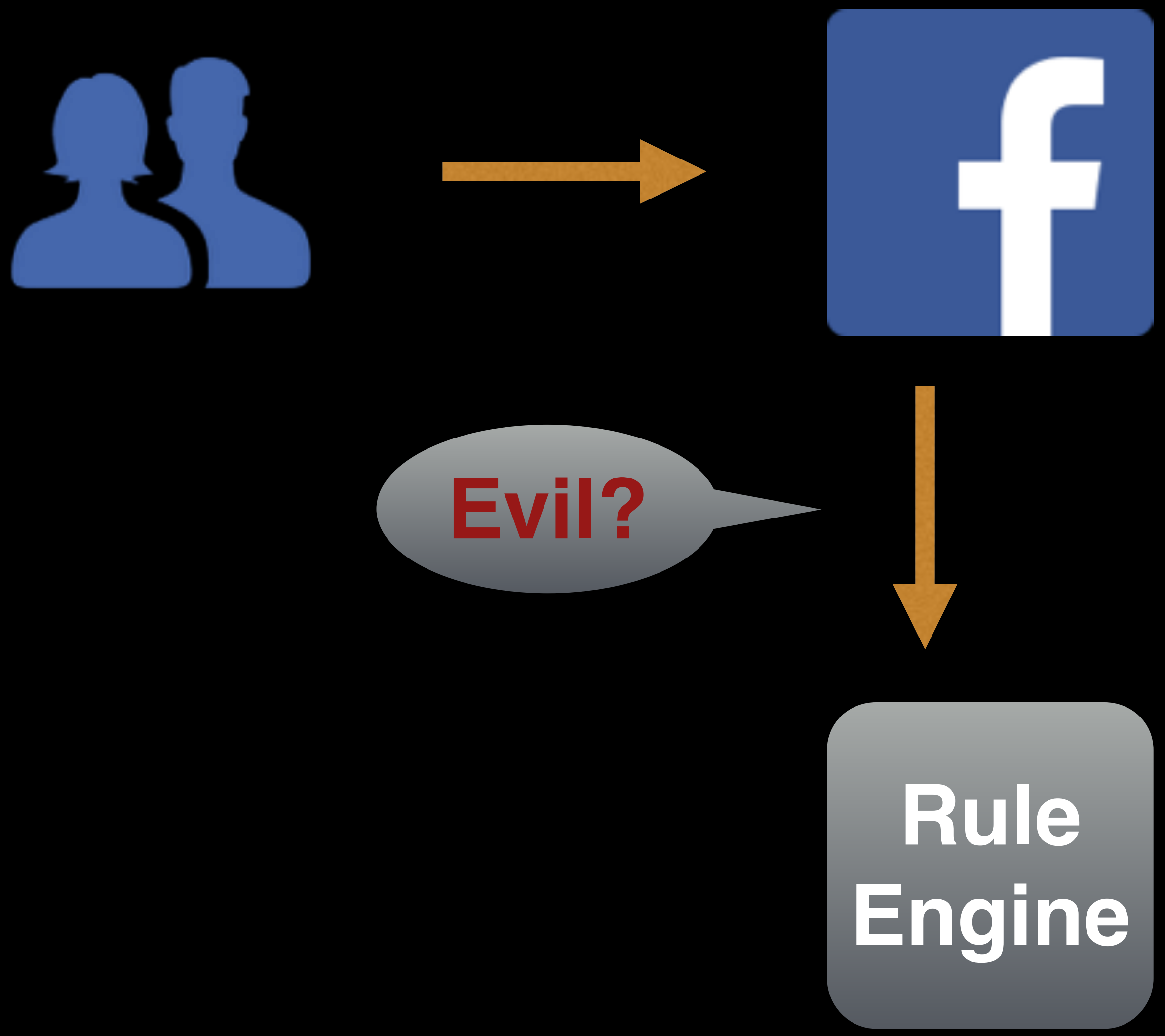
Journey to Protect Facebook with Haskell

Our World

Our World



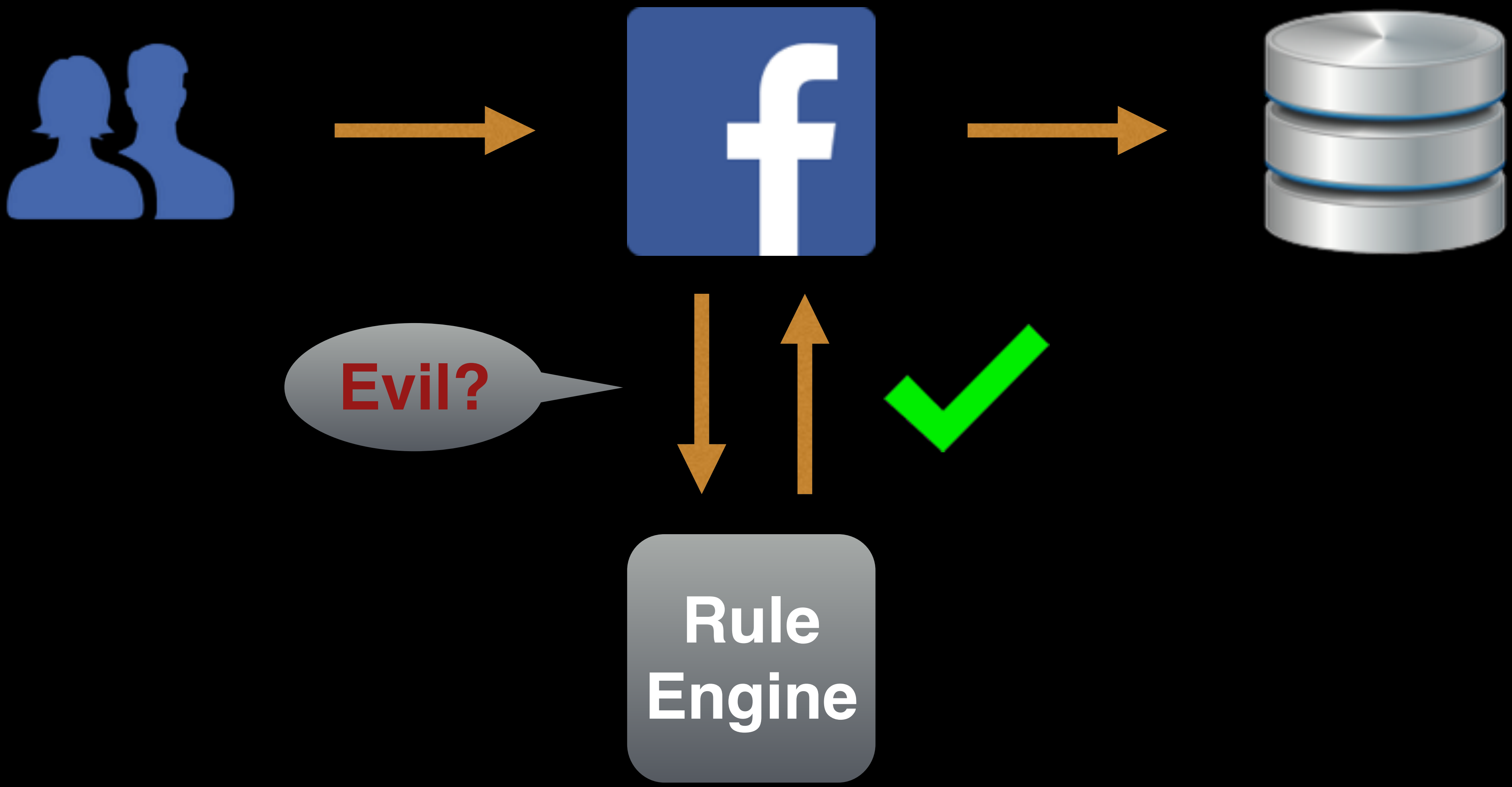
Our World



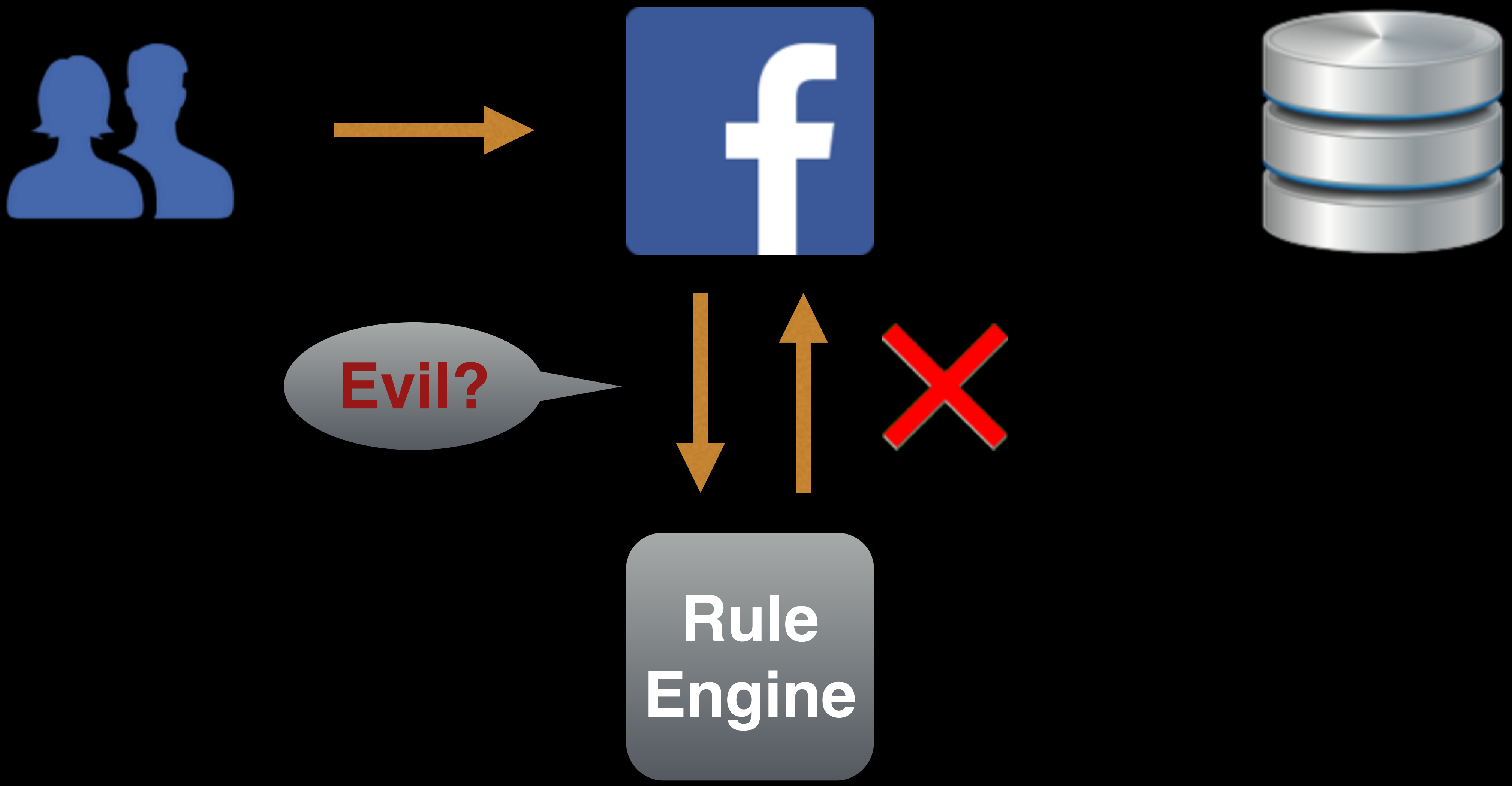
Evil?

Rule Engine

Our World



Our World



Evil?

Rule Engine

We want a rule that says

- If someone is posting about Monads
- And they have more than 100 friends
- And more than half of those friends like C++
- Then block, else allow

We want a rule that says

Need info about content

- If someone is posting about Monads
- And they have more than 100 friends
- And more than half of those friends like C++
- Then block, else allow

We want a rule that says

Fast

- If someone is posting about Monads
- And they have more than 100 friends
- And more than half of those friends like C++
- Then block, else allow

We want a rule that says

Fast

- If someone is posting about Monads
- And they have more than 100 friends

Need to fetch the friend list

- And more than half of those friends like C++
- Then block, else allow

We want a rule that says

Fast

- If someone is posting about Monads
- And they have more than 100 friends

Slow

- And more than half of those friends like C++
- Then block, else allow

We want a rule that says

Fast

- If someone is posting about Monads
- And they have more than 100 friends

Slow

- And more than half of those friends like C++

Need info about each friend

- Then block, else allow

We want a rule that says

Fast

- If someone is posting about Monads
- And they have more than 100 friends

Slow

- And more than half of those friends like C++

Very Slow

- Then block, else allow

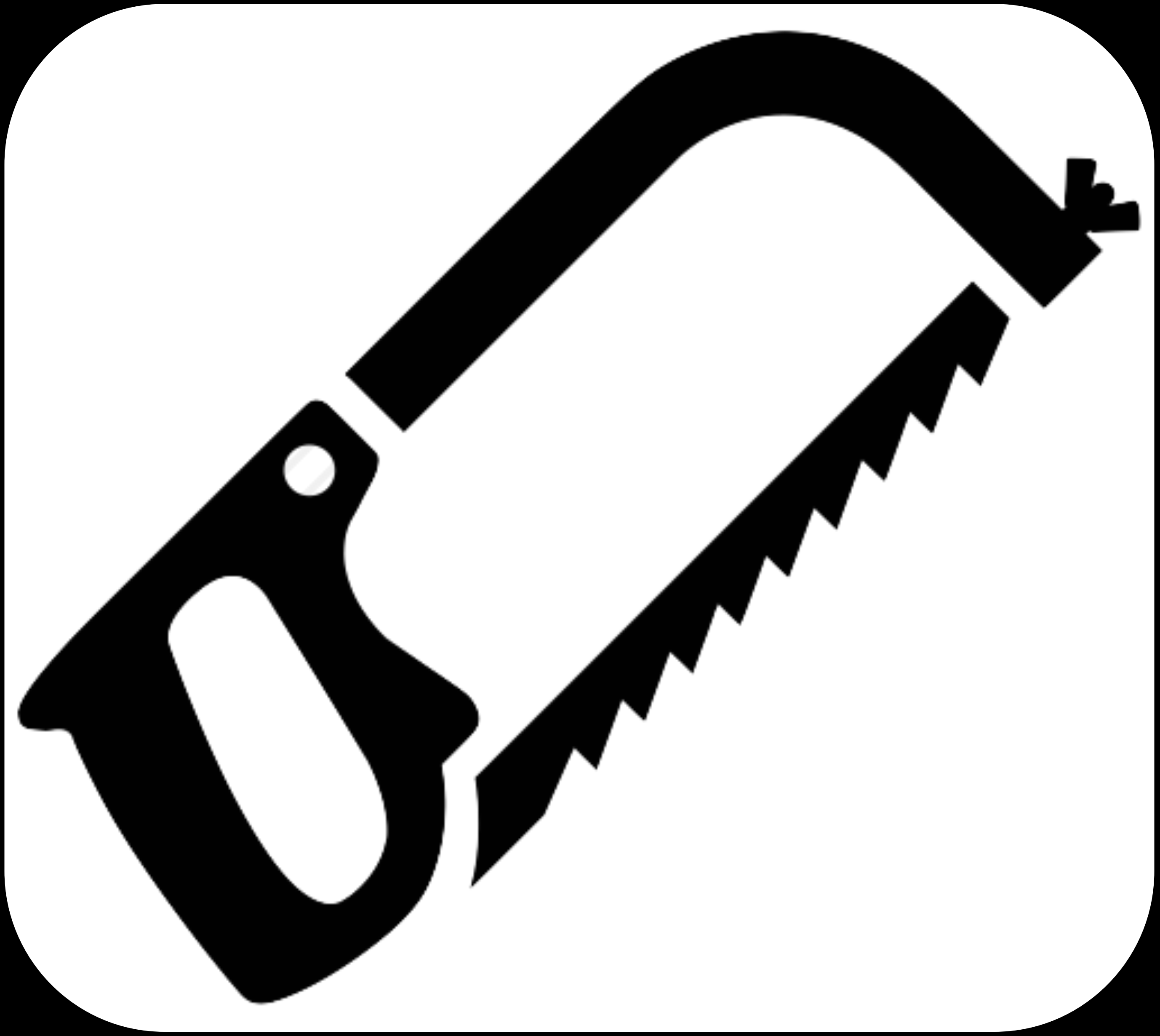
Rule Engine Language

Pure functions + Data fetching

Rule Engine Language

Pure functions + Data fetching

What did we build?



Requirements

- Latency-sensitive
- Complex expressive application logic
- Ship new code ASAP
- 100K reqs/sec

Built an Interpreter

- Latency-sensitive
- Complex expressive application logic
- Ship new code ASAP
- 100K reqs/sec

Built an Interpreter

Batch data fetches at will

- Latency-sensitive
- Complex expressive application logic
- Ship new code ASAP
- 100K reqs/sec

Built an Interpreter

Batch data fetches at will

No concurrency constructs

- Latency-sensitive
- Complex expressive application logic
- Ship new code ASAP
- 100K reqs/sec

Built an Interpreter

- Latency-sensitive

Batch data fetches at will

- Complex expressive application logic

No concurrency constructs

- Ship new code ASAP

Build dynamic loader

- 100K reqs/sec

Built an Interpreter

- Latency-sensitive

Batch data fetches at will

- Complex expressive application logic

No concurrency constructs

- Ship new code ASAP

Build dynamic loader

- 100K reqs/sec

Fast enough on our hardware

Scale!

???? Million reqs / sec

Scale!

- Hardware won’t catch-up
- Limited by language constructs

```
MapMap2(kv_grip,map) =
  Let
    kgrip(k) = kv_grip(k, MapGet(map, k))
  In
    VMap(kgrip, MapKeys(map));

MapItems(map) = MapMap2(
  \(a,b) = Pair(a,b),
  map);
```

We need a rewrite

If writing this from scratch today,
what would we build?

Batching Data Fetches

```
// find common friends
intersect(friendsOf(x), friendsOf(y))

// classic N+1 SELECTs problem
map(accountAge, friendsOf(x))

// ...and combinations of
map(accountAge, intersect(friendsOf(x), friendsOf(y))
```

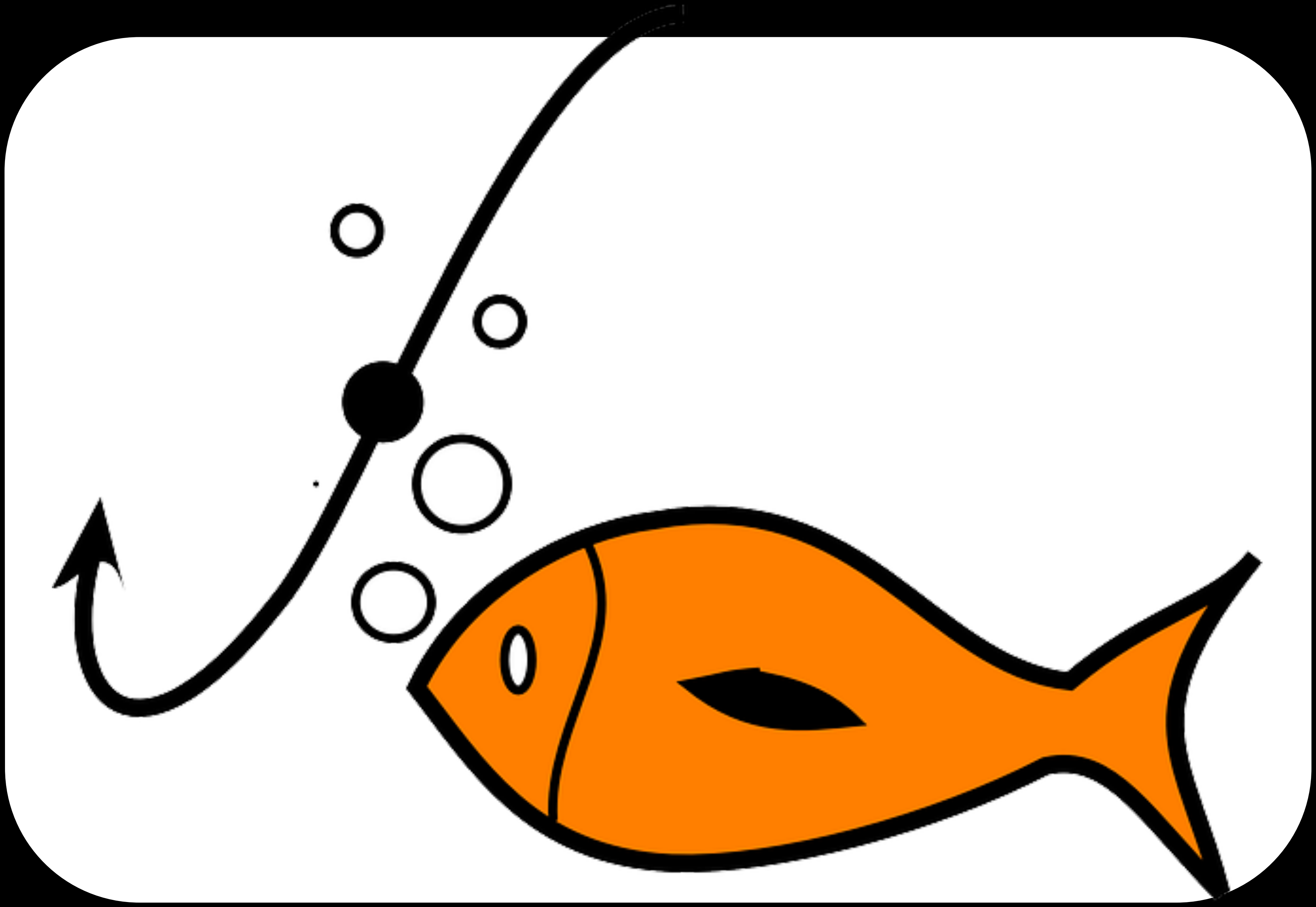
No Concurrency Constructs

```
// find common friends
intersect(friendsOf(x), friendsOf(y))

// classic N+1 SELECTs problem
map(accountAge, friendsOf(x))

// ...and combinations of
map(accountAge, intersect(friendsOf(x), friendsOf(y))
```

Writing expressive code in your favorite language that auto-batches I/O is almost impossible

* consider this a challenge

Write Your Own Language?



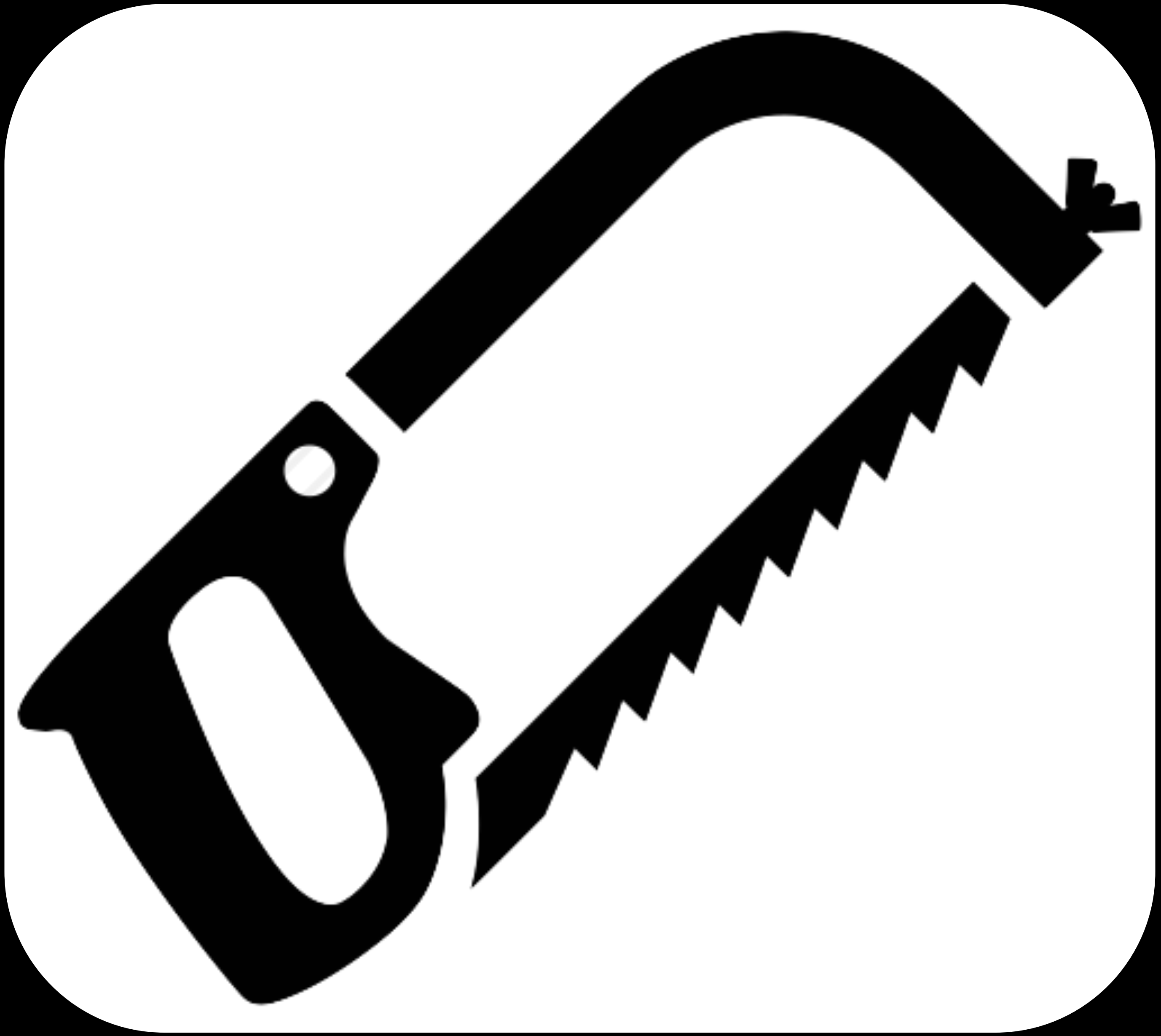
Don’t Write Your Own Language

- Implement your killer feature
- Profilers, debuggers, REPLs
- Libraries
- Maintenance forever

“Reordering execution of the syntax tree”

Is there anything performant for running DSLs?

What should we build?



Haskell

- Latency-sensitive
- Complex expressive application logic
- Ship new code ASAP
- > 1M reqs/sec

Batch Data Fetches

http://github.com/facebook/Haxl

Expressive I/O Batching

```
-- “naive” friendsOf
friendsOf :: Id -> [Id]

-- Haxl friendsOf
friendsOf :: Id -> Haxl [Id]


-- “naive” multi-fetch
map friendsOf [id1, id2, …]

-- Haxl multi-fetch
mapM friendsOf [id1, id2, …]
```

Haskell

Haxl

- Latency-sensitive

Haxl

- Complex expressive application logic
- Ship new code ASAP
- > 1M reqs/sec

Ship new code ASAP

- GHC runtime has a built-in linker
- Needed to implement unloading
- Run new code without restarting server

Haskell

- Latency-sensitive

Haxl

- Complex expressive application logic

Haxl

- Ship new code ASAP

GHC Linker

- > 1M reqs/sec

Haskell

- Latency-sensitive

Haxl

- Complex expressive application logic

Haxl

- Ship new code ASAP

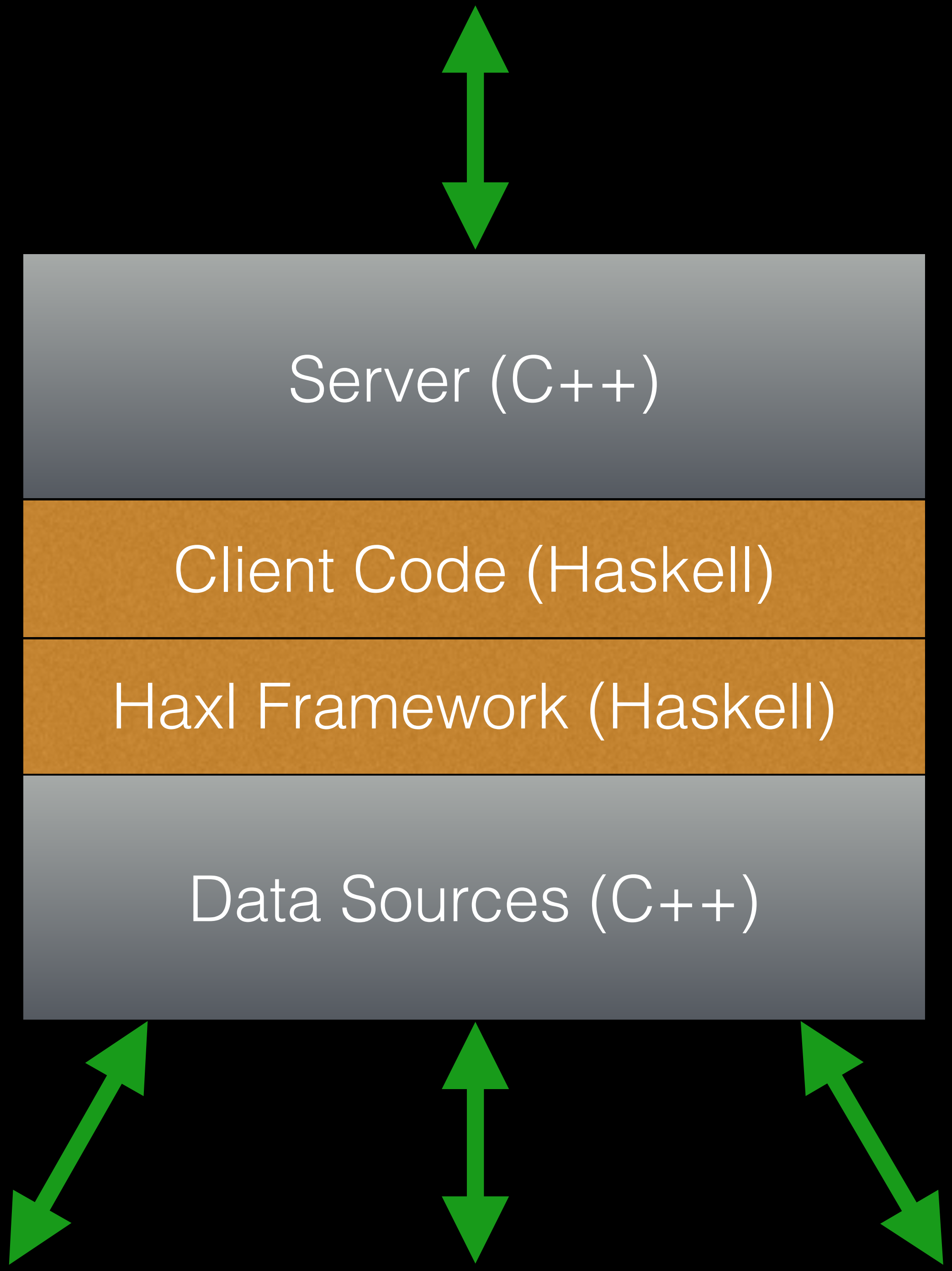
GHC Linker

- > 1M reqs/sec

?

Foreign Function Interface

Foreign Function Interface



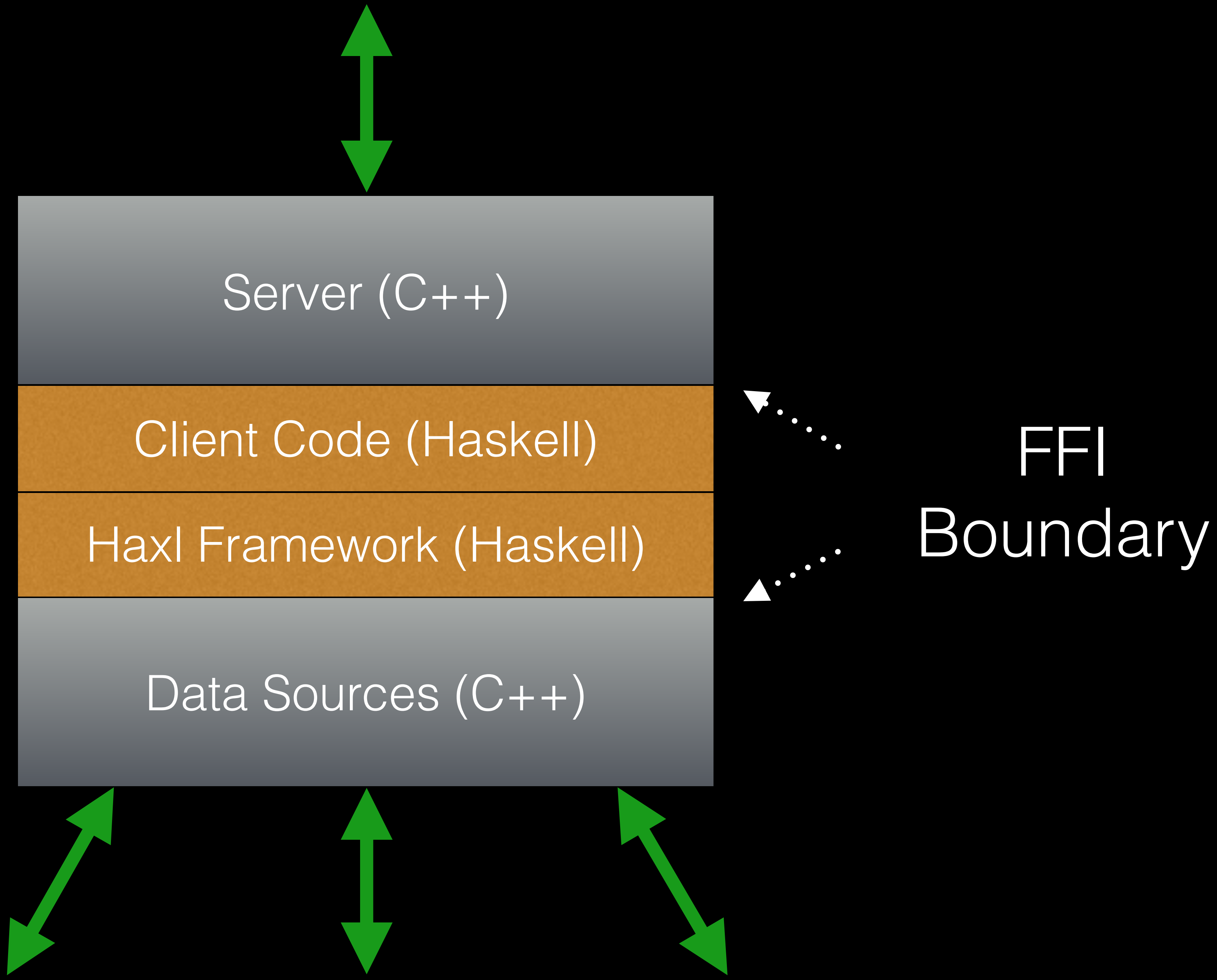
Server (C++)

Client Code (Haskell)

Haxl Framework (Haskell)

Data Sources (C++)

Foreign Function Interface



Server (C++)

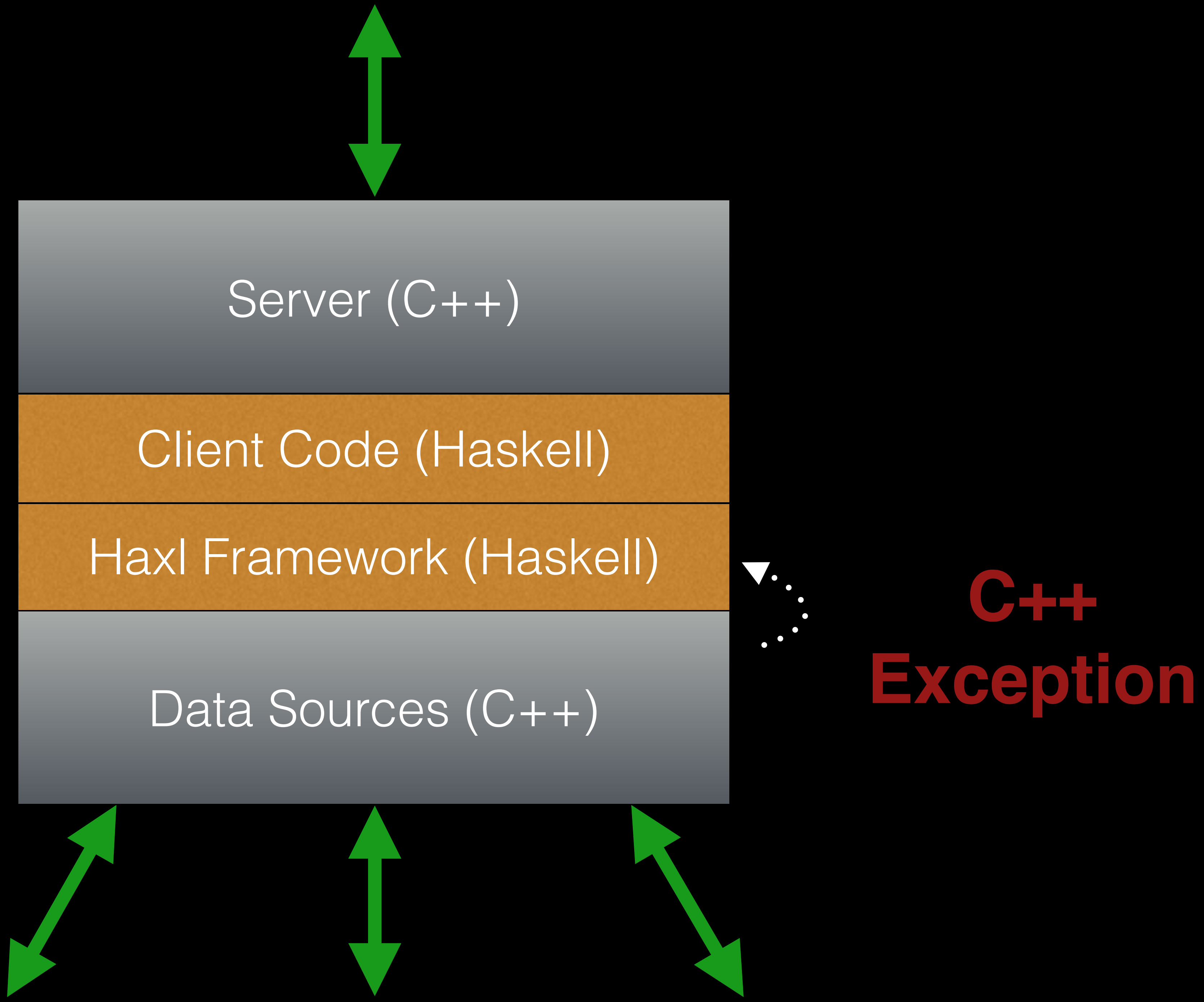
Client Code (Haskell)

Haxl Framework (Haskell)

Data Sources (C++)

FFI Boundary

Foreign Function Interface



Foreign Function Interface

Exception handling can be tricky

* ripe place to make Haskell better

Foreign Function Interface

```
foreign import ccall unsafe “countAardvarks”
  countAardvarks :: Int -> CString -> IO Int
```

Foreign Function Interface

😦

```
foreign import ccall unsafe “goOutToLunch”
  goOutToLunch :: Int -> CString -> IO Int
```

Foreign Function Interface

```
foreign import ccall safe “goOutToLunch”
  goOutToLunch :: Int -> CString -> IO Int
```

Foreign Function Interface

☹️

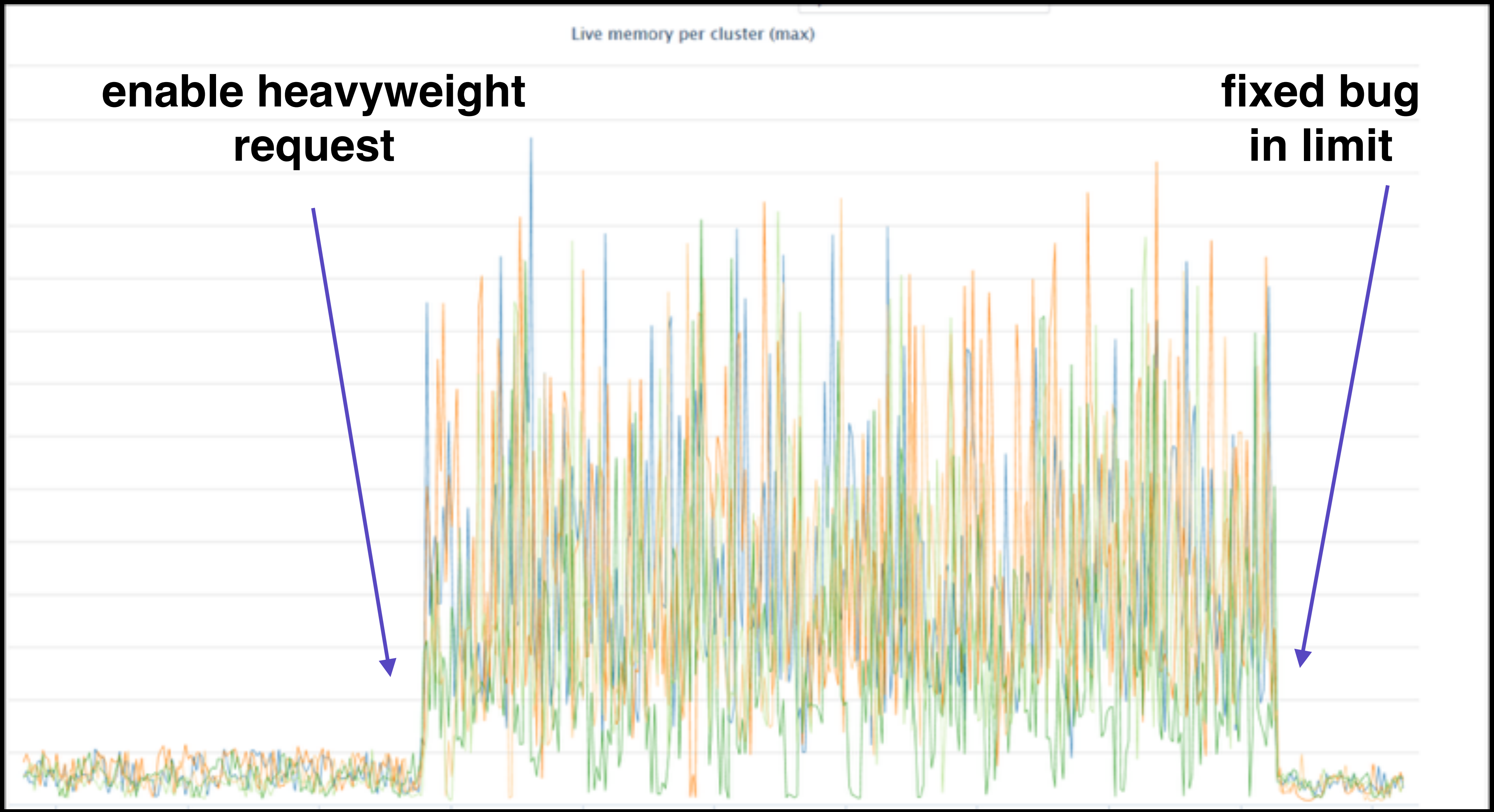
```
foreign import ccall safe “countAardvarks”
  countAardvarks :: Int -> CString -> IO Int
```

Foreign Function Interface

For best performance, you need a balance of safe and unsafe calls

Allocation Limits

Allocation Limits



enable heavyweight request

fixed bug in limit

Allocation Limits

```
setAllocationCounter :: Int64 -> IO ()
getAllocationCounter :: IO Int64

enableAllocationLimit :: IO ()
disableAllocationLimit :: IO ()
```

- Triggers AsyncException in thread
- Easy in Haskell, very difficult in C++

Allocation Limits

Limit resources per request,
not just the runtime

Semantic Differences

Semantic Differences

"Small differences in implementation will get lost in the noise"

Semantic Differences

“Small differenc implementation
will get lo he noise”

Semantic Differences

```
fxlsh> Round(0.5)
1
```

```
haxlsh> round 0.5
0
```

Semantic Differences

```
fxlsh> StrRegexReplace(“XXX”, “a*”, “b”)
“bXbXbXb”
```

```
haxlsh> substitute “XXX” “a*” “b”
“XXX”
```

Semantic Differences

```
fxlsh> Floor(inf)
-9223372036854775808
```

```
haxlsh> floor (1.0/0.0)
17976931348623159077293051907890247336179769
9423065727343008115773267580550009…
```

Semantic Differences

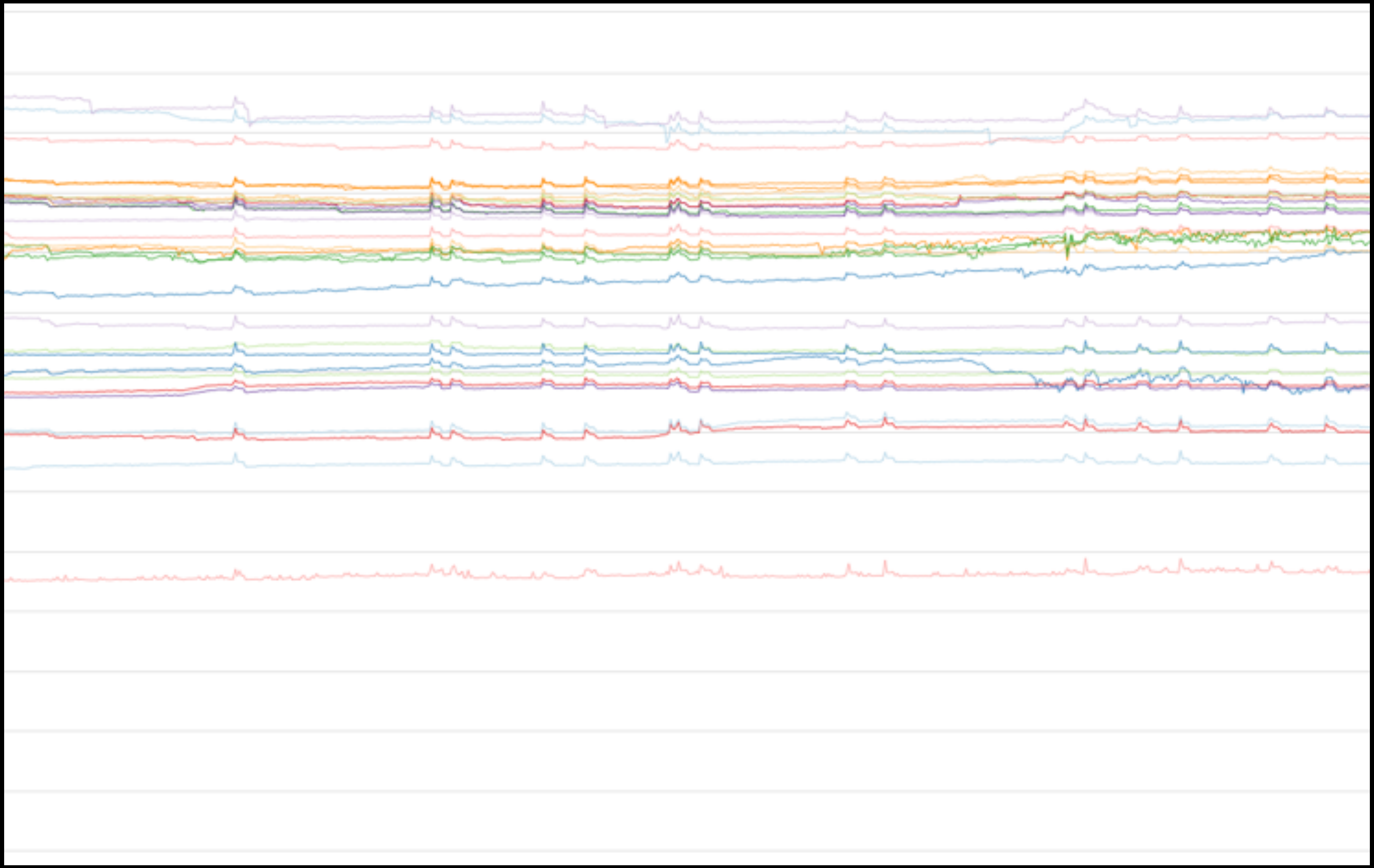
One in a million happens all the time

Perf Difference

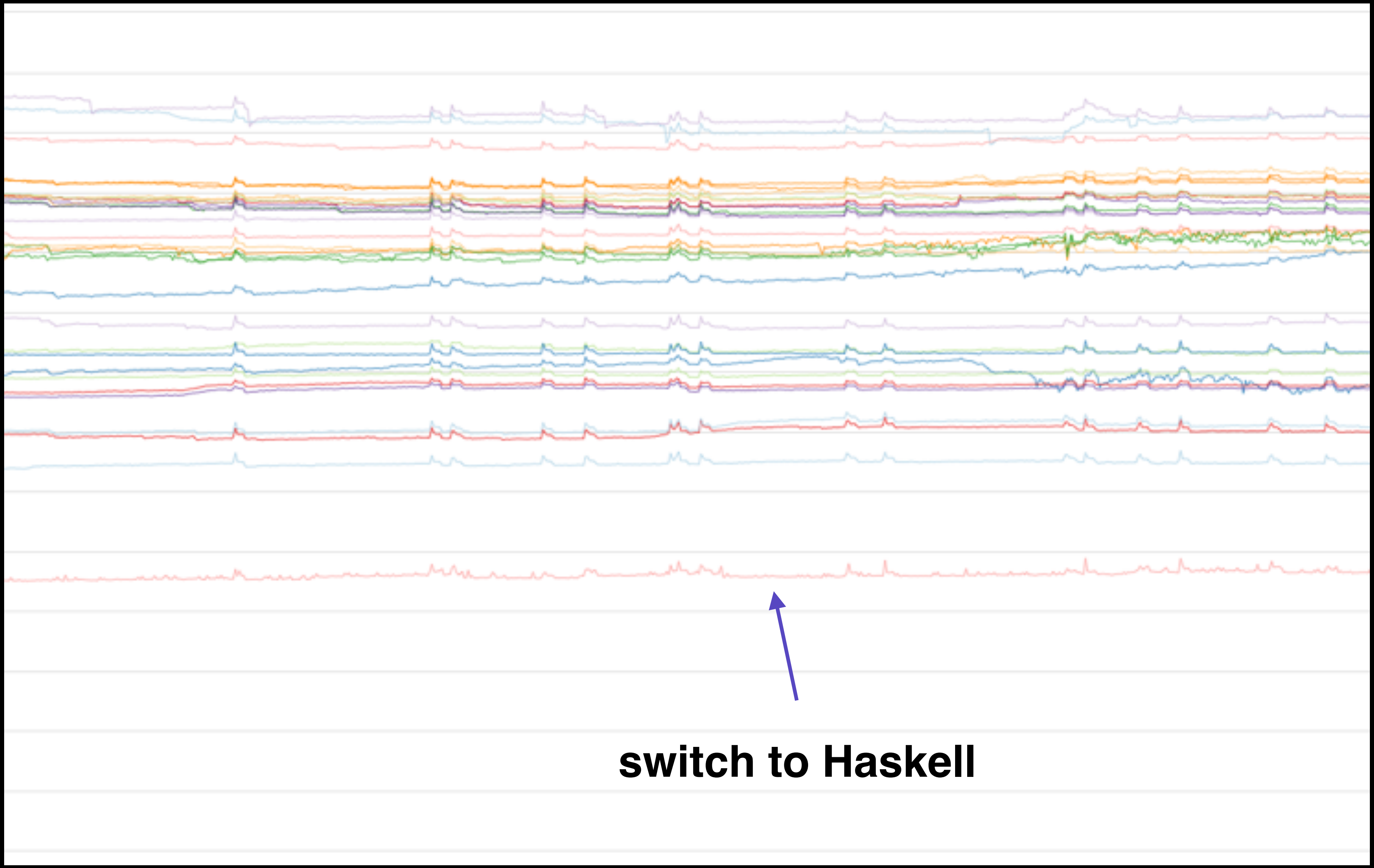
```
j <- parseJson “\\\\\\\\\\\\\\\\\\\\\…
```

http://www.serpentine.com/blog/2015/05/13/sometimes-the-old-ways-are-the-best/

Flip The Switch



Flip The Switch



* little white lie

It Works!

Spoils of Victory

GHC gave a 30% throughput increase

GC?

GC

- Fixed 1 bug that was around for years
- Never a first-order problem itself
- High GC times were allocation limit issues
- Upstreamed multiple low-hanging fruit optimizations

Upstream

- Applicative-Do (GHC 8.0)
- Allocation limits (GHC 7.10)
- GC optimizations + bug fix
- Linker functionality
- GHCi improvements (REPL)

Developers

Developers

- Multi-day hands-on workshops
- “Therapy” Facebook group
- Rosetta stone from translator

Spoils of Victory

Dozens of Haskell developers writing production code daily

Brave New Haskell World

Haskell is ready for industry

Haxl Devs (Past + Present)

- Simon Marlow
- Louis Brandy
- Aaron Roth
- Jon Purdy
- Bartosz Nitka
- Kubo Kovac
- Zejun Wu
- Jake Lengyel
- Katie Miller
- Noam Zilberstein
- Andrew Farmer
- Mehmet Yatbaz

Questions?

Jon Coens
Haskell Shepherd at Facebook

@JonCoens
http://github.com/facebook/Haxl