

Erlang, The Road Movie

Lessons learned in Erlang Land



Kresten Krab Thorup
CTO, Trifork

[Java, ...] What are we missing?

- Virtual Machine
 - Modularity, Lifecycle & Isolation
 - Lower memory footprint
 - Predictable GC pauses
- Platform
 - Distributed system as a system
 - Provisioning and Metering
 - Cloud Operating System APIs
 - Persistence (including key/value)
 - Map/Reduce-style processing
- Application Definition
 - Packaging, Resource Declaration
 - Security



Erlang, The Road Movie

Lessons learned in Erlang Land

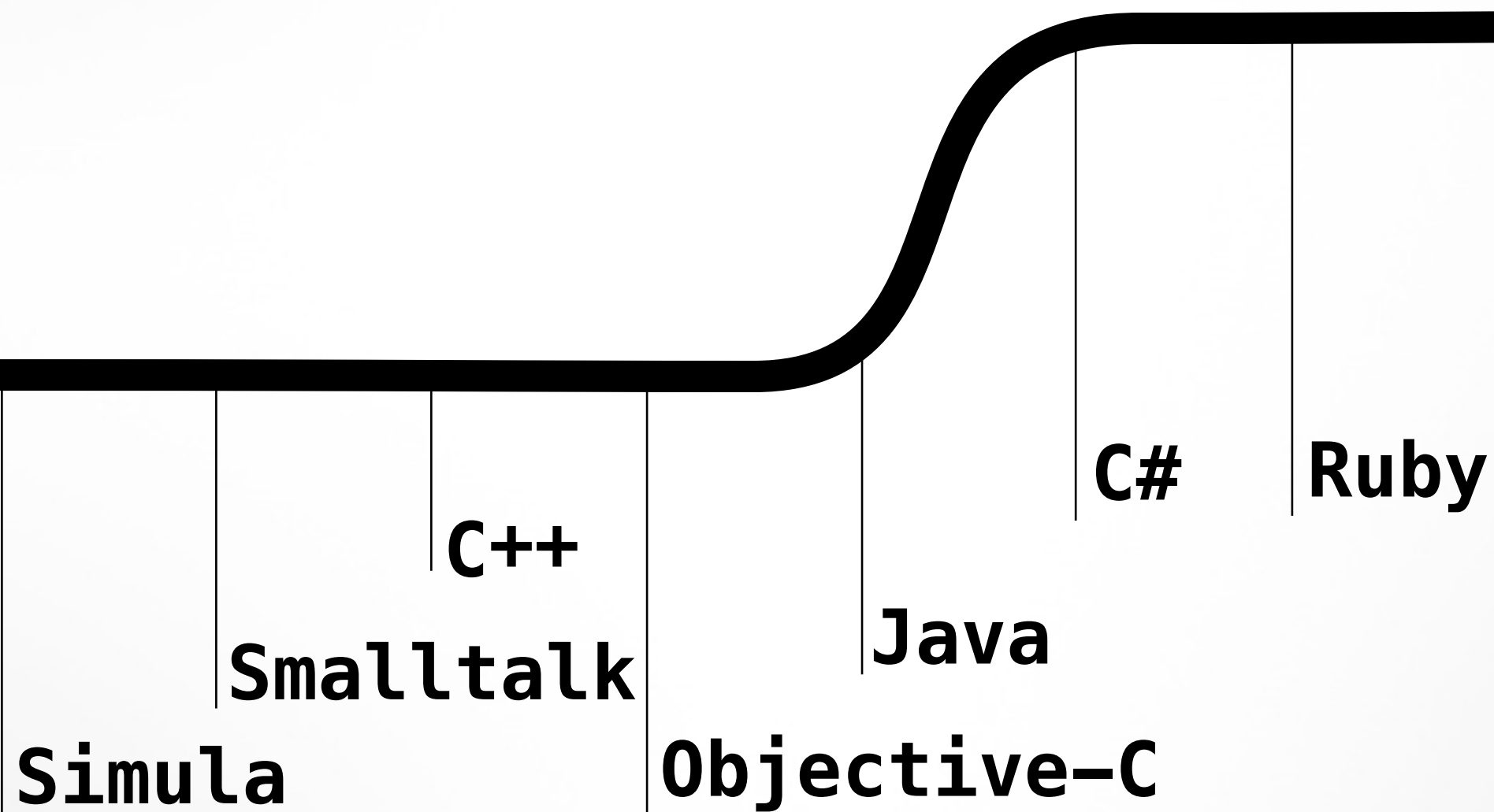
Kresten Krab Thorup
CTO, Trifork

About the Speaker

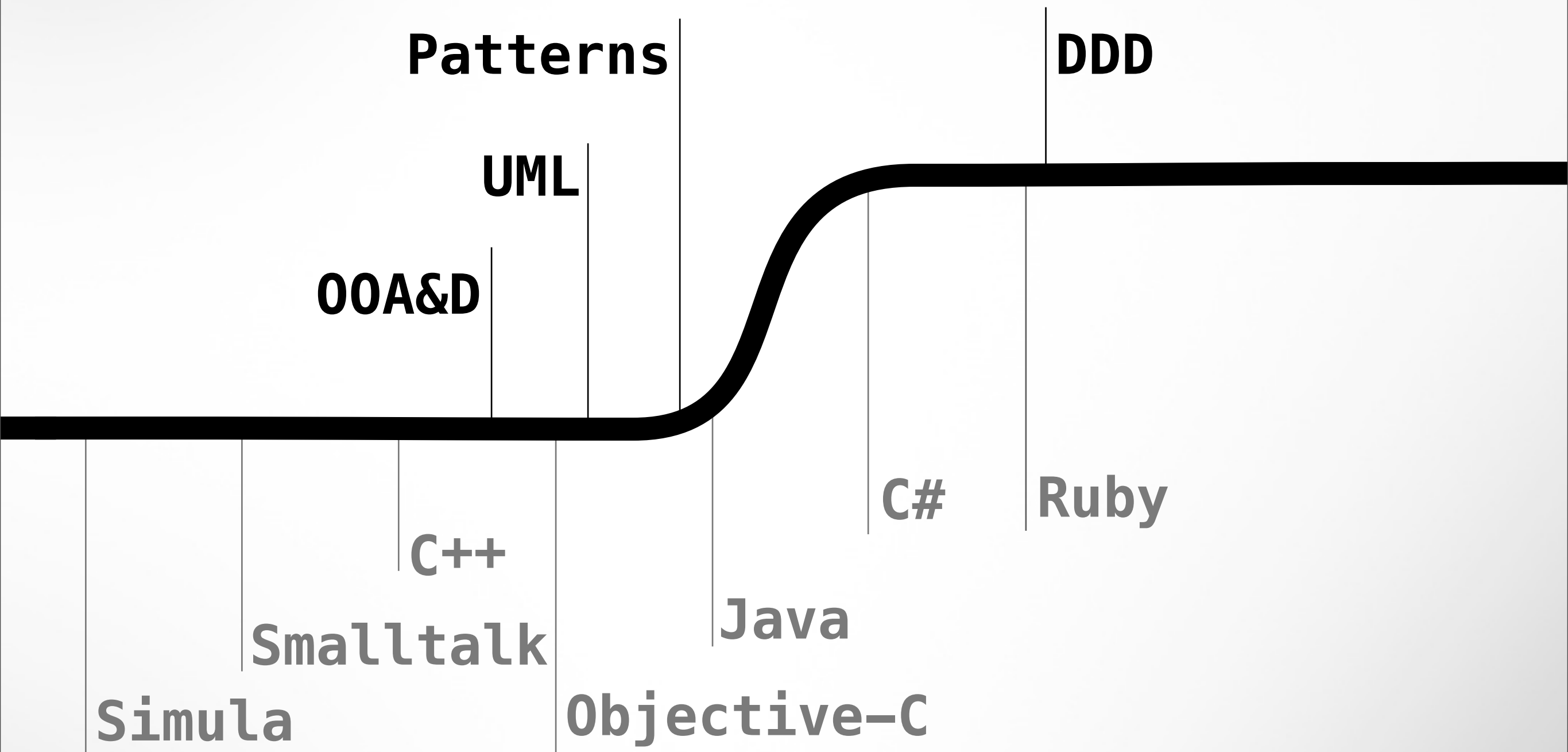
- **Language Geek** Emacs/TeX Hacker, Objective C, NeXT, GNU Compiled Java, Java Generics, Ph.D.
- **Developer** J2EE AppServer, CORBA/RMI, XA-TM, Java Firefighter
- **Trifork CTO** Conference “Editor”, Technology Adoption

'90s Object Revolution

Object Technology



Object Thinking Tools



What was pushing us?

**Increased
Complexity**

A thick black line that starts horizontally on the left, then curves upwards and to the right, leveling off horizontally again.

What was pushing us?

**Increased
Complexity**

Internet



What was pushing us?

Internet

**Increased
Complexity**

Better Technology

What was pushing us?

Internet

**Increased
Complexity**

Component Reuse

Better Technology

What was pushing us?

Internet

**Increased
Complexity**

Domain Modeling

Component Reuse

Better Technology

More Complexity

More Complexity

You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。

More Complexity

More Complexity

More Complexity

Fault Tolerance, Availability, QoS

More Complexity

**Integration, Coordination
Fault Tolerance, Availability, QoS**

More Complexity

**Cloud, Multi-Core
Integration, Coordination
Fault Tolerance, Availability, QoS**

More Complexity

We're struggling to
handle these with an
object mindset!

**Cloud, Multi-Core
Integration, Coordination
Fault Tolerance, Availability, QoS**

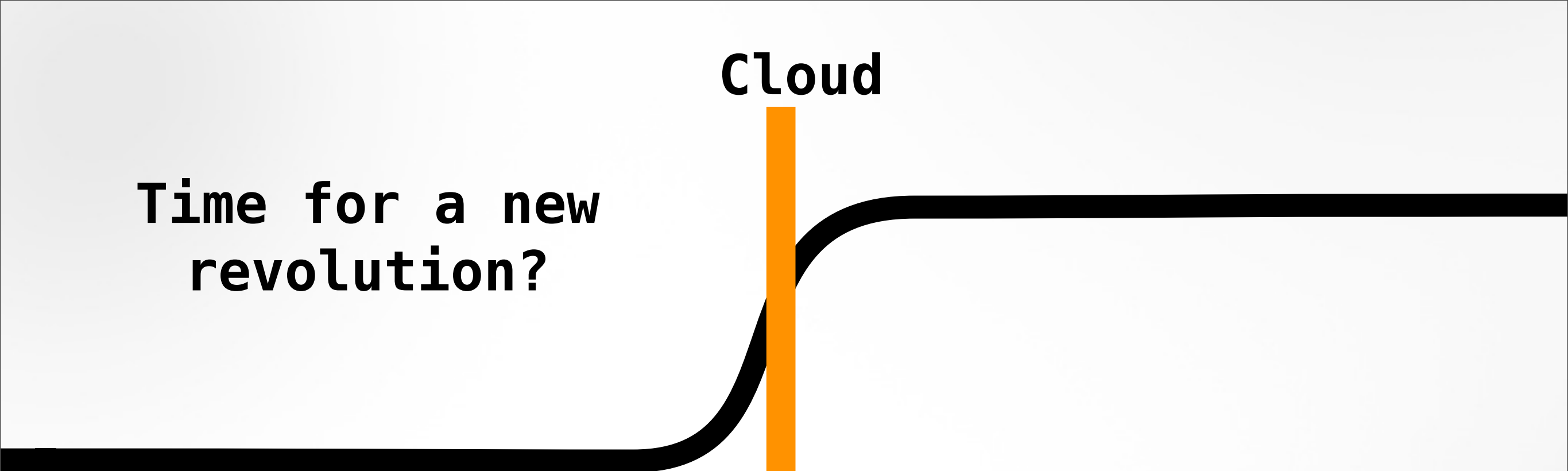
Time for a new revolution?



Cloud, Multi-Core
Integration, Coordination
Fault Tolerance, Availability, QoS

Cloud

**Time for a new
revolution?**



**Cloud, Multi-Core
Integration, Coordination
Fault Tolerance, Availability, QoS**

What's a Revolution?



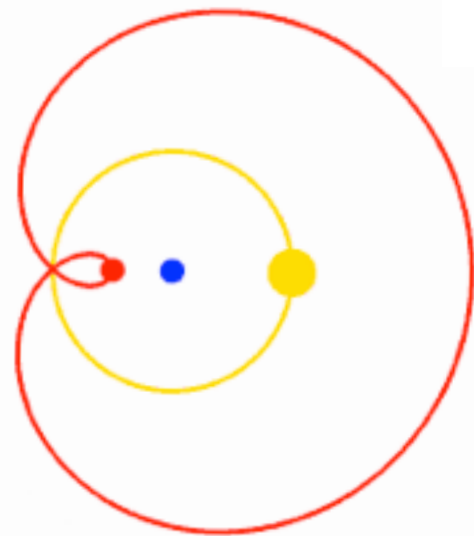
Thomas Kuhn
The Structure of Scientific Revolutions

paradigm

paradigm

paradigm

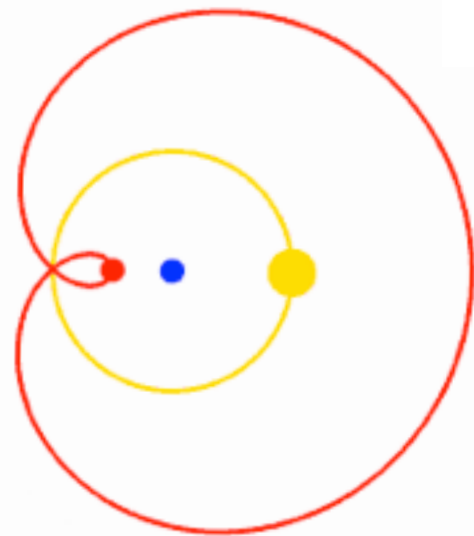
paradigm



paradigm

paradigm


*normal
science*



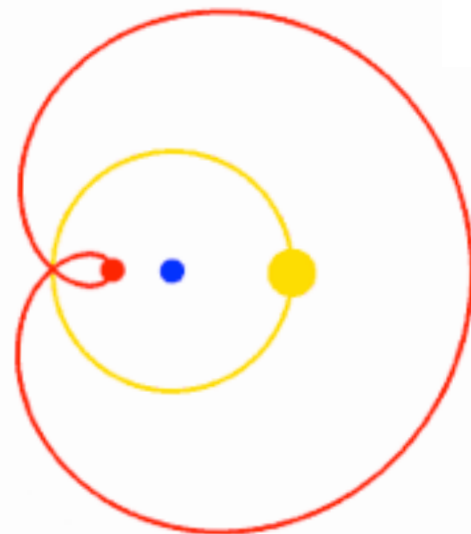
paradigm

*observe
anomalies*



paradigm

*normal
science*



paradigm

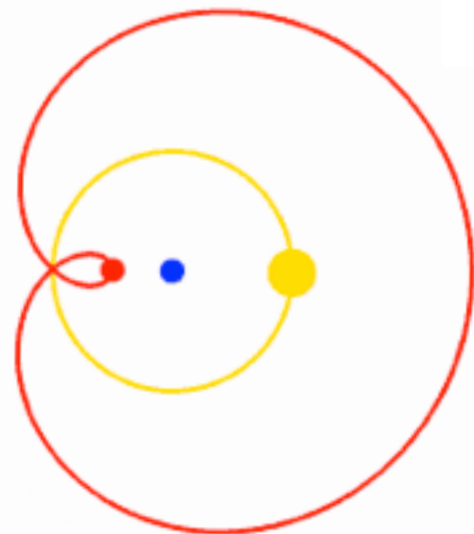
*observe
anomalies*



paradigm

CRISIS

*normal
science*



paradigm

*observe
anomalies*

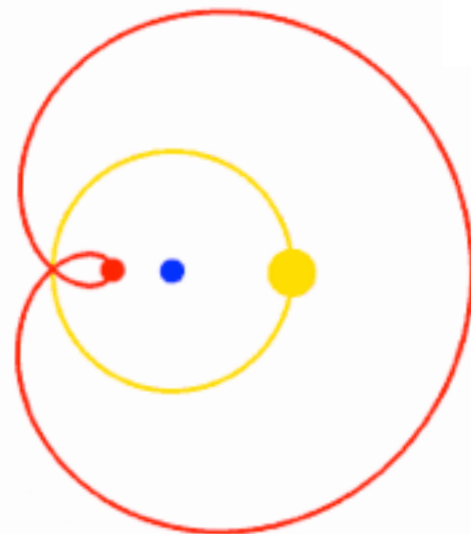
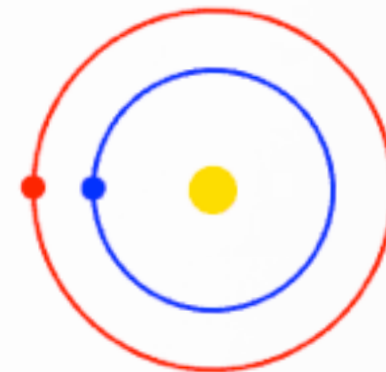


paradigm

CRISIS

*normal
science*

*revolutionary
science*



paradigm

*observe
anomalies*



paradigm

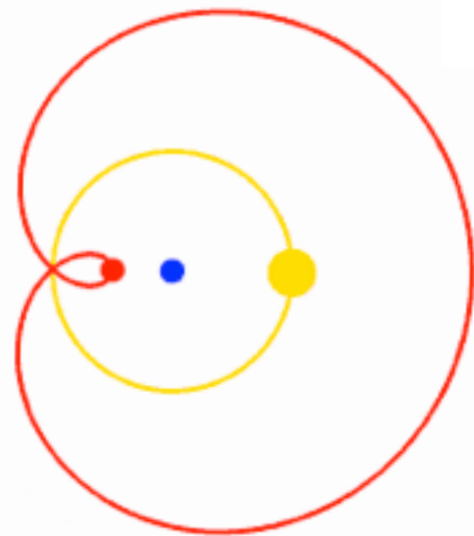
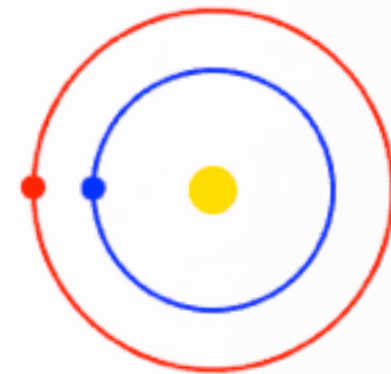
CRISIS

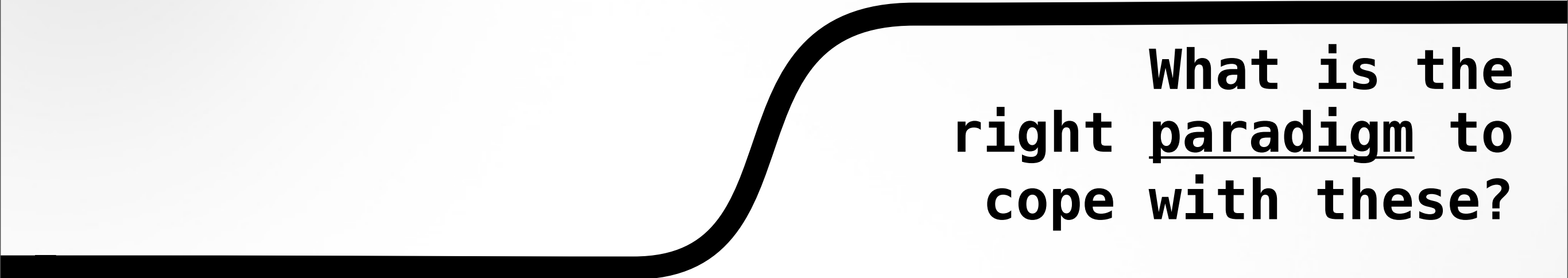
*normal
science*



*normal
science*

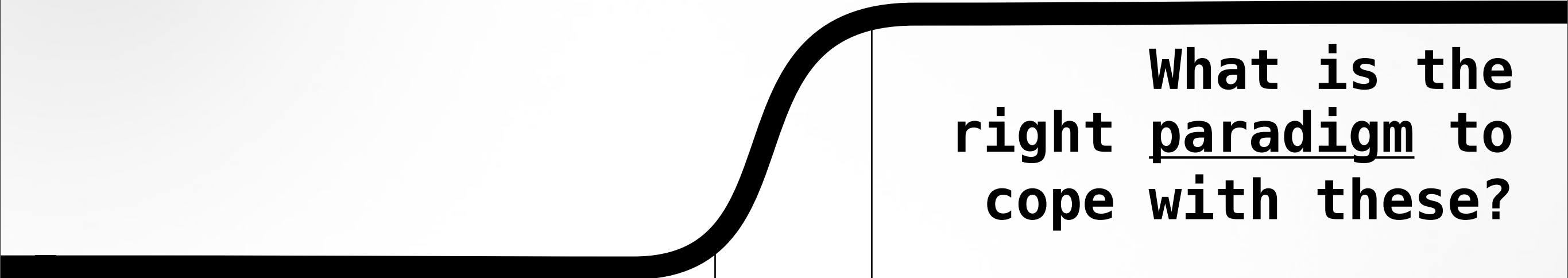
*revolutionary
science*





**What is the
right paradigm to
cope with these?**

**Cloud, Multi-Core
Integration, Coordination
Fault Tolerance, Availability, QoS**



**What is the
right paradigm to
cope with these?**

**Cloud, Multi-Core
Integration, Coordination
Fault Tolerance, Availability, QoS**

**Clojure, Node.js
Erlang, Actor Models
Haskell, Scala**

Ralph Johnson's blog "Erlang, the Next Java"

... Erlang is going to be a very important language ... Its main advantage is that it is perfectly suited for the multi-core, web services future. In fact, it is the ONLY mature, rock-solid language that is suitable for writing highly scalable systems to run on multicore machines.

is the
digm to
these?

Fault

js

cs

Haskell, Scala



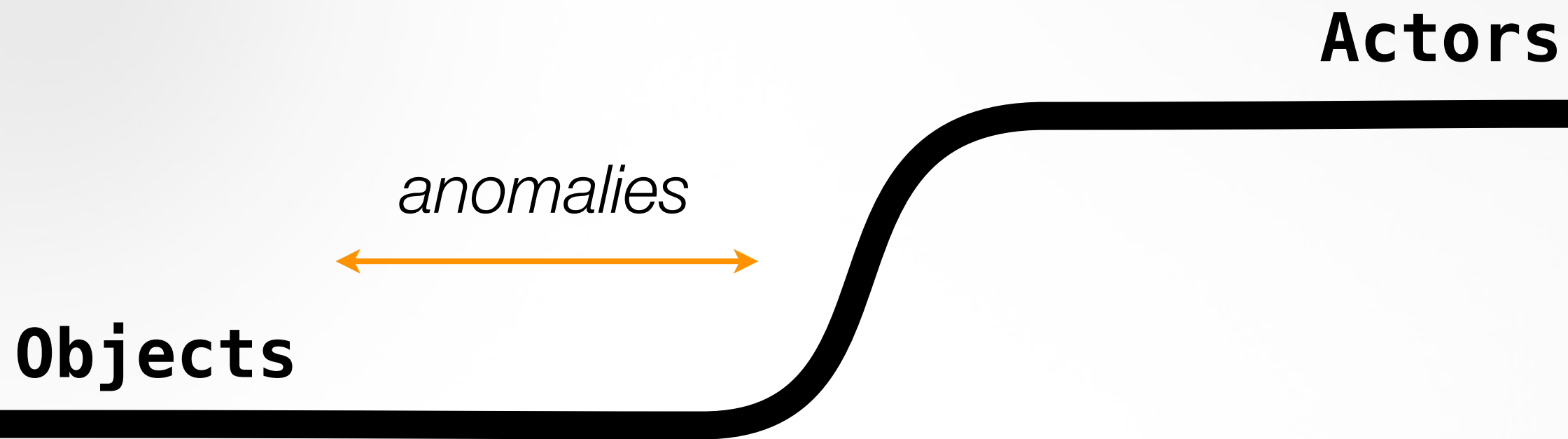
Objects

Actors

Objects

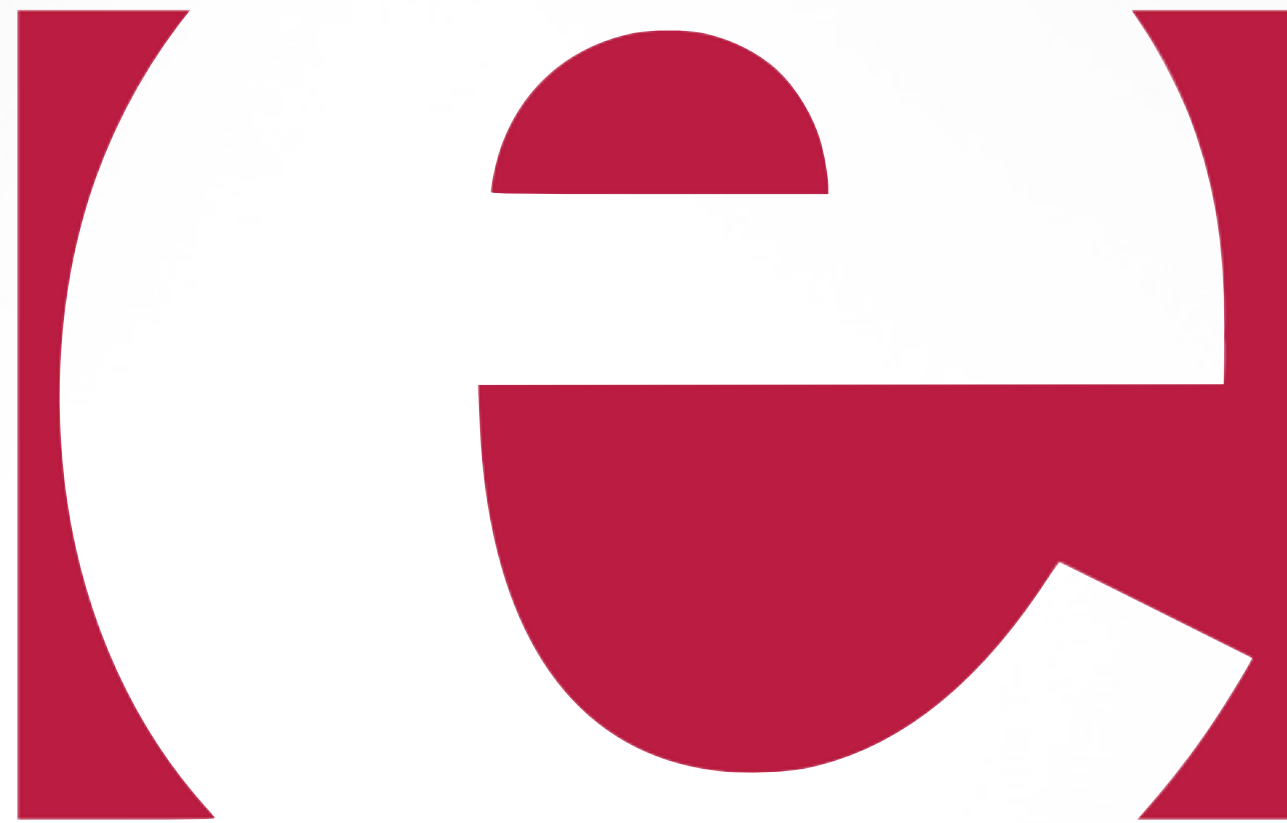


Actors



Quest in this talk:

If Actor-Programming
is the new Paradigm,
what are the anomalies
we should see now?



ERLANG

Making reliable
distributed control systems
in the presence of errors



Making reliable distributed control systems in the presence of errors



ERLANG

Making reliable distributed control systems in the presence of errors

- The “secret weapon” for Ericsson’s market leading, real-time telephony systems.



ERLANG

Making reliable distributed control systems in the presence of errors

- The “secret weapon” for Ericsson’s market leading, real-time telephony systems.
- 20+ years of experience to learn from.



ERLANG

How to Learn Erlang

How to Learn Erlang

Don't dissect a frog,
Build One!

Nicolas Negroponte

Your Erlang Program

Platform Framework



BEAM Emulator

BIFs

Your favorite OS

Your Erlang Program

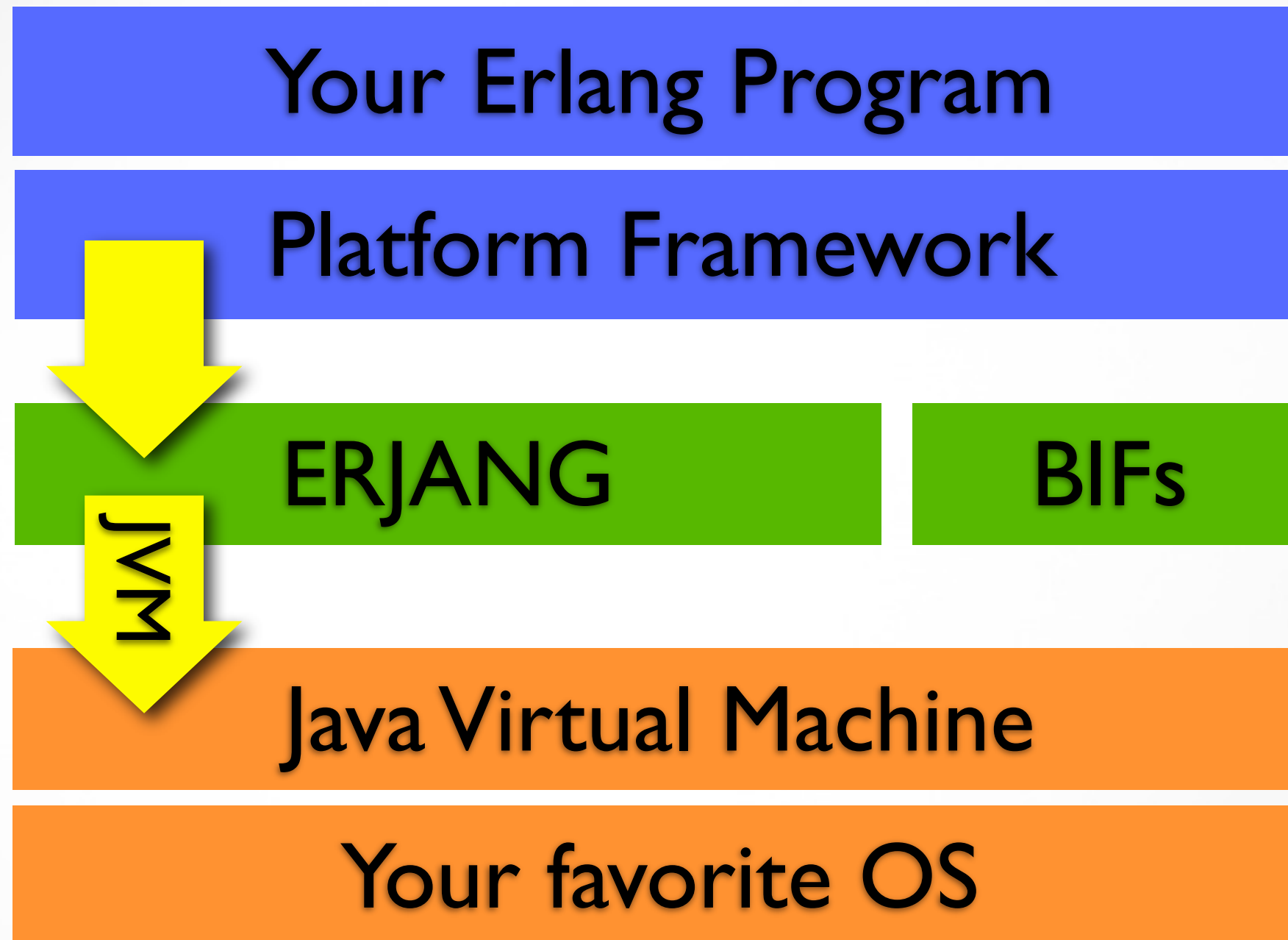
Platform Framework



BEAM Emulator

BIFs

Your favorite OS

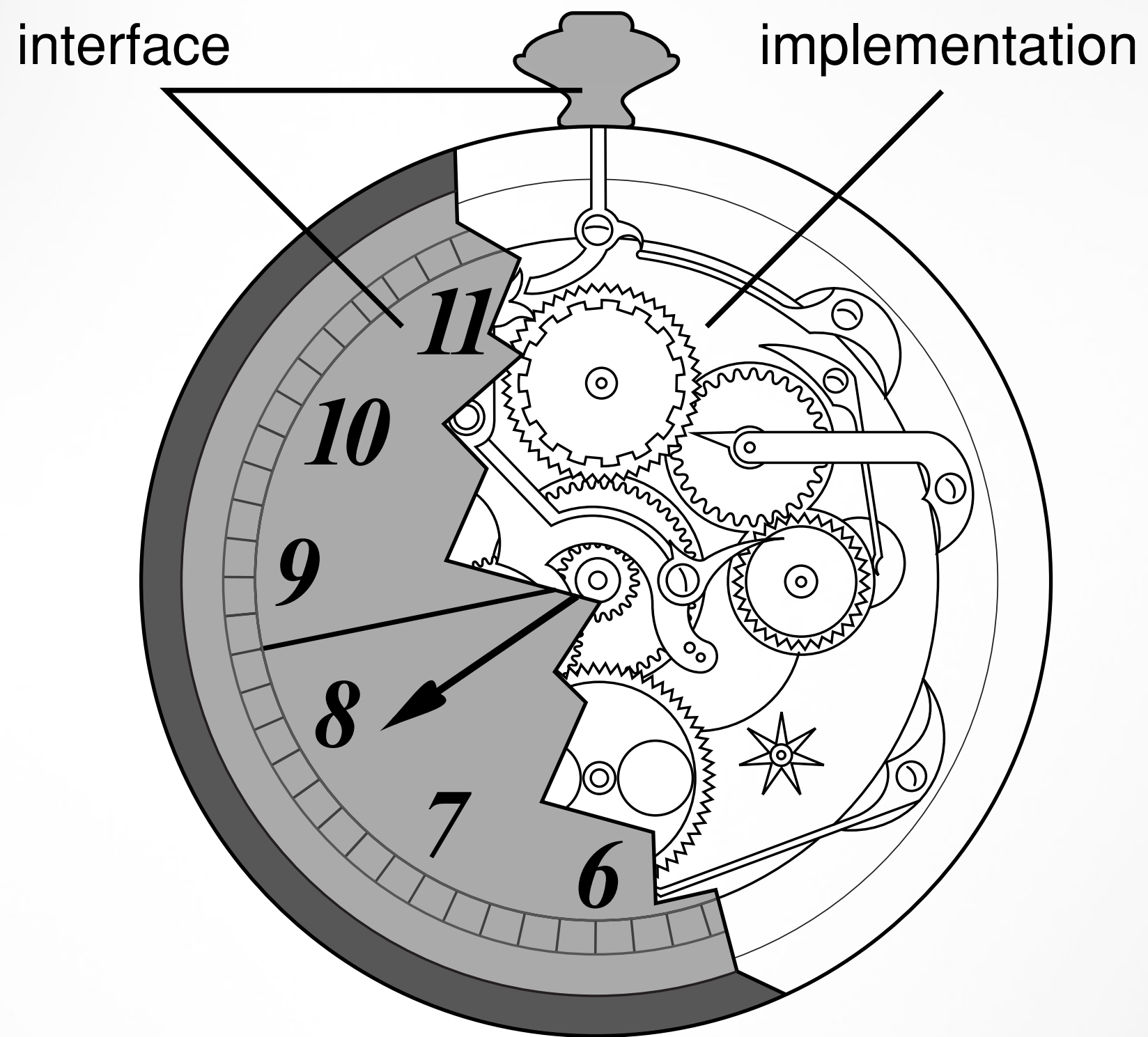




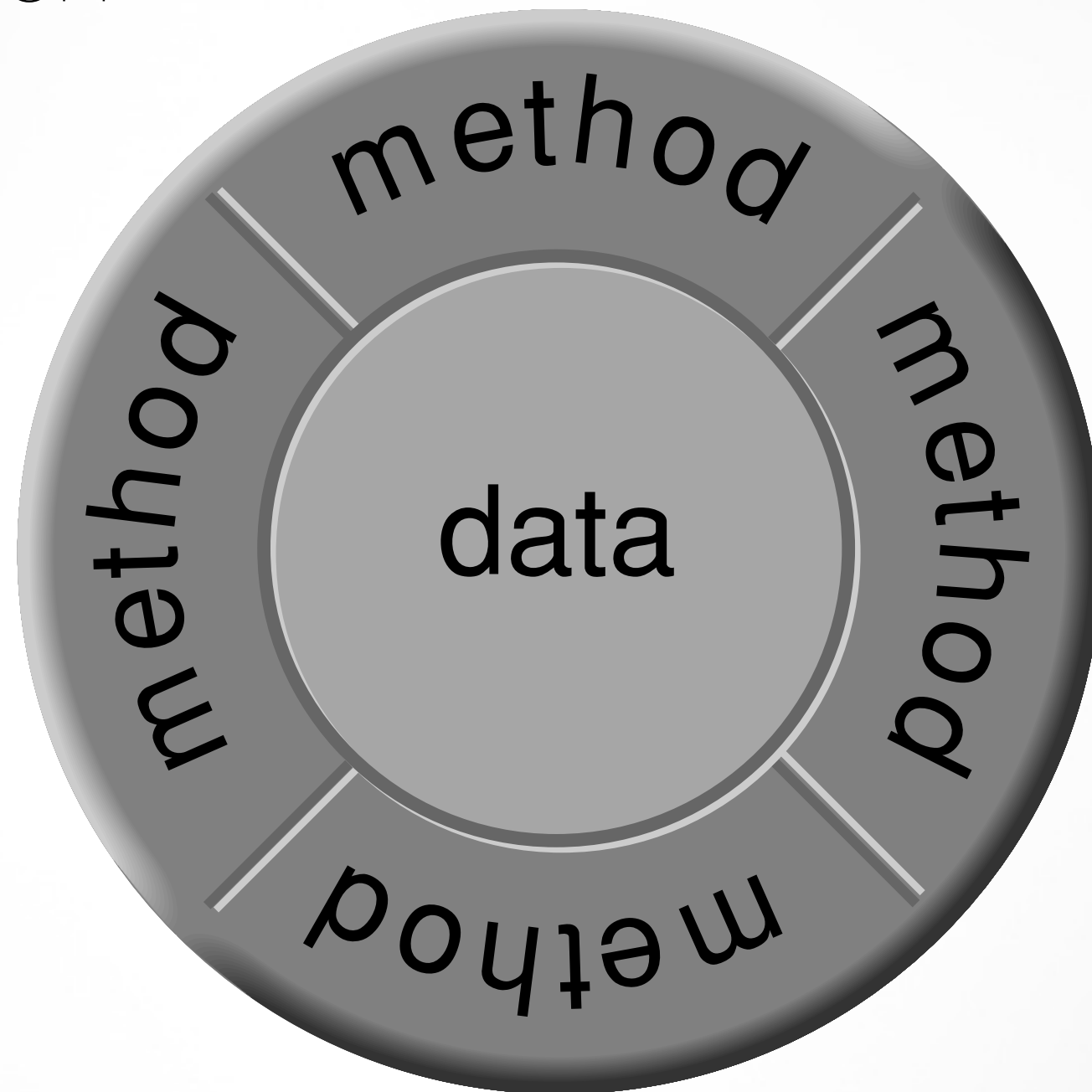
Randomized Tests

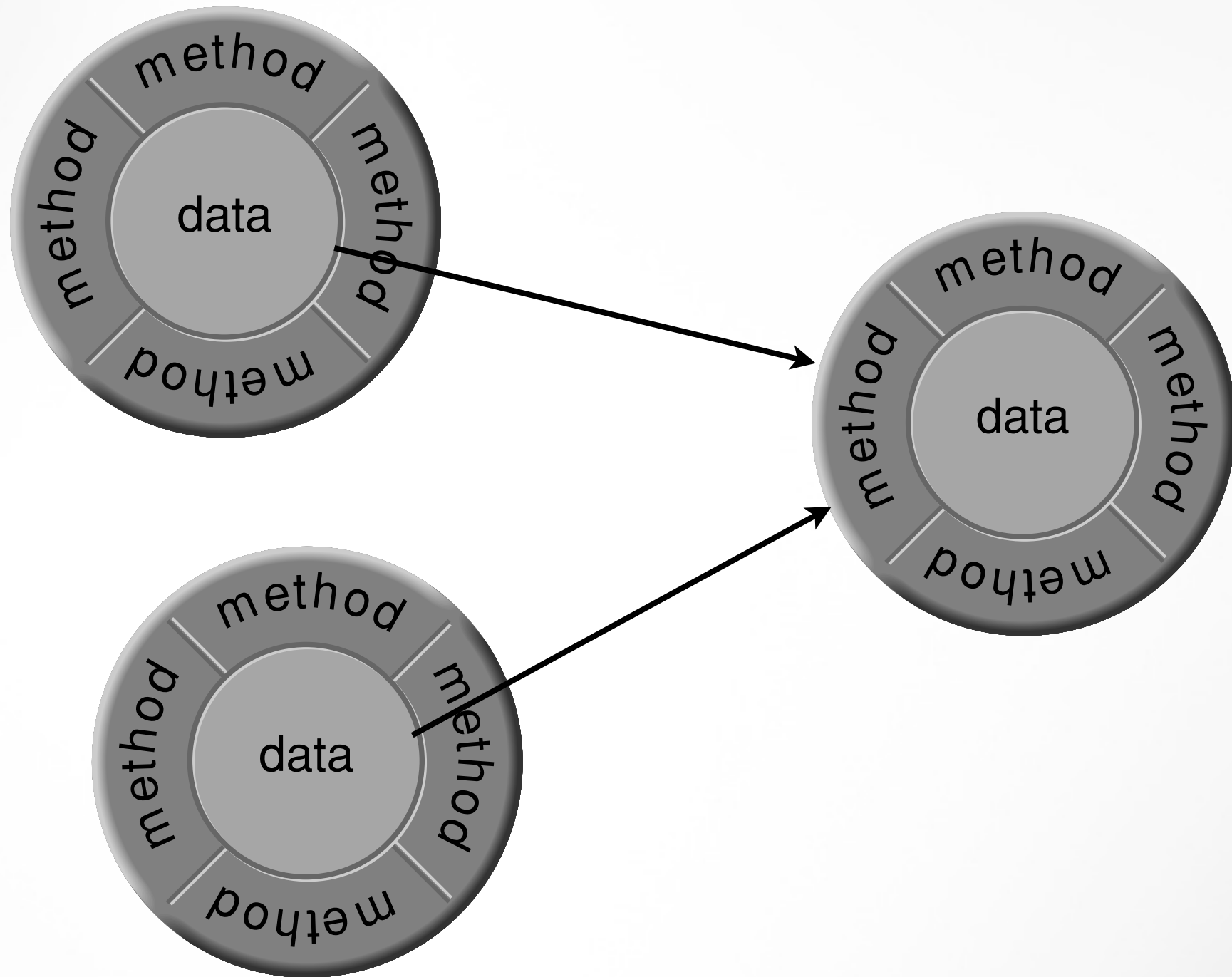


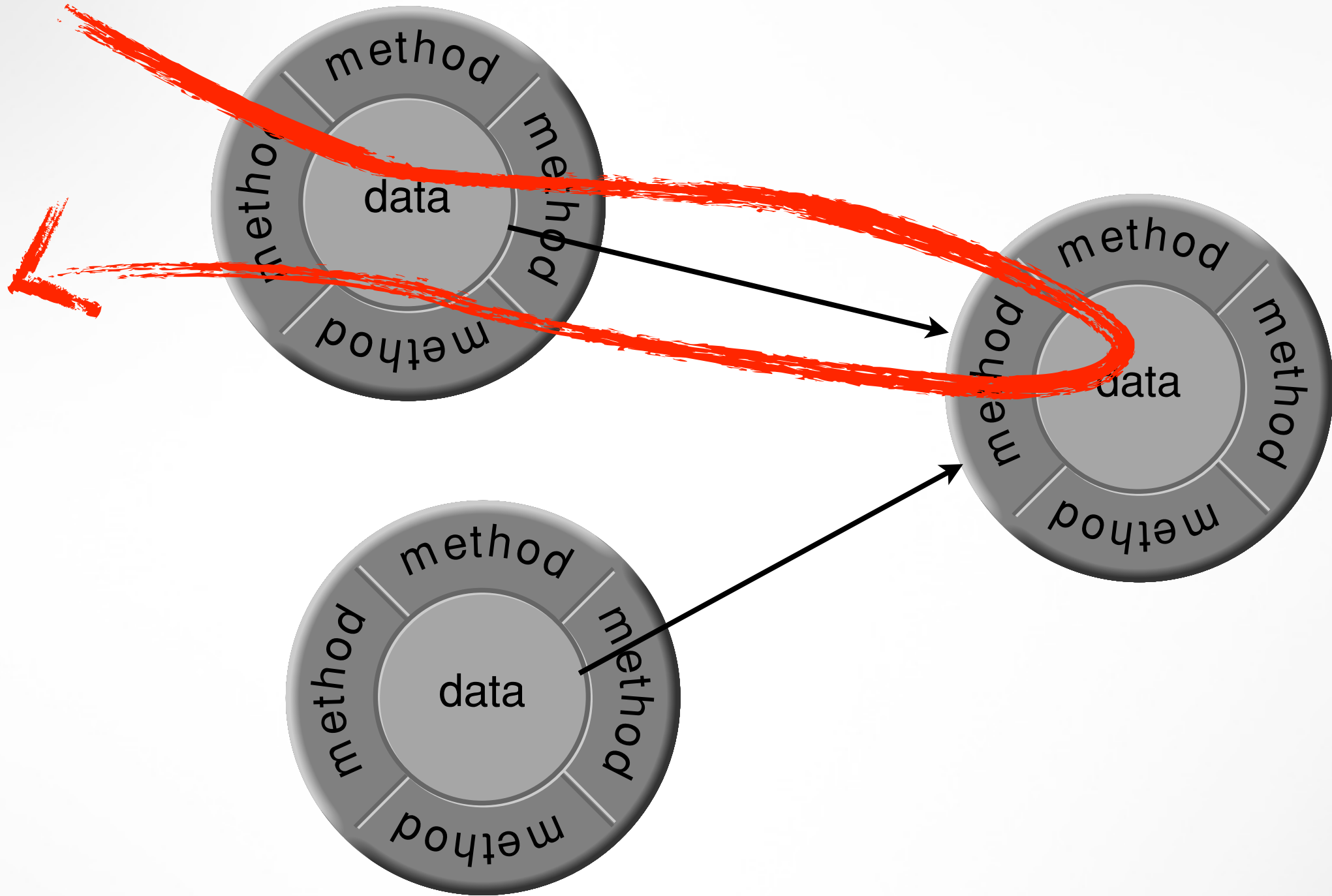
Anomalies in the object-oriented world view

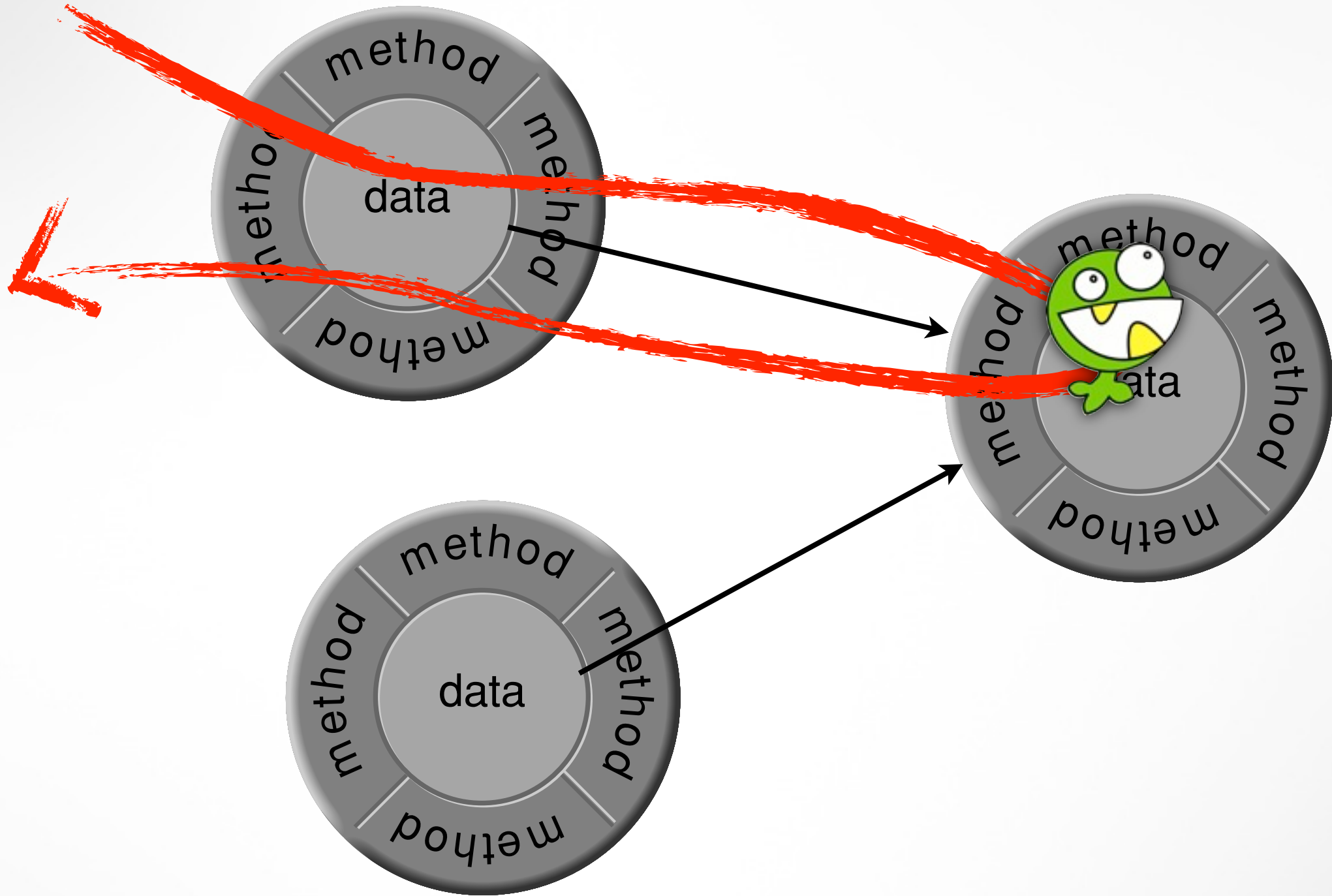


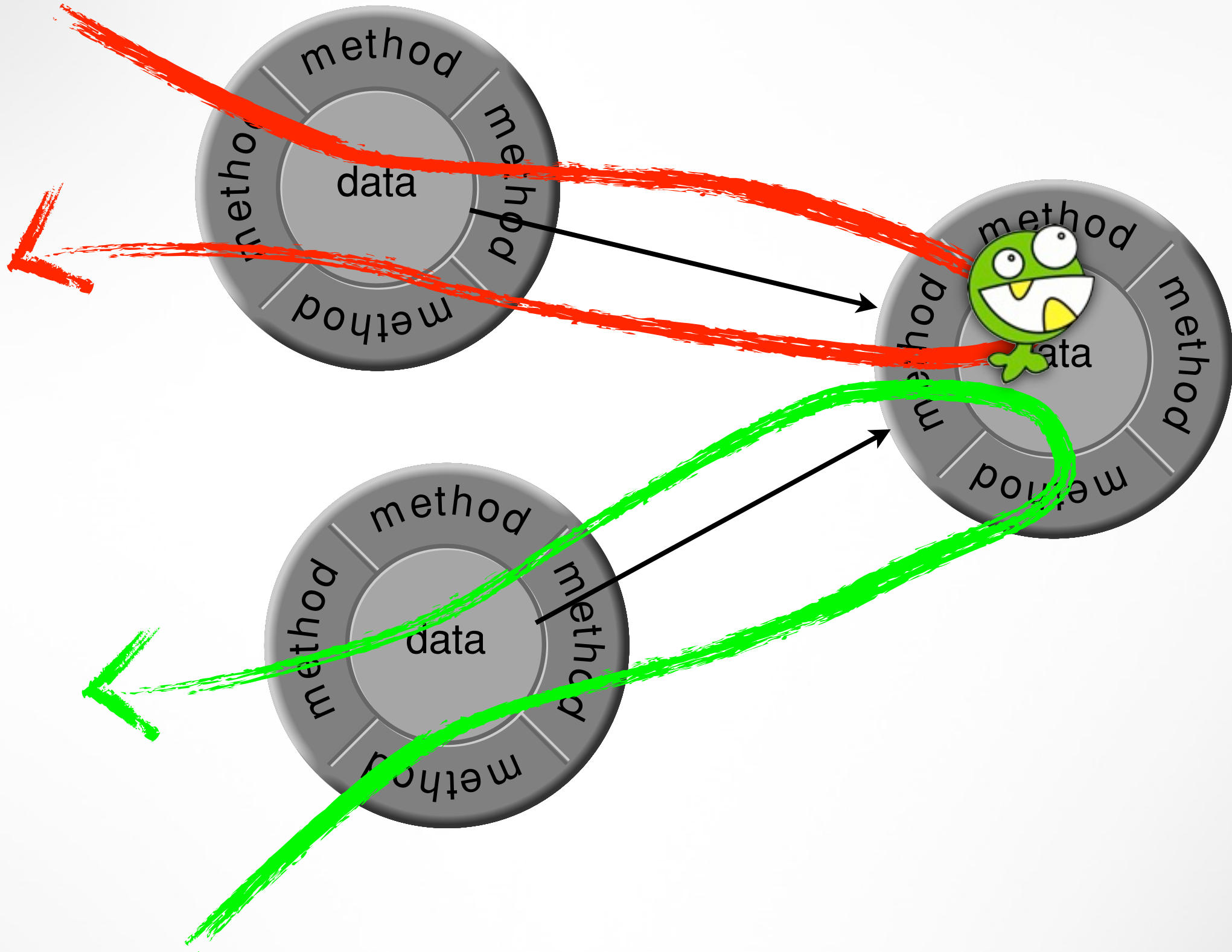
Encapsulation

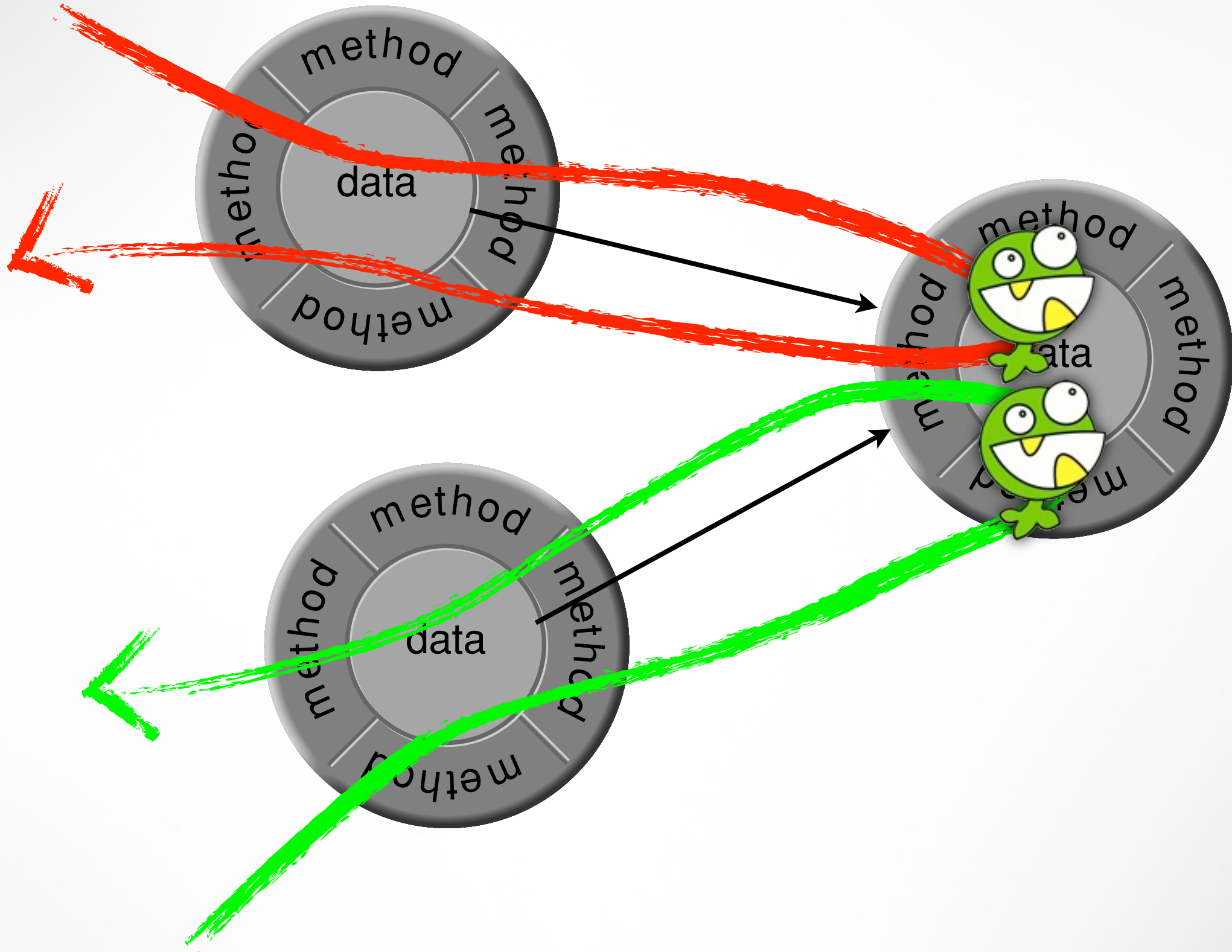


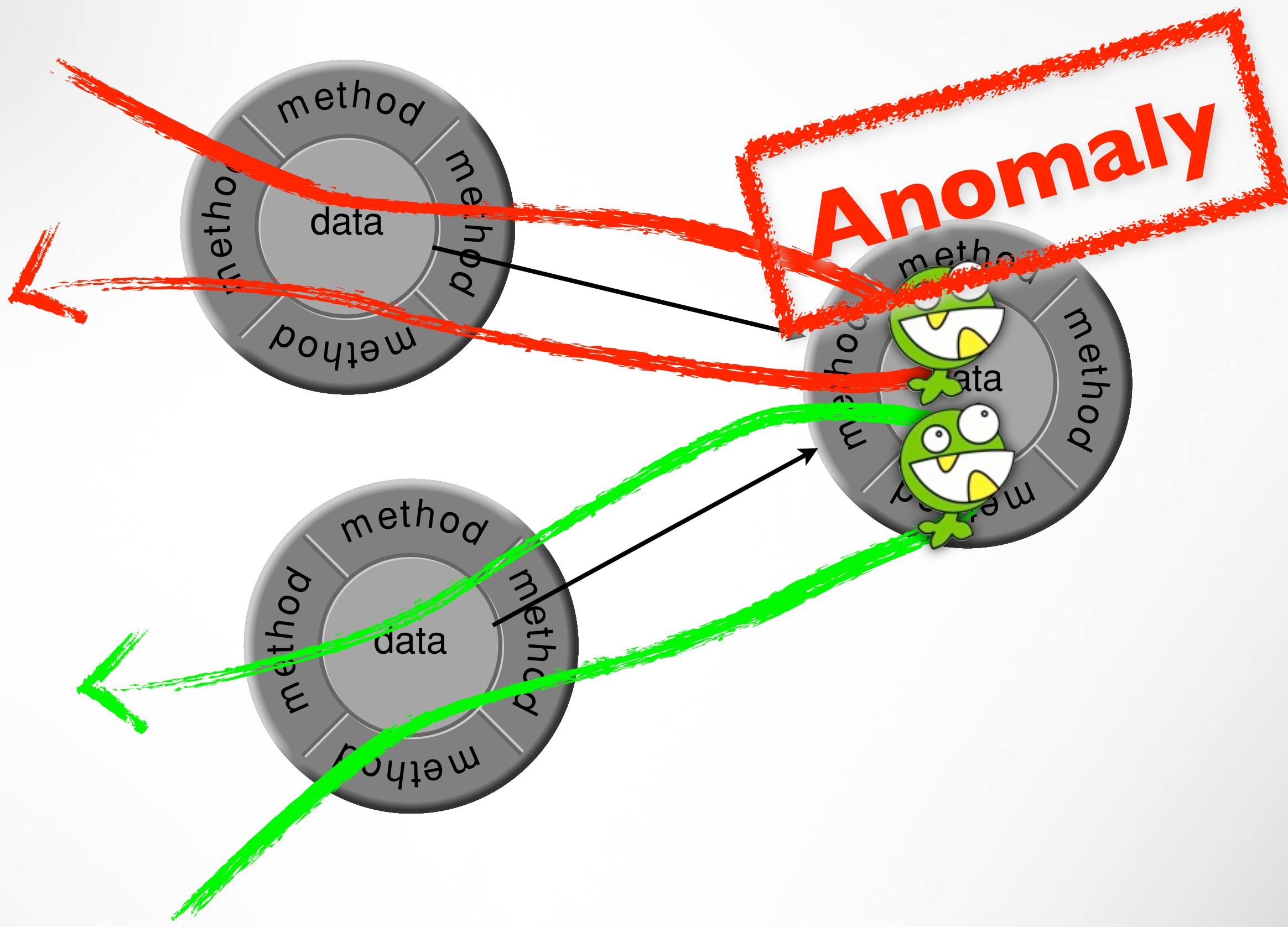


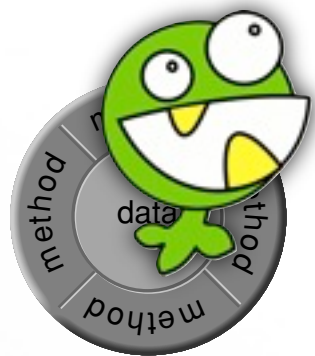
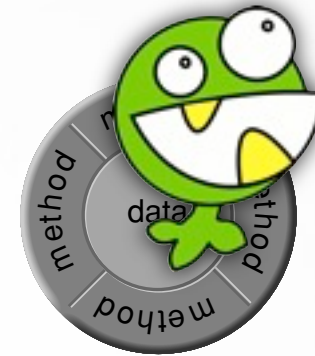
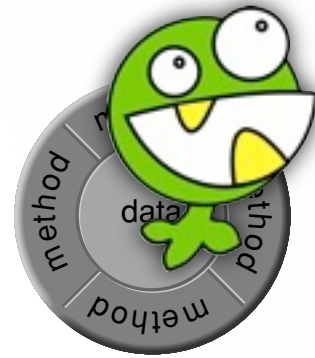


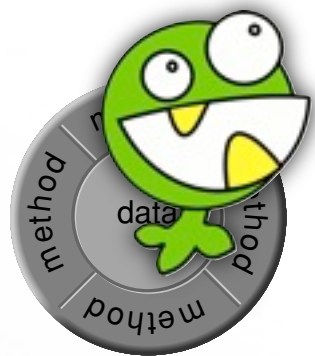
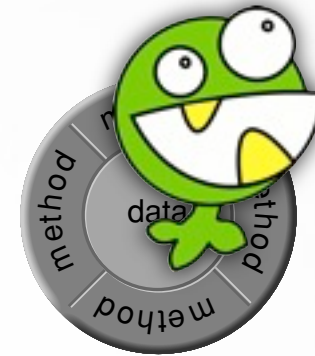
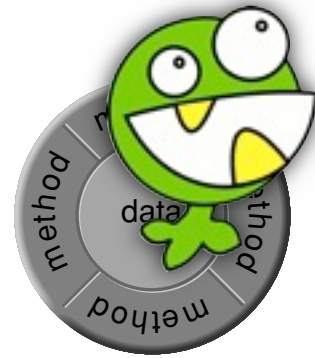


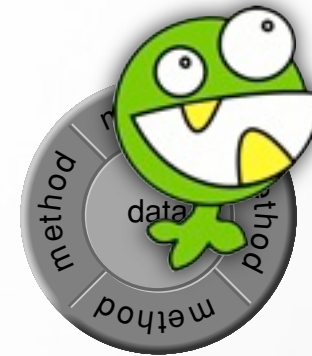
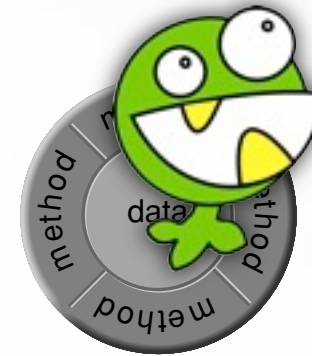






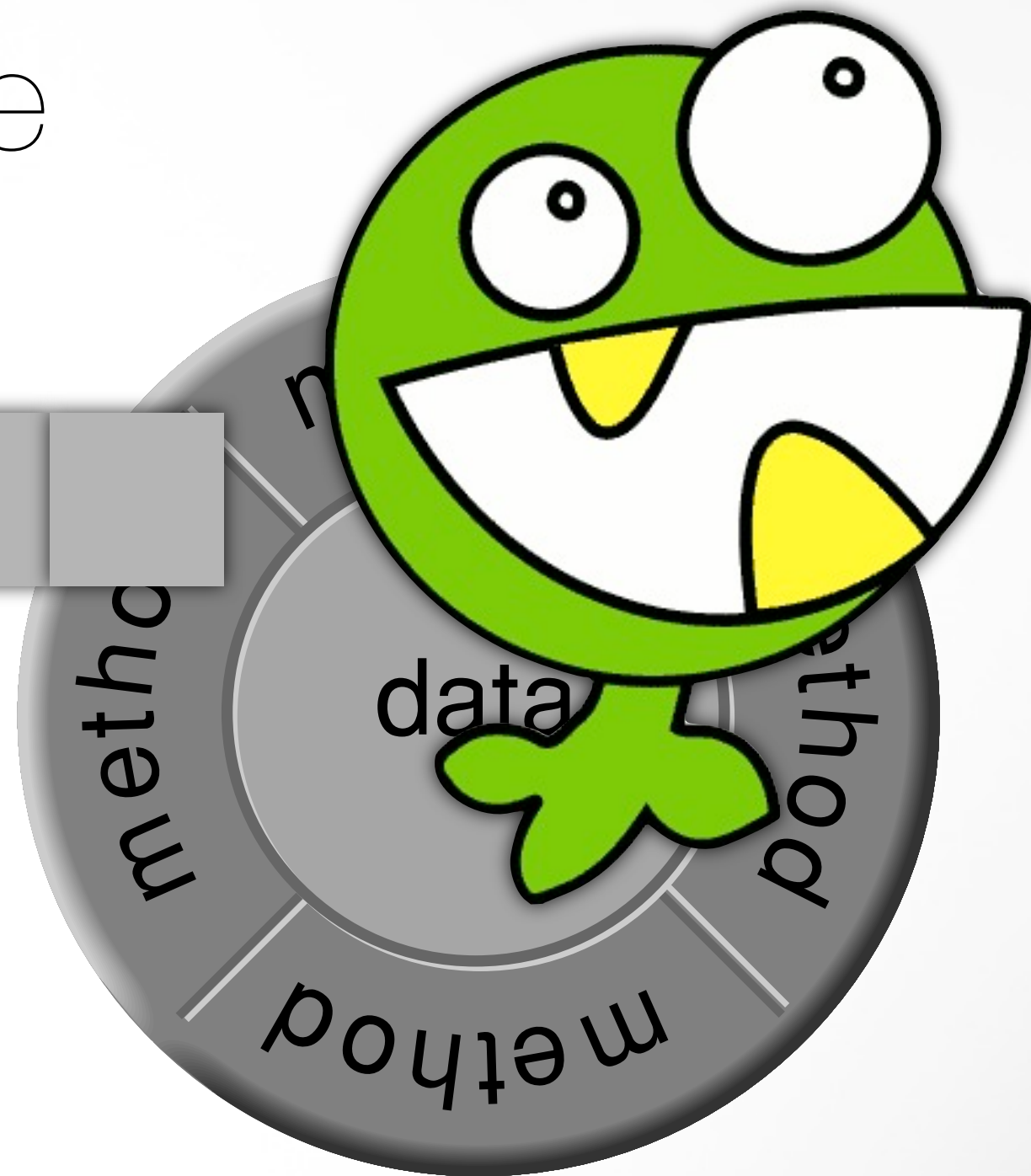






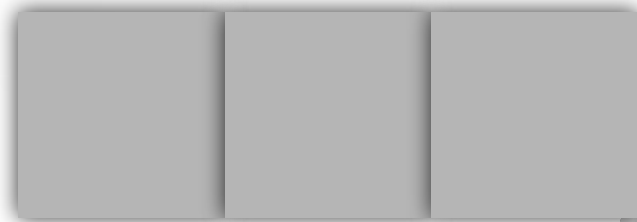
Active Object + State Machine

mailbox



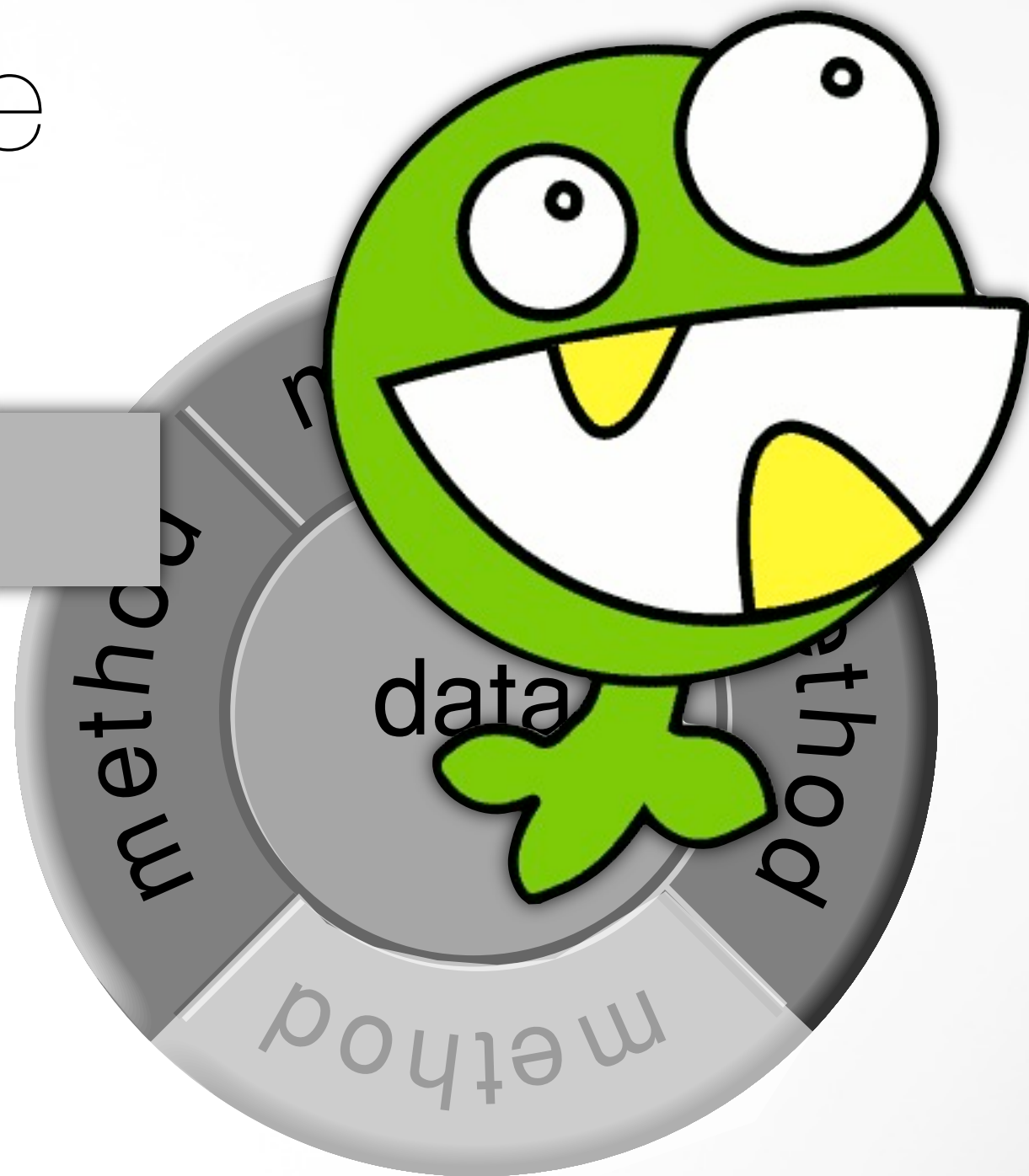
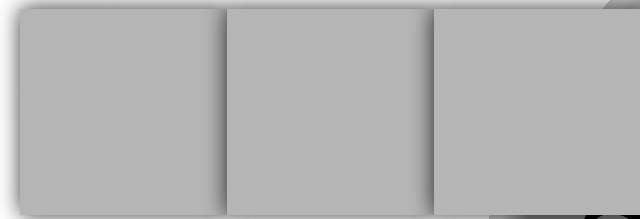
Active Object + State Machine

mailbox



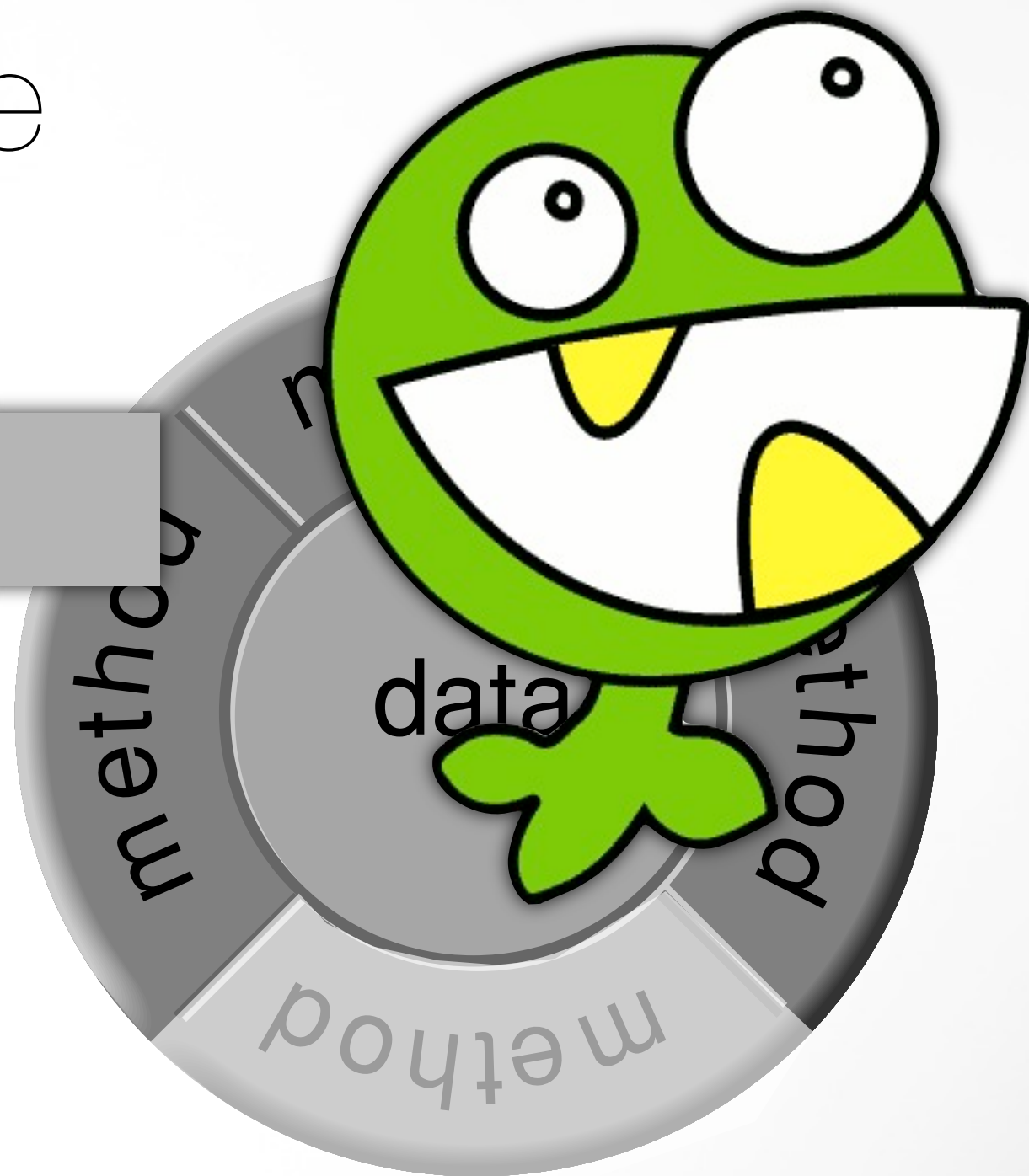
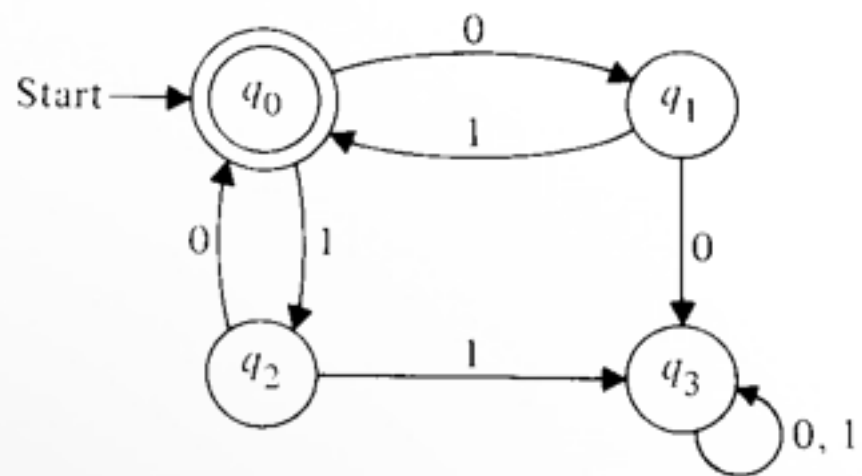
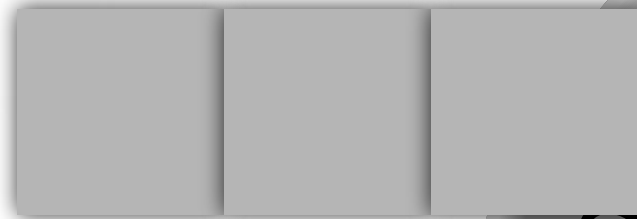
Active Object + State Machine

mailbox



Active Object + State Machine

mailbox



Objects

Interface
Fixed API

Actors

Protocol
API changes
with internal state

Objects

Anomaly

Interface

Fixed API

Actors

Protocol

API changes
with internal state

But isn't this expensive?

But isn't this expensive?

... heard in the '90s



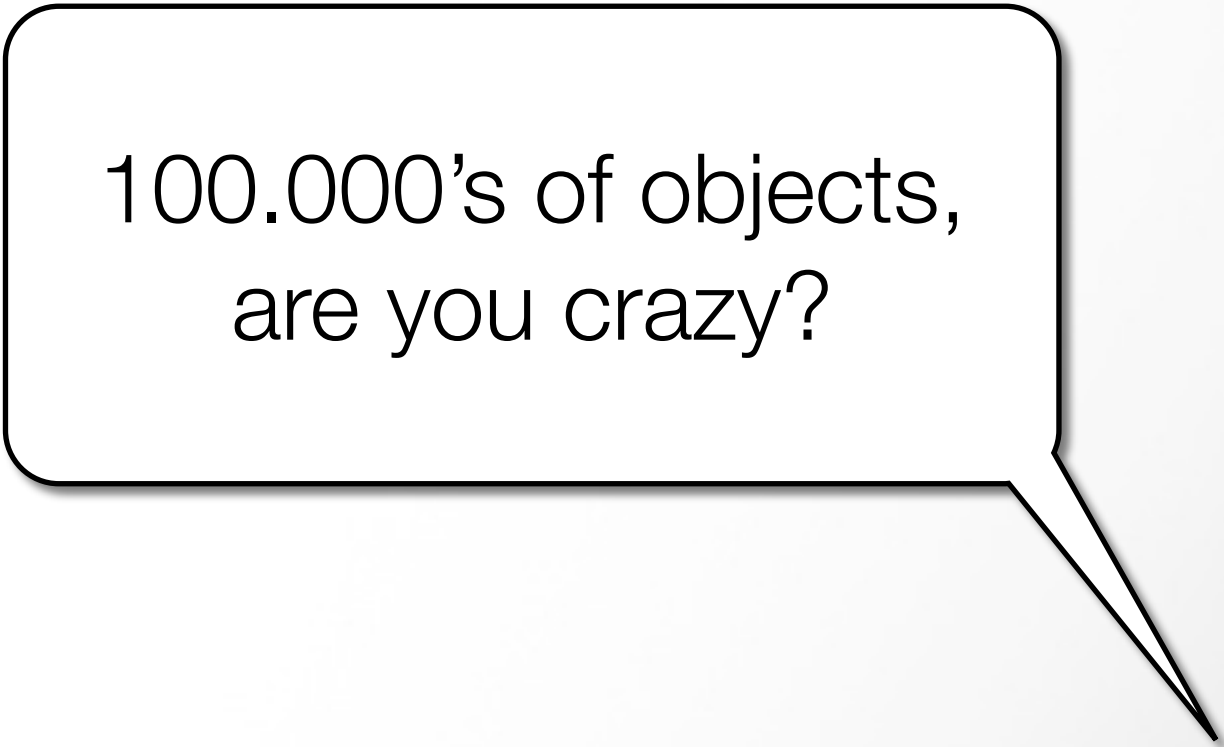
Use Objects!

But isn't this expensive?

... heard in the '90s



Use Objects!



100.000's of objects,
are you crazy?

But isn't this expensive?

... heard yesterday



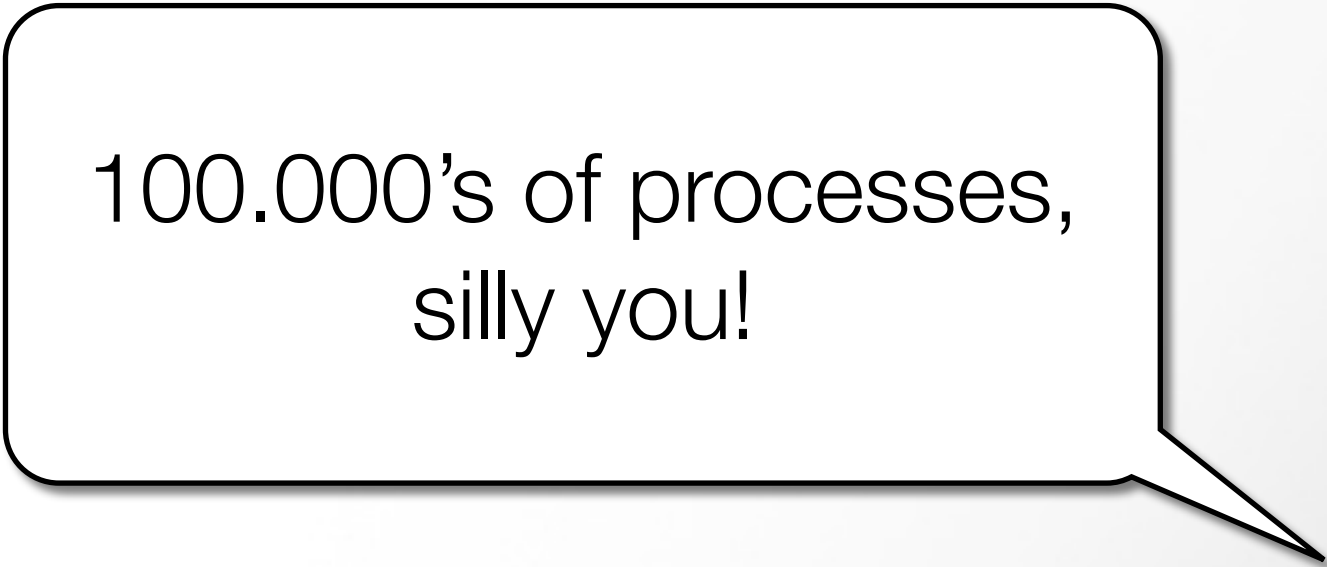
Use Actors!

But isn't this expensive?

... heard yesterday



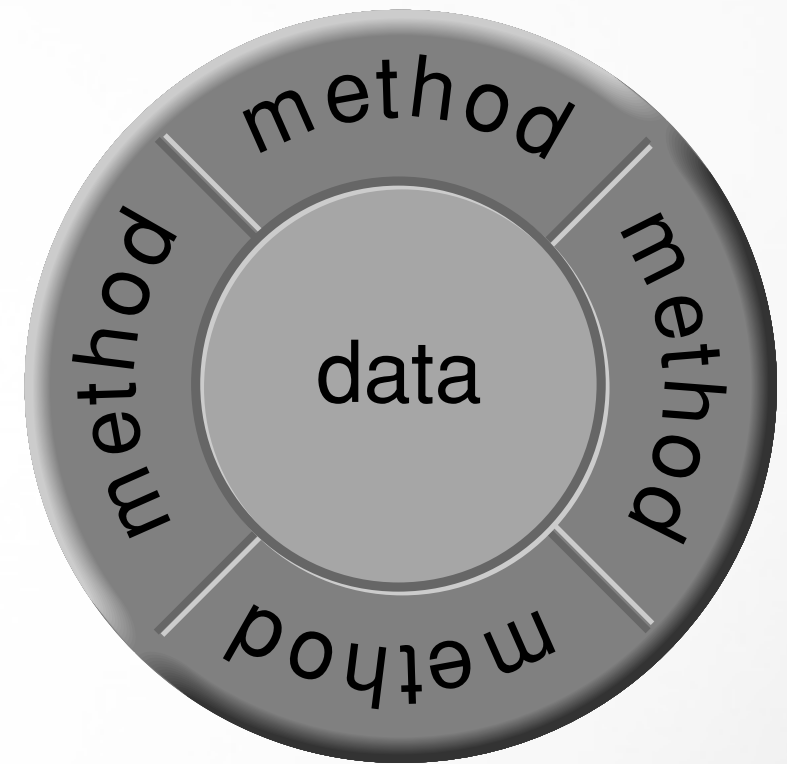
Use Actors!



100.000's of processes,
silly you!

Effect Containment

- Functional languages *disallow effects*
- Many object-oriented styles *encourage side effects*.
- Actors *confine effects*



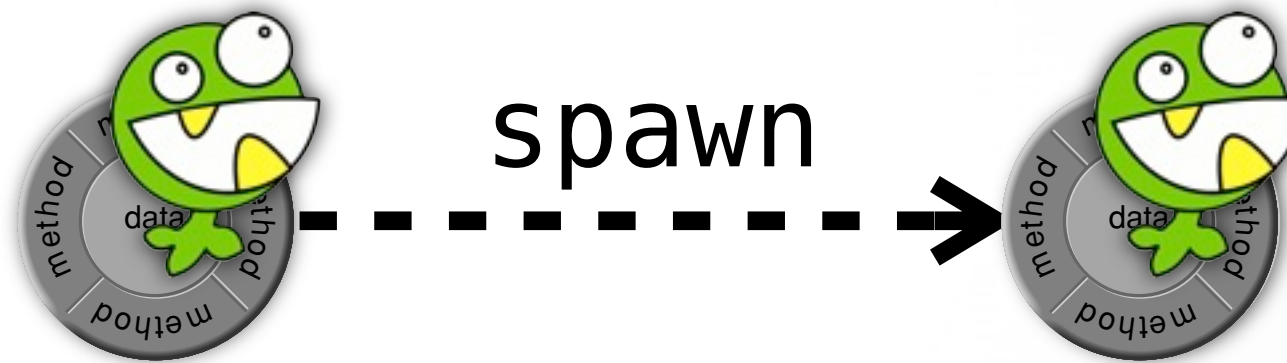
From C++ to Java

Garbage Collection

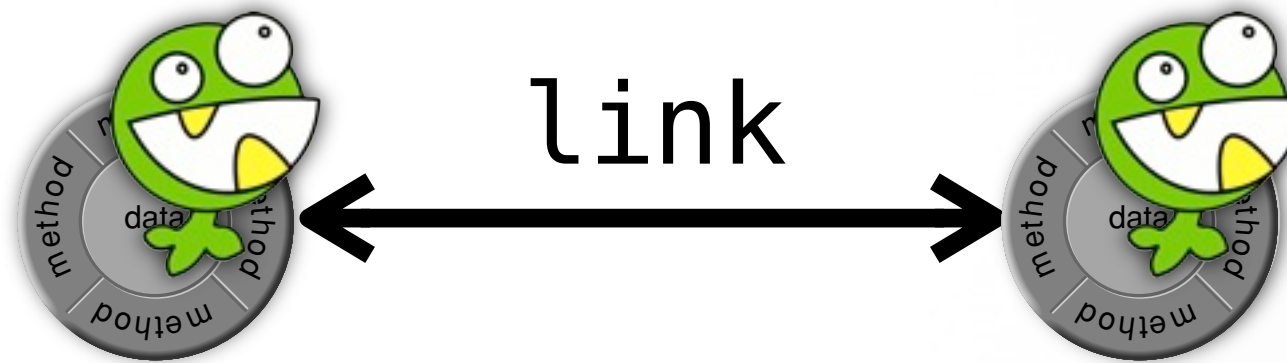
From Java to Erlang

State Confinement

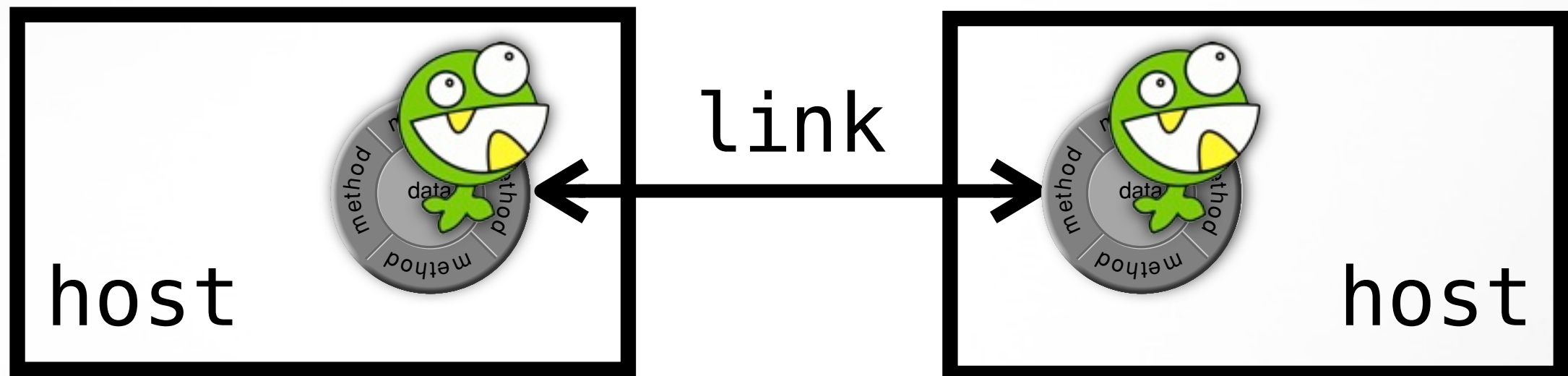
Activity Composition



Activity Composition



Activity Composition



“Let it Fail” philosophy



“Let it Fail” philosophy

- Write code with lots of assertions



“Let it Fail” philosophy

- Write code with lots of assertions
- Let a meta-level do fault handling



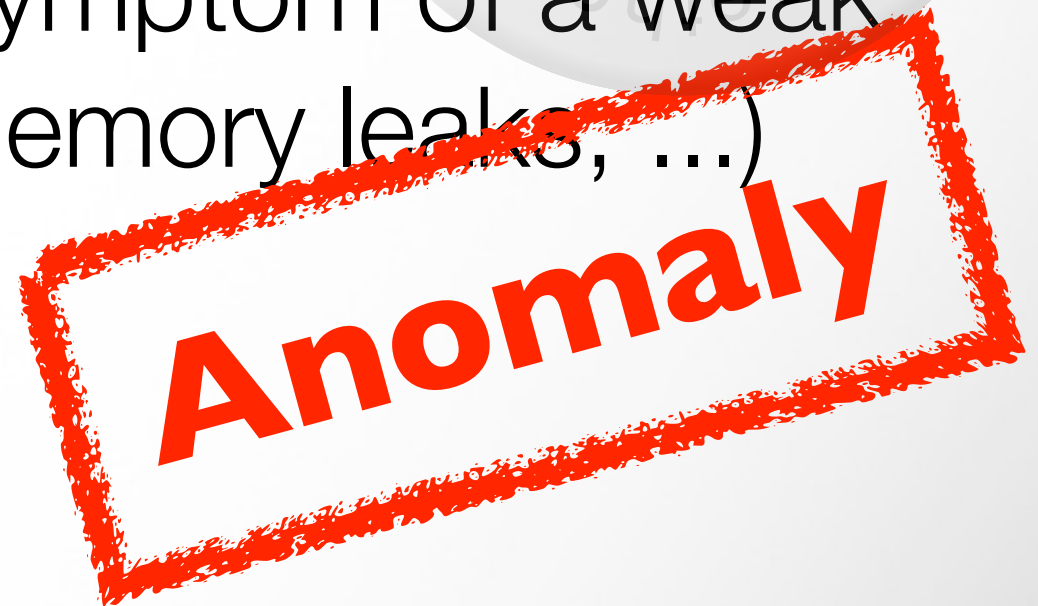
“Let it Fail” philosophy

- Write code with lots of assertions
- Let a meta-level do fault handling
- Defensive code is a symptom of a weak platform (segfaults, memory leaks, ...)

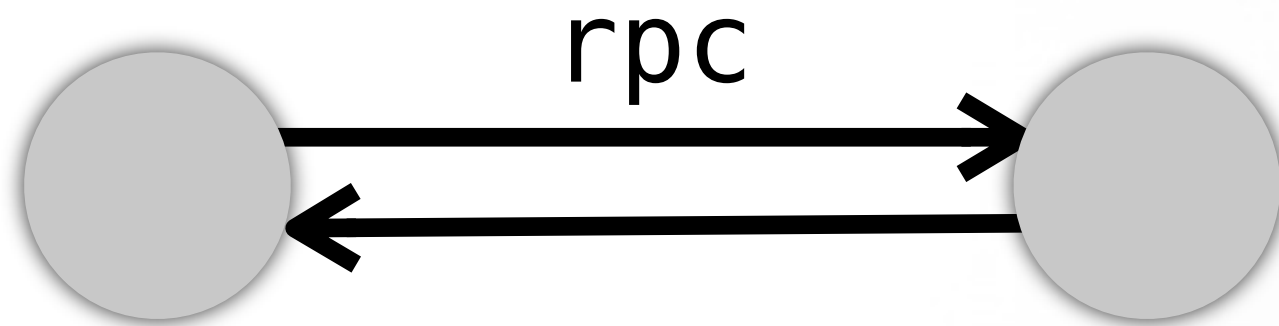


“Let it Fail” philosophy

- Write code with lots of assertions
- Let a meta-level do fault handling
- Defensive code is a symptom of a weak platform (segfaults, memory leaks, ...)

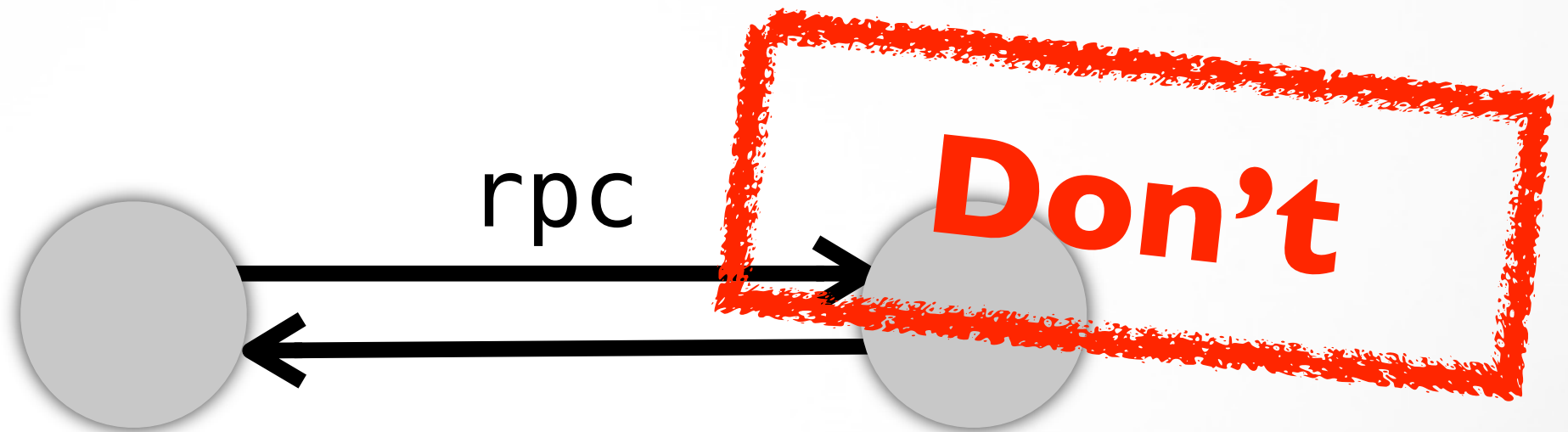


Martin Fowler's First Law of Distributed Objects Design



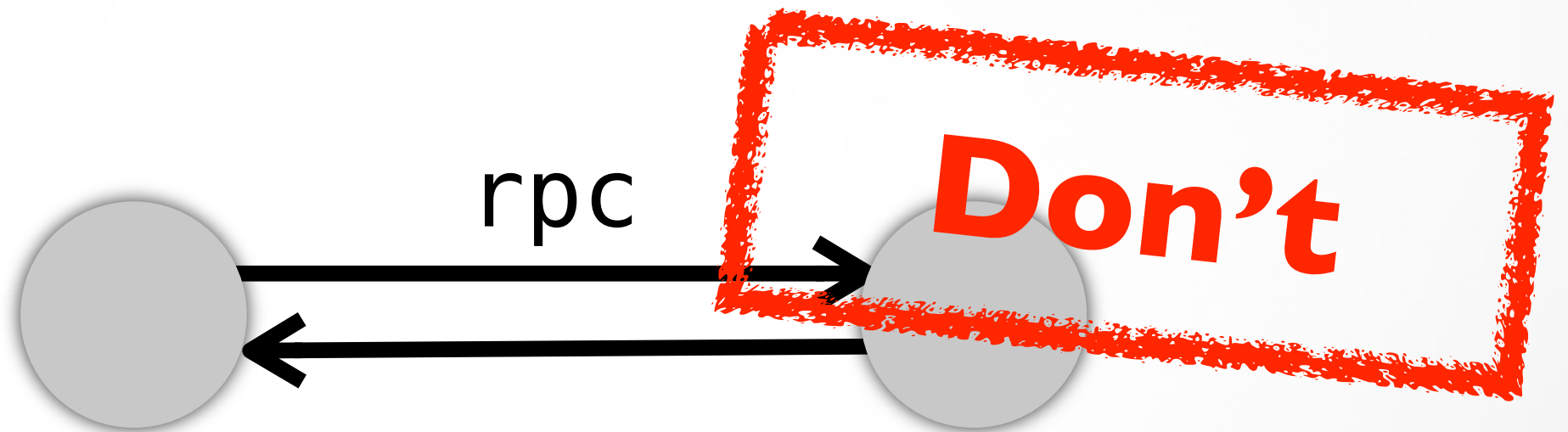
Starbucks doesn't use transactions

Martin Fowler's First Law of Distributed Objects Design



Starbucks doesn't use transactions

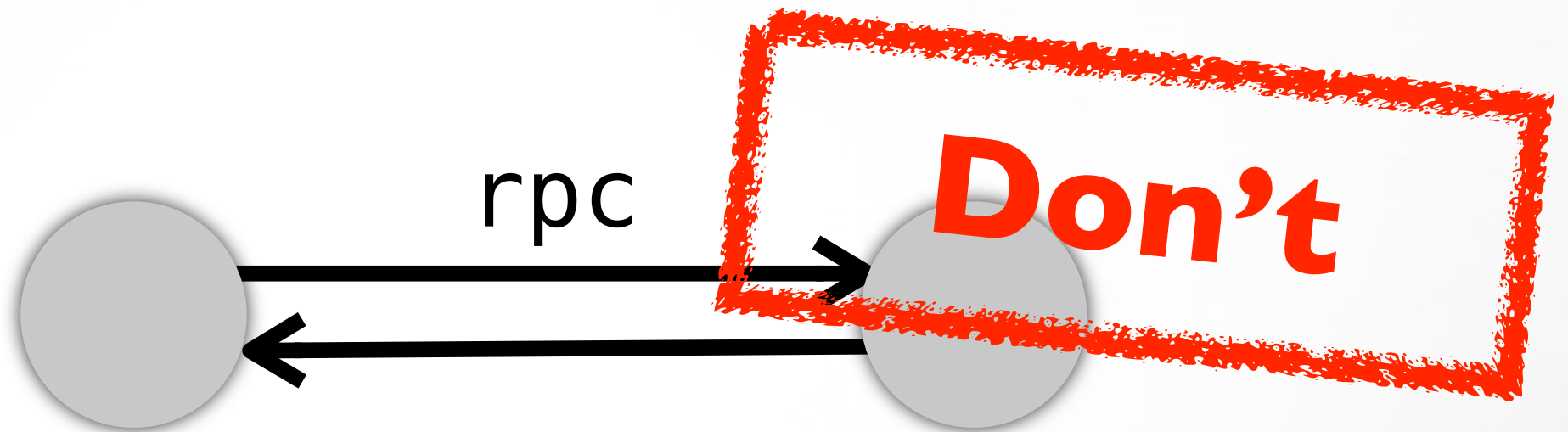
Martin Fowler's First Law of Distributed Objects Design



Starbucks doesn't use transactions

Steve Vinoski: *RPC and its Offspring: Convenient, Yet Fundamentally Flawed*

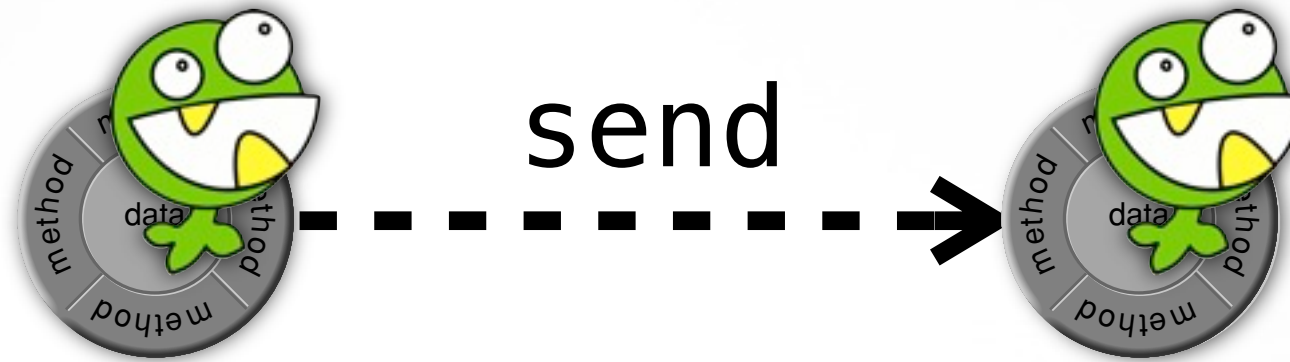
Martin Fowler's First Law of Distributed Objects Design



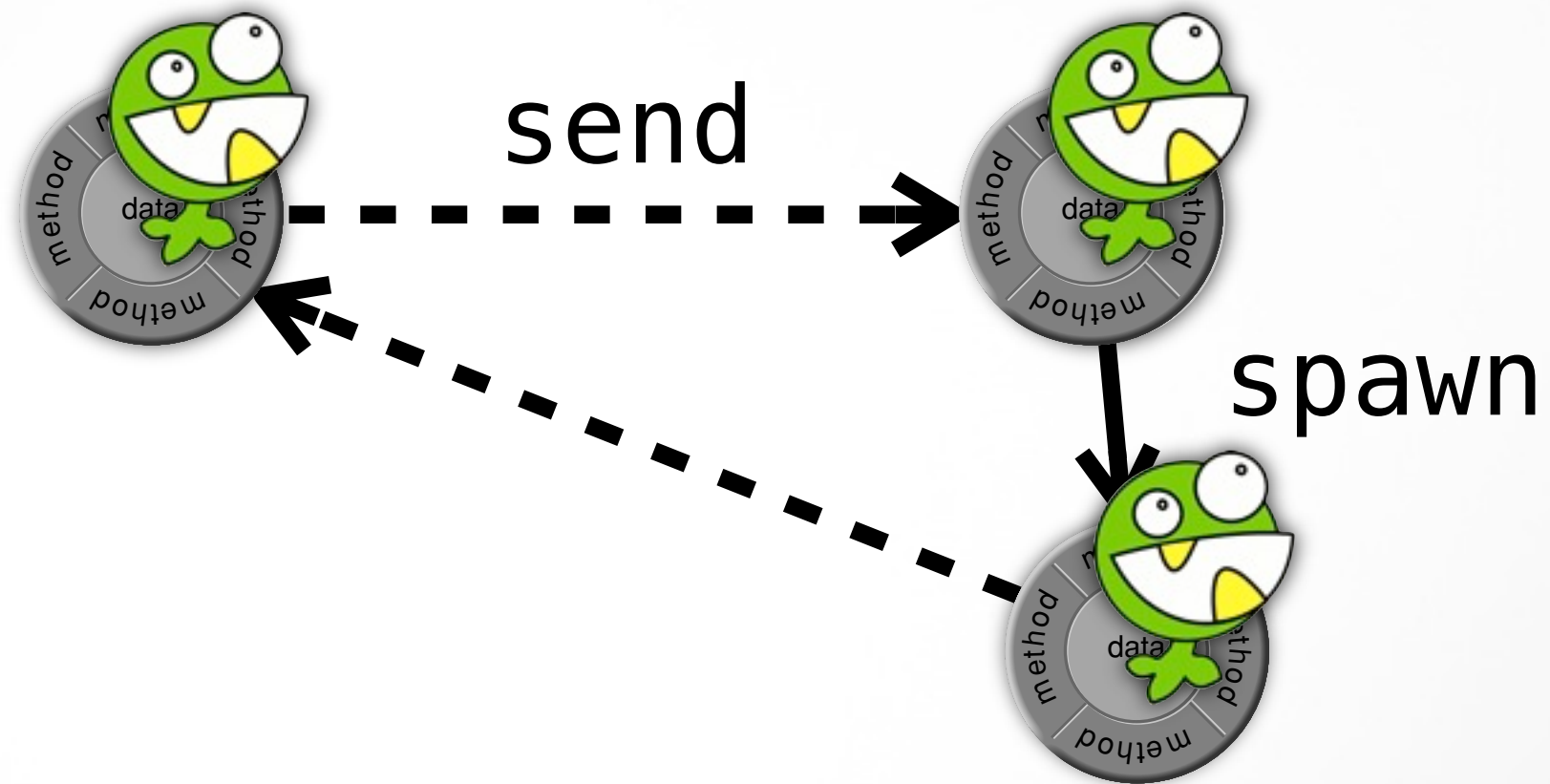
Starbucks doesn't use transactions

Steve Vinoski: *RPC and its Offspring: Convenient, Yet Fundamentally Flawed*

Actor Composition



Actor Composition



Abstractions

Abstractions

- Any abstraction (hiding code) is problematic distribute, persist, etc.

Abstractions

- Any abstraction (hiding code) is problematic distribute, persist, etc.
- You want to distribute/persist simple data (as in sql databases, document stores)

Abstractions

- Any abstraction (hiding code) is problematic distribute, persist, etc.
- You want to distribute/persist simple data (as in sql databases, document stores)
- JSON's popularity is a testament to this anomaly.

Abstractions

- Any abstraction (hiding code) is problematic distribute, persist, etc.
- You want to distribute/persist simple data (as in sql databases, document stores)
- JSON's popularity is a testament to this anomaly.



Simple Values

Simple Values

- Tuple, List, Record, Number, String

Simple Values

- Tuple, List, Record, Number, String
- Pattern matching is polymorphism for values

Simple Values

- Tuple, List, Record, Number, String
- Pattern matching is polymorphism for values
- Erlang data stores (like Mnesia) just store values, not bytes or objects.

Simple Values

- Tuple, List, Record, Number, String
- Pattern matching is polymorphism for values
- Erlang data stores (like Mnesia) just store values, not bytes or objects.
- Too much of my Java programs are boilerplate code.

Simple Values

- Tuple, List, Record, Number, String
- Pattern matching is polymorphism for values
- Erlang data stores (like Mnesia) just store values, not bytes or objects.
- Too much of my Java programs are boilerplate code.



“Object” Model Anomalies

- Thread & Locks
- Interfaces with Fixed API
- Defensive Code
- RPC/RMI
- Boilerplate code for persistence

Actor “Solutions”

- Processes w/ state containment
- Protocols
- Let it Fail
- Async Messaging
- Send & store simple Data

Back in The Real World

Erlang/OTP

Open Telecommunications Platform

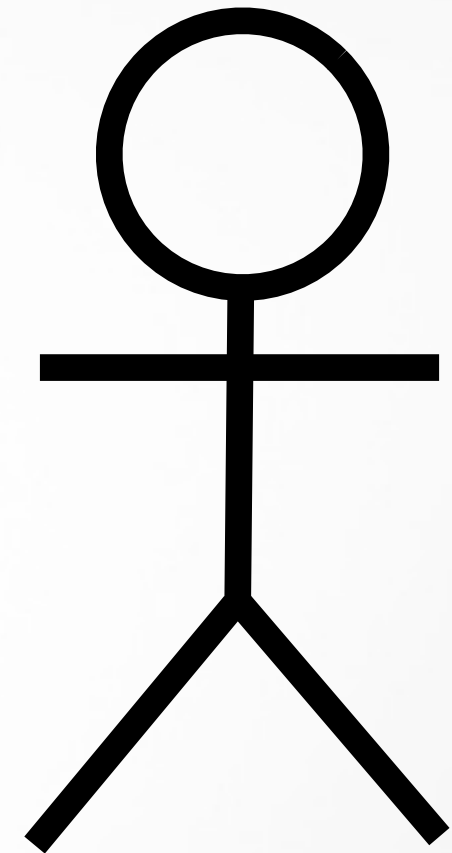
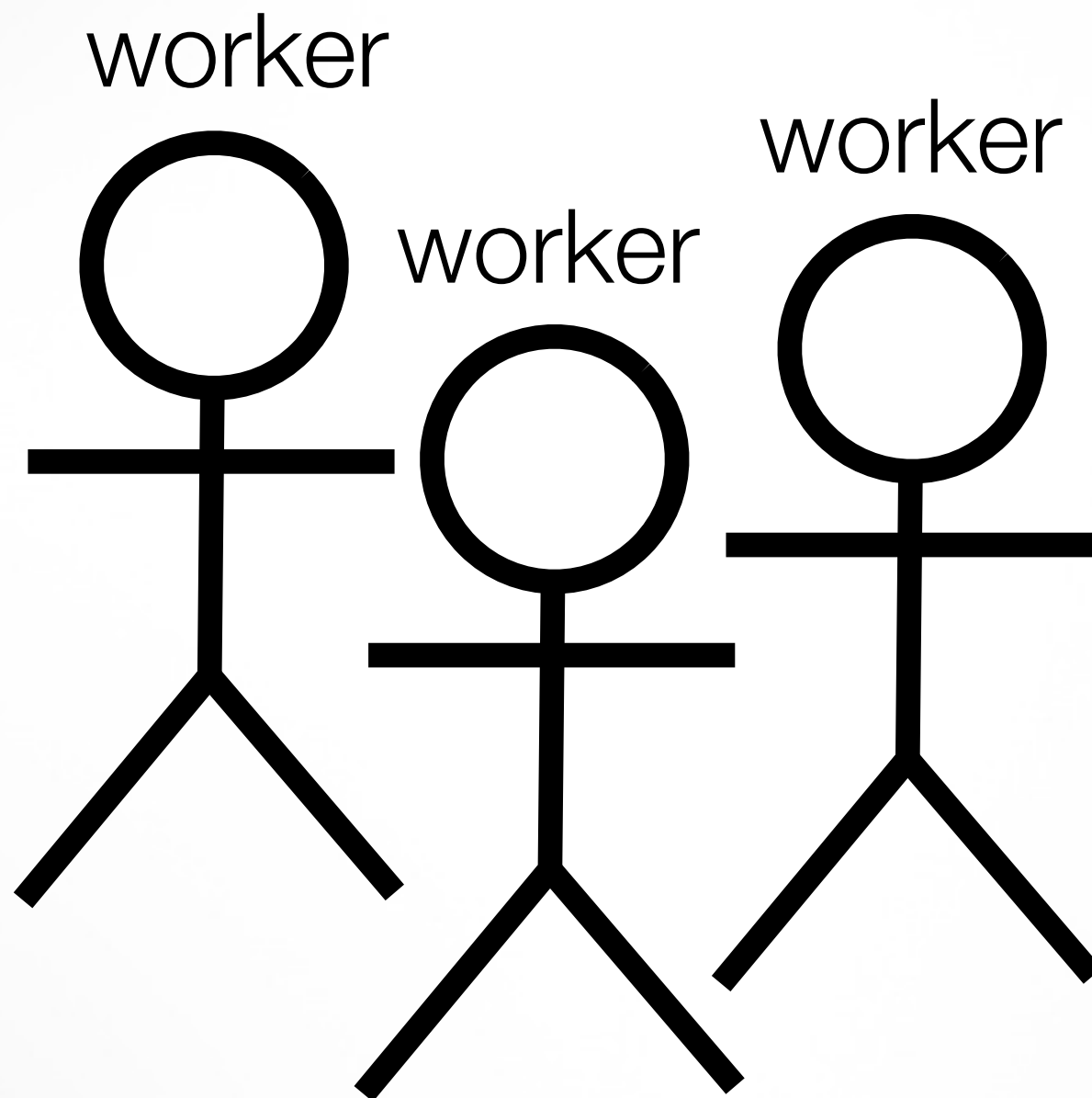
- Embedded Distributed Systems
- High Availability
- In-Production Upgrades

OTP Actor Behaviors

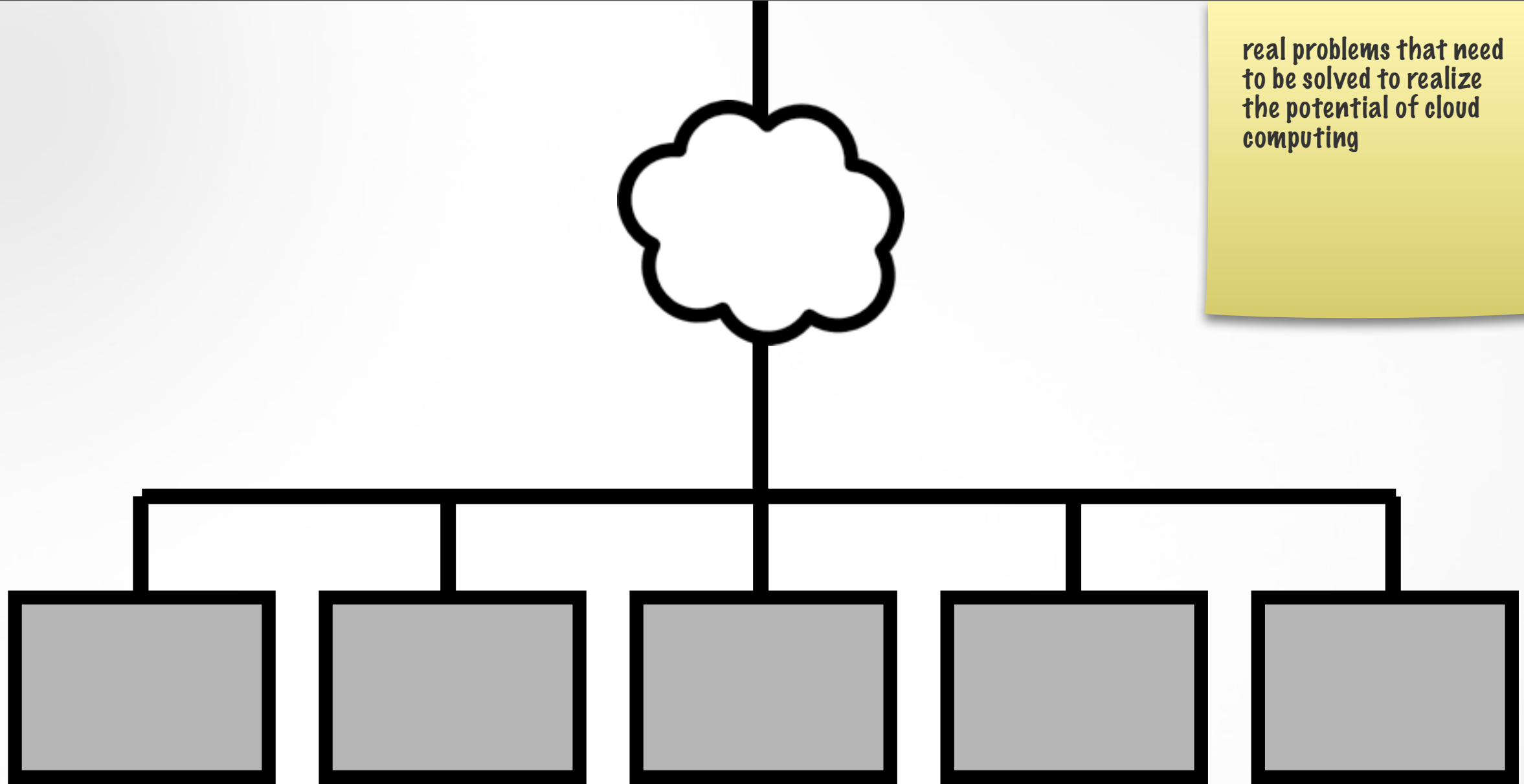
- Servers
- Event Handlers
- Finite State Machine
- Supervisors
- Networking: TCP, HTTP

AML

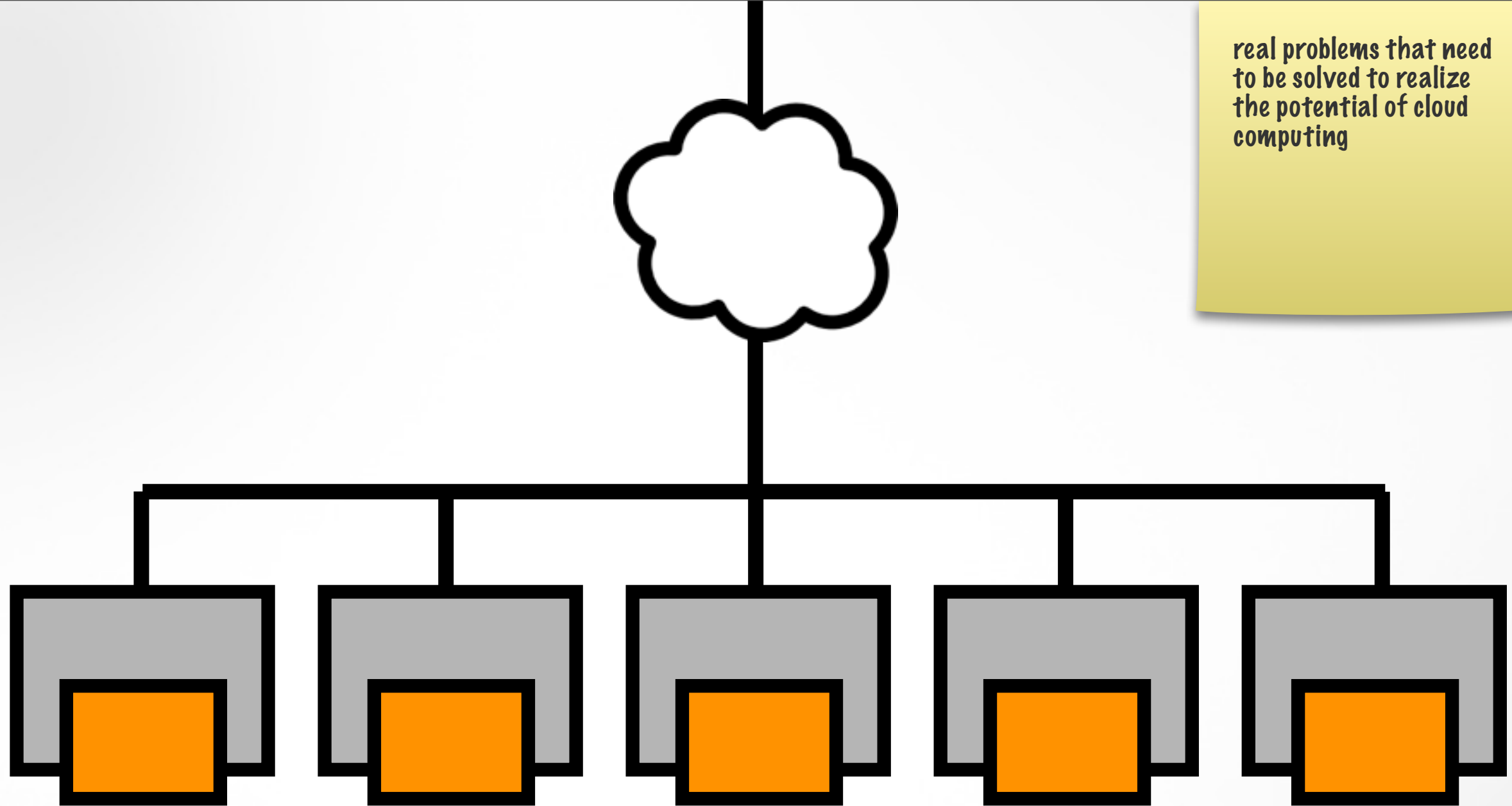
supervisor



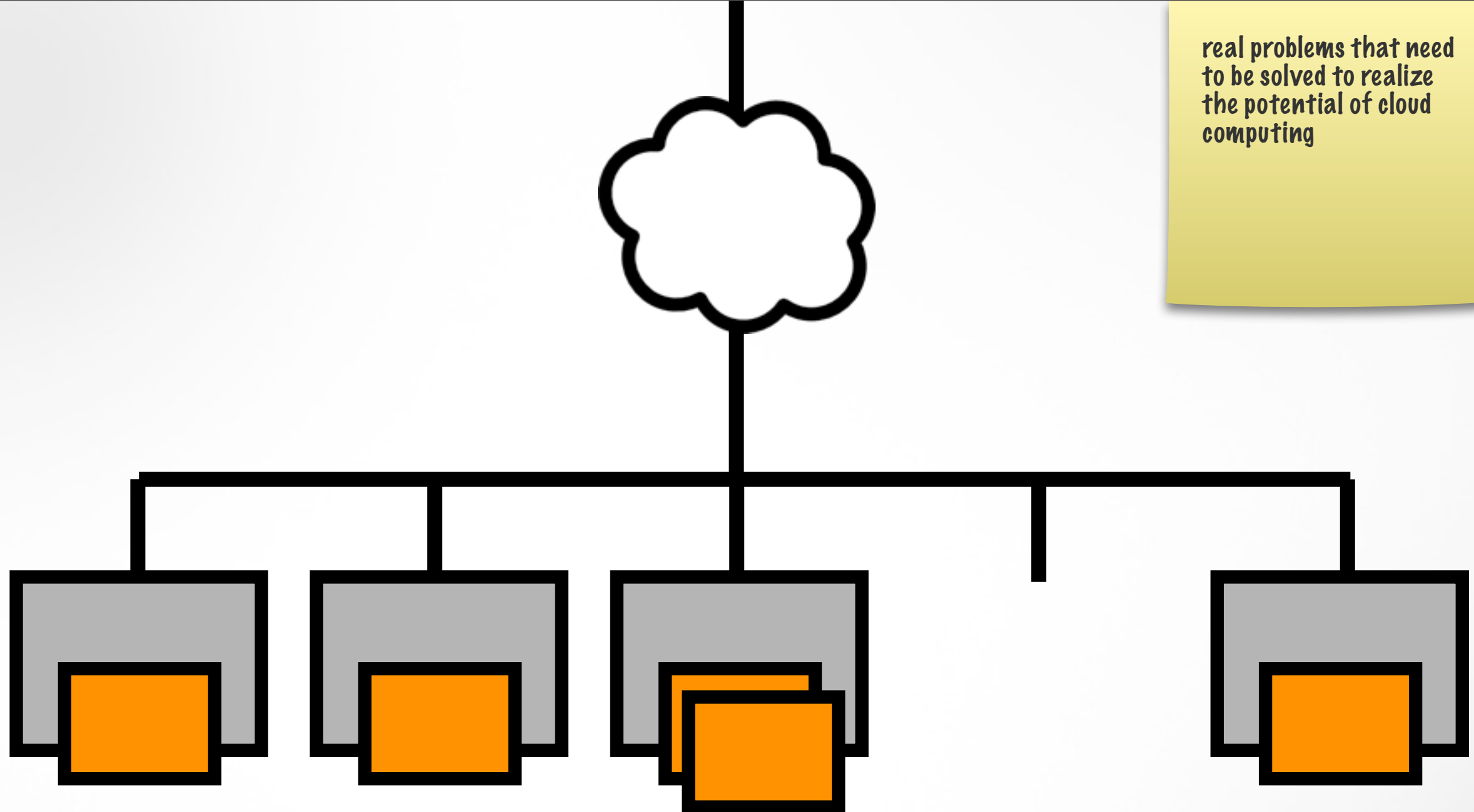
real problems that need
to be solved to realize
the potential of cloud
computing



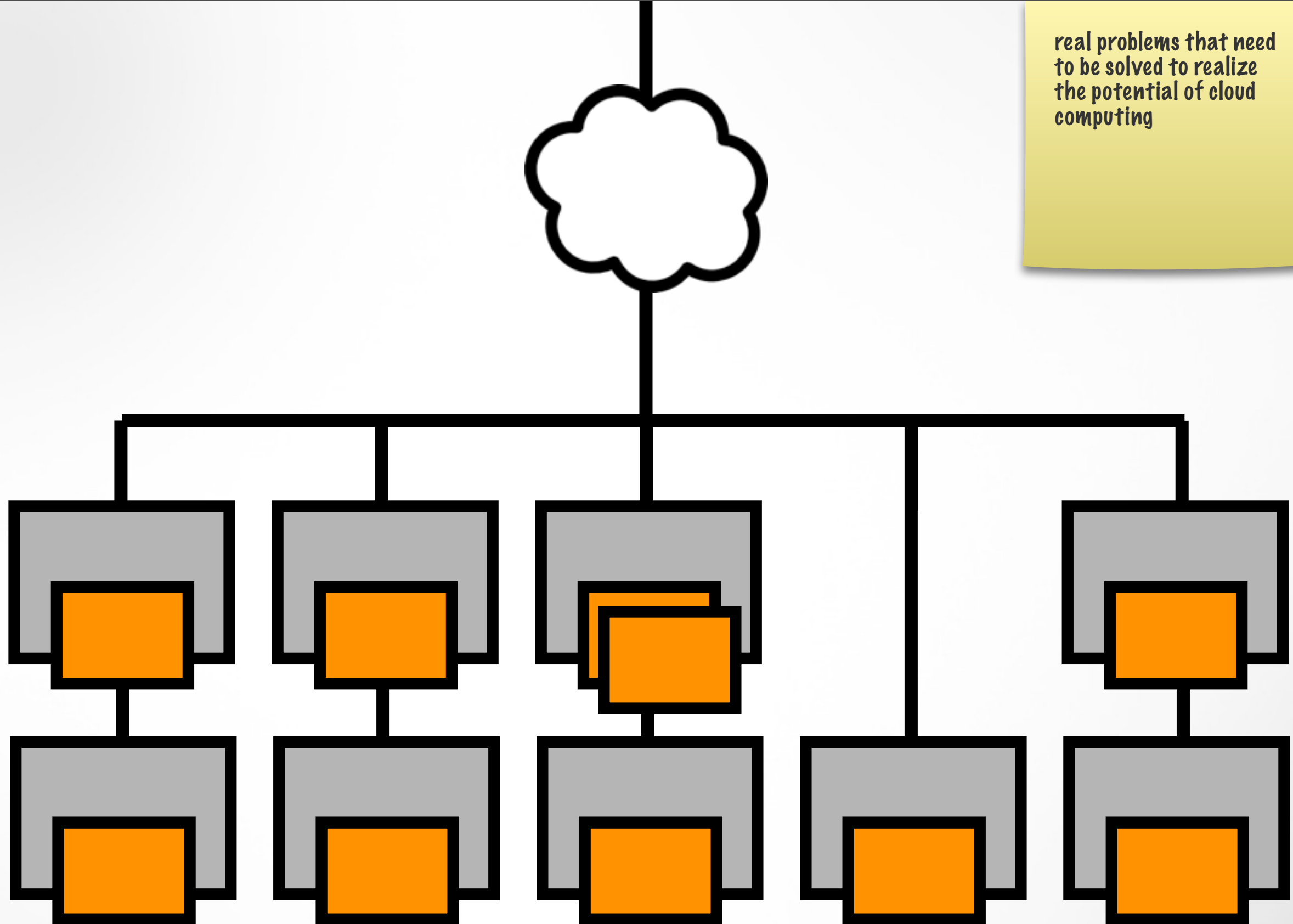
real problems that need
to be solved to realize
the potential of cloud
computing



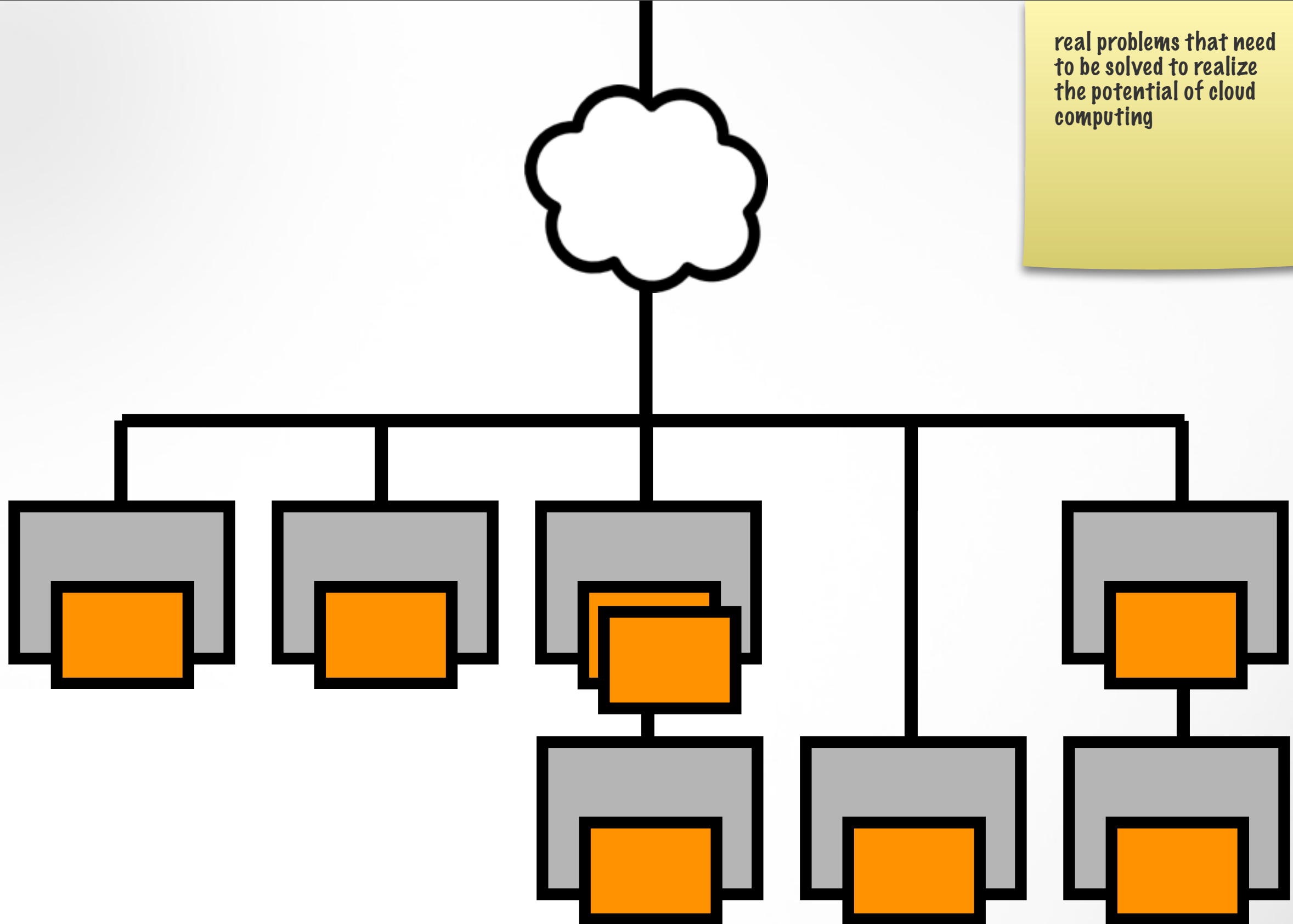
real problems that need
to be solved to realize
the potential of cloud
computing



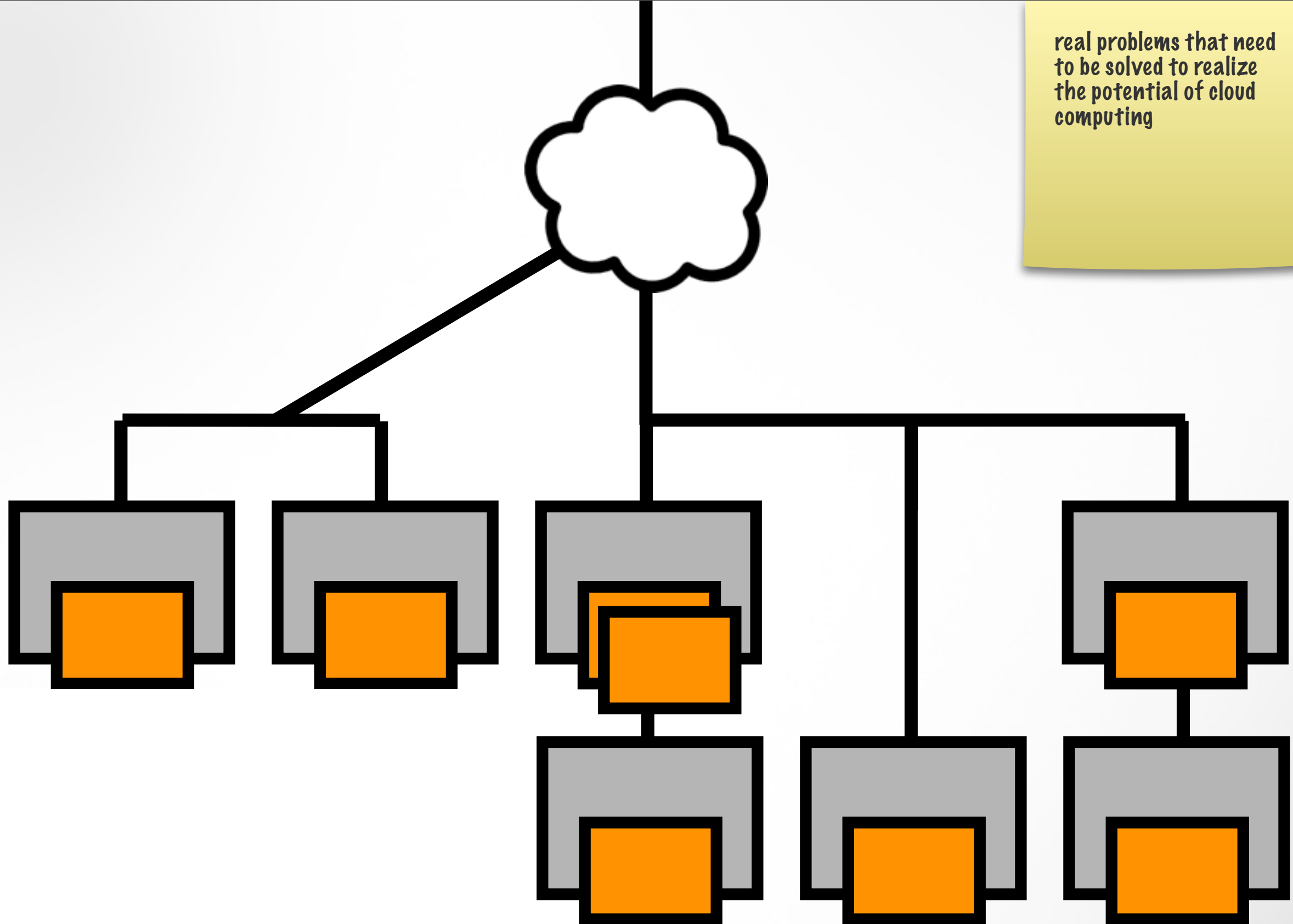
real problems that need
to be solved to realize
the potential of cloud
computing

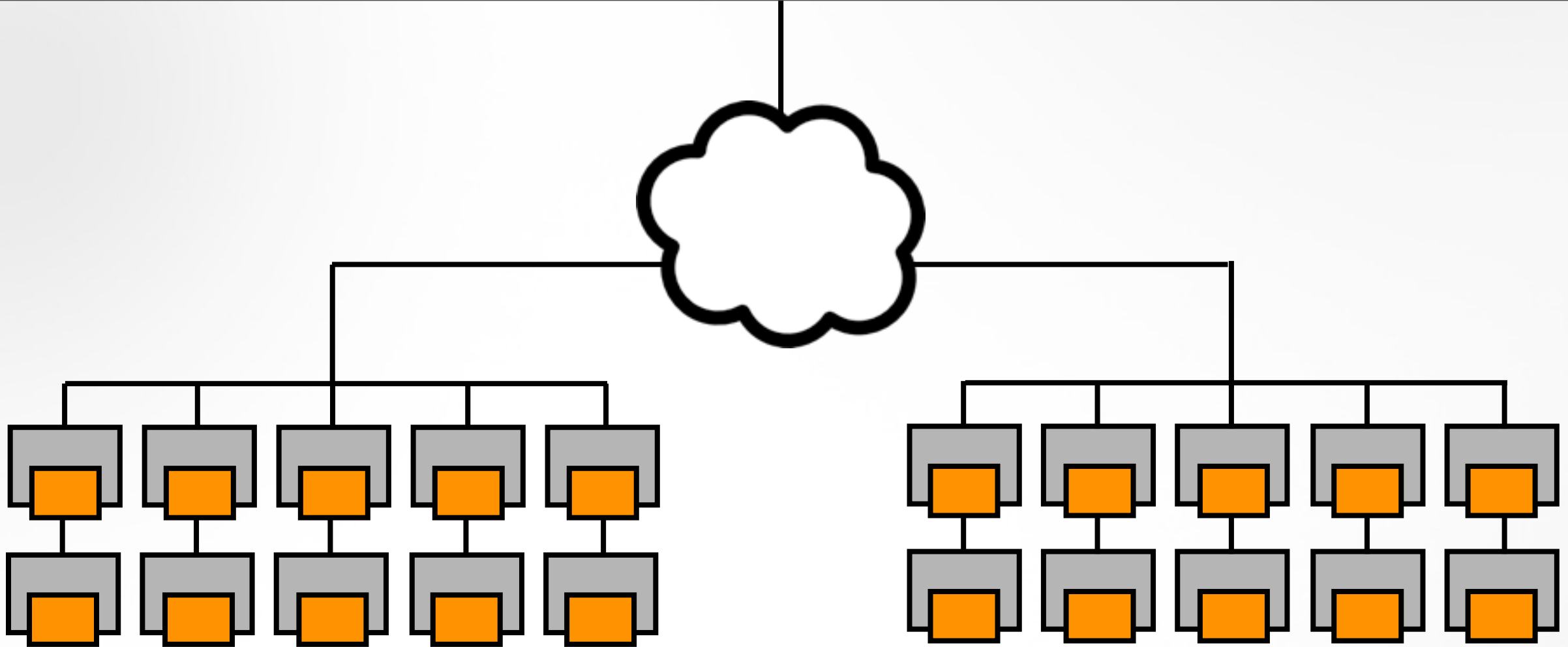


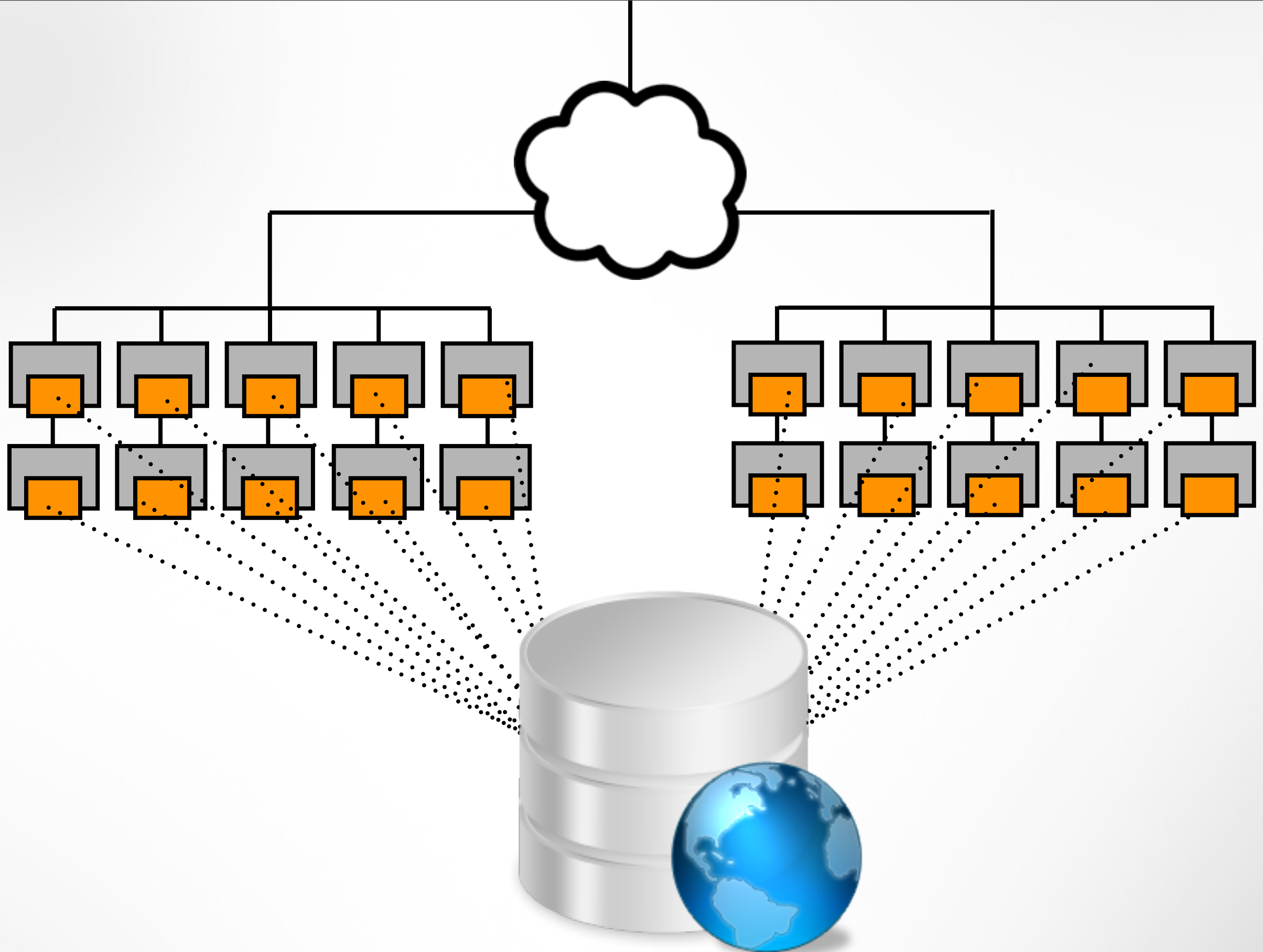
real problems that need
to be solved to realize
the potential of cloud
computing

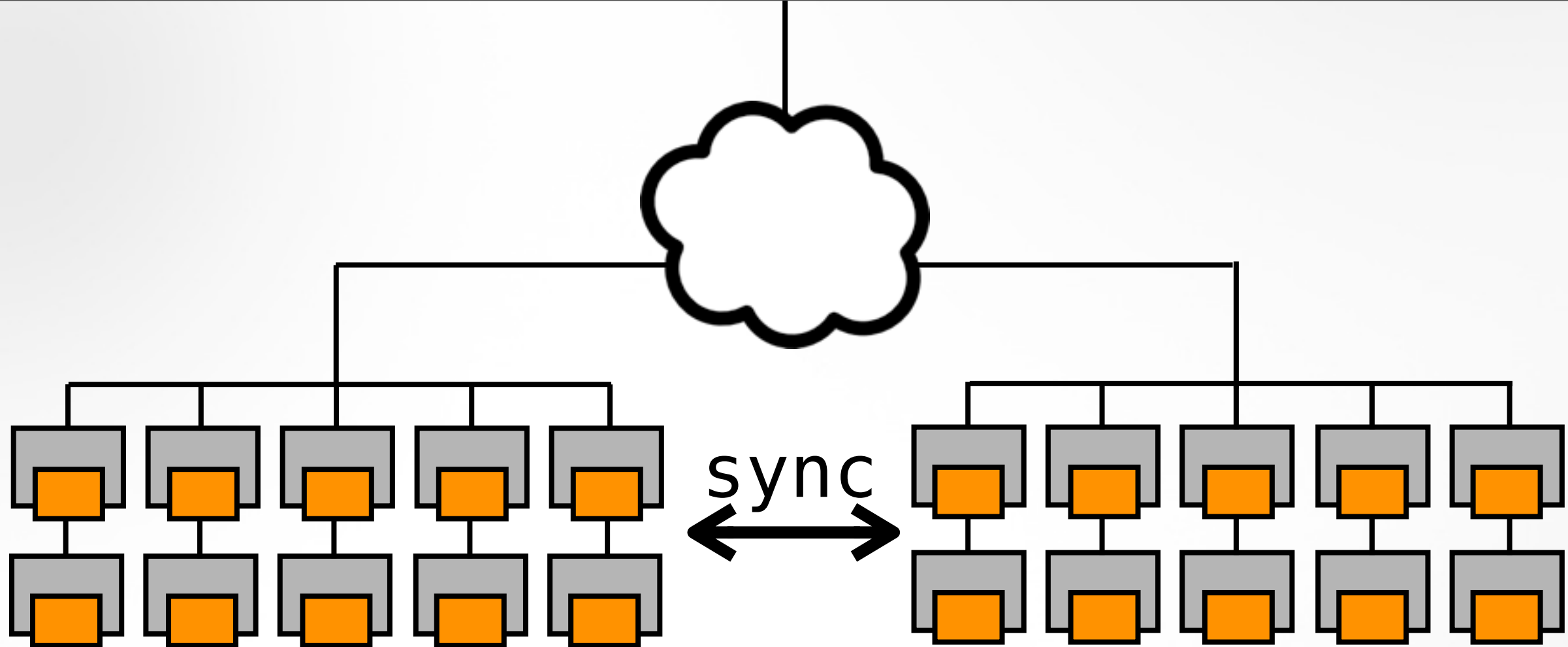


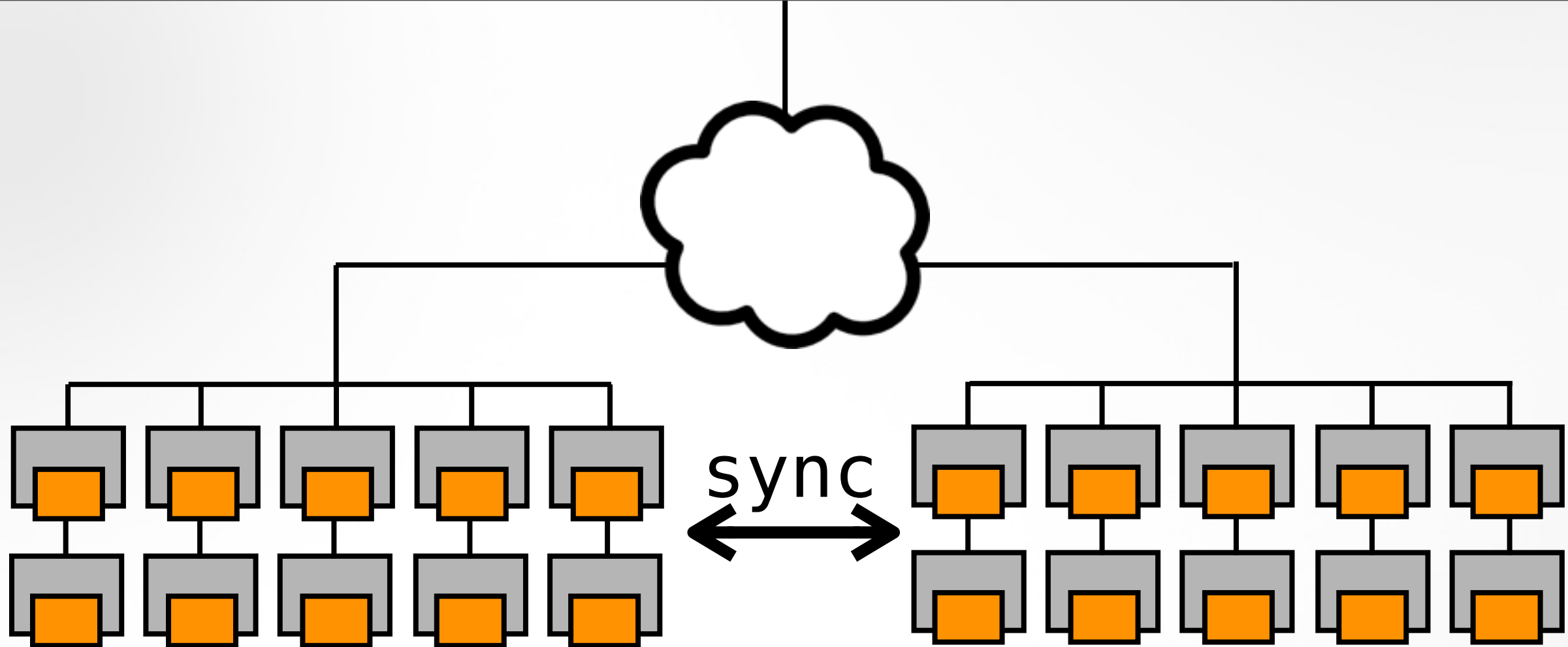
real problems that need
to be solved to realize
the potential of cloud
computing



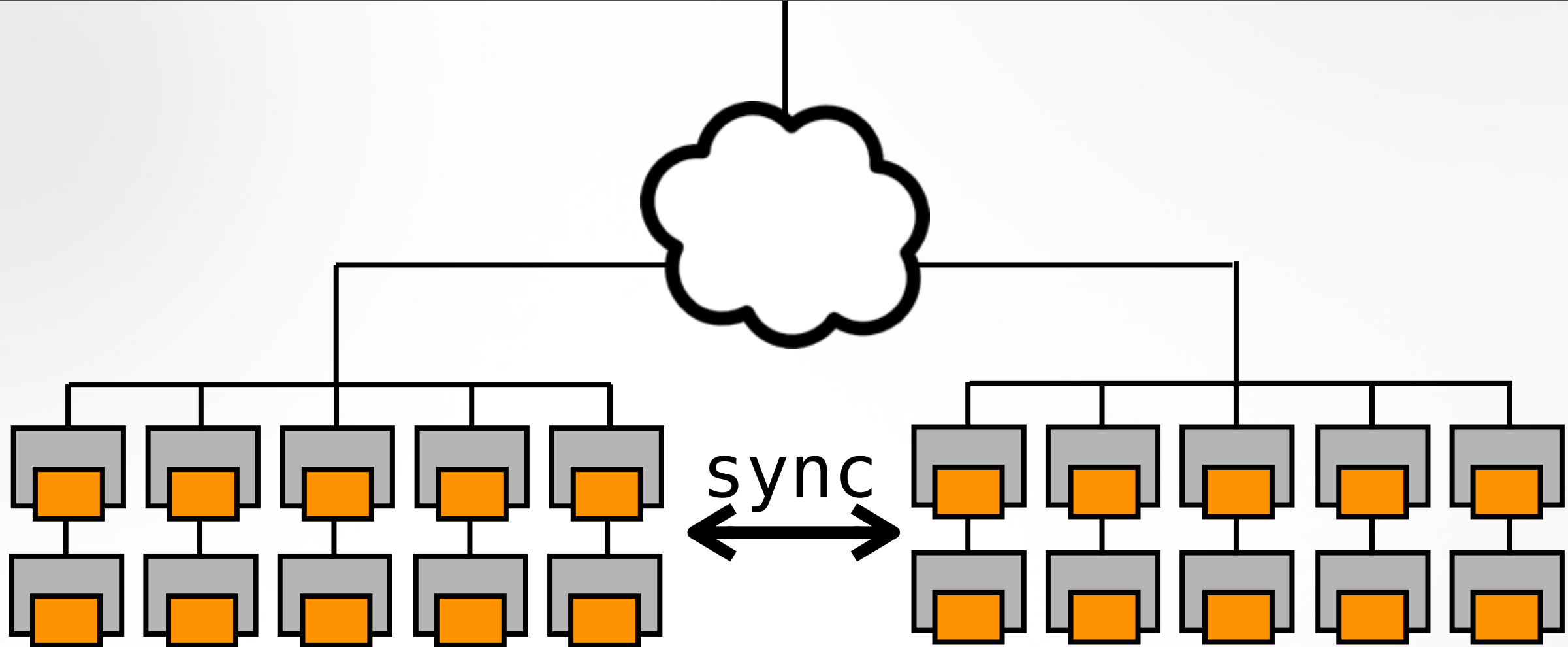






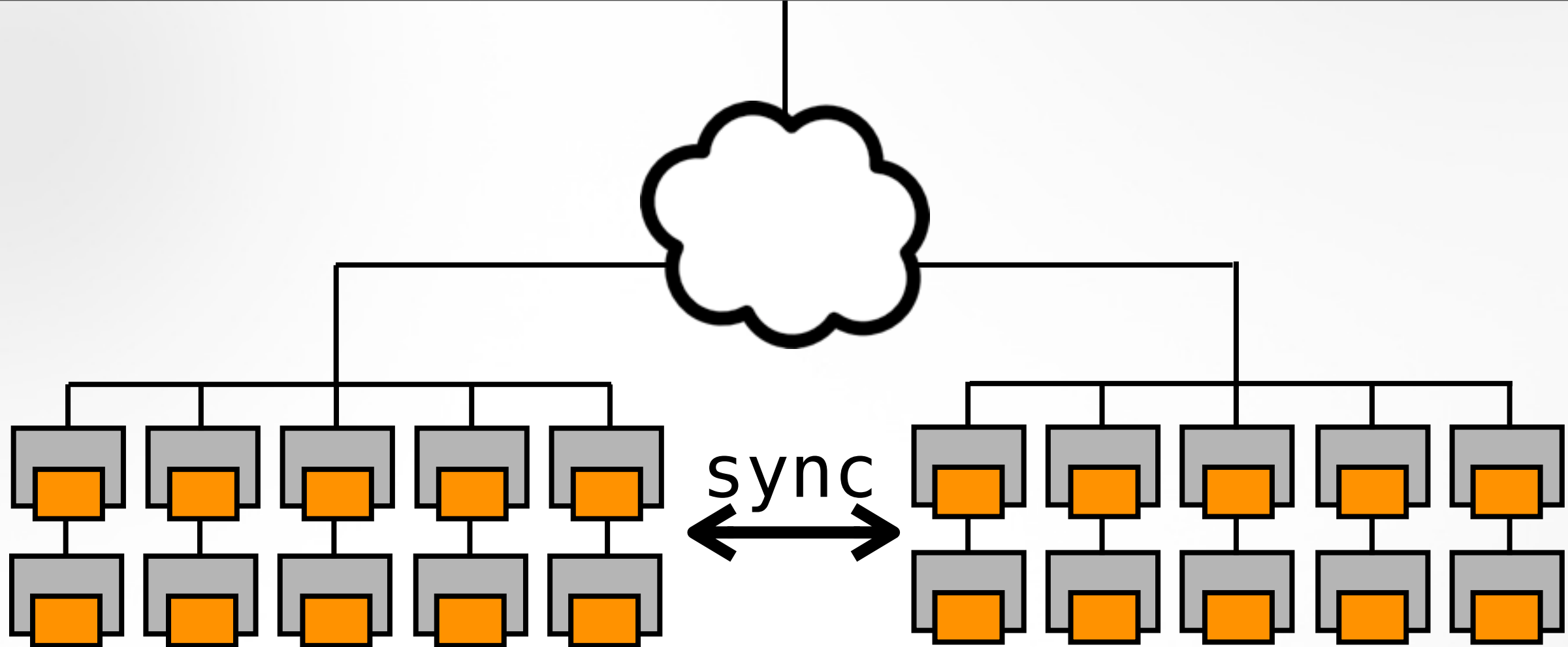


Async Messaging

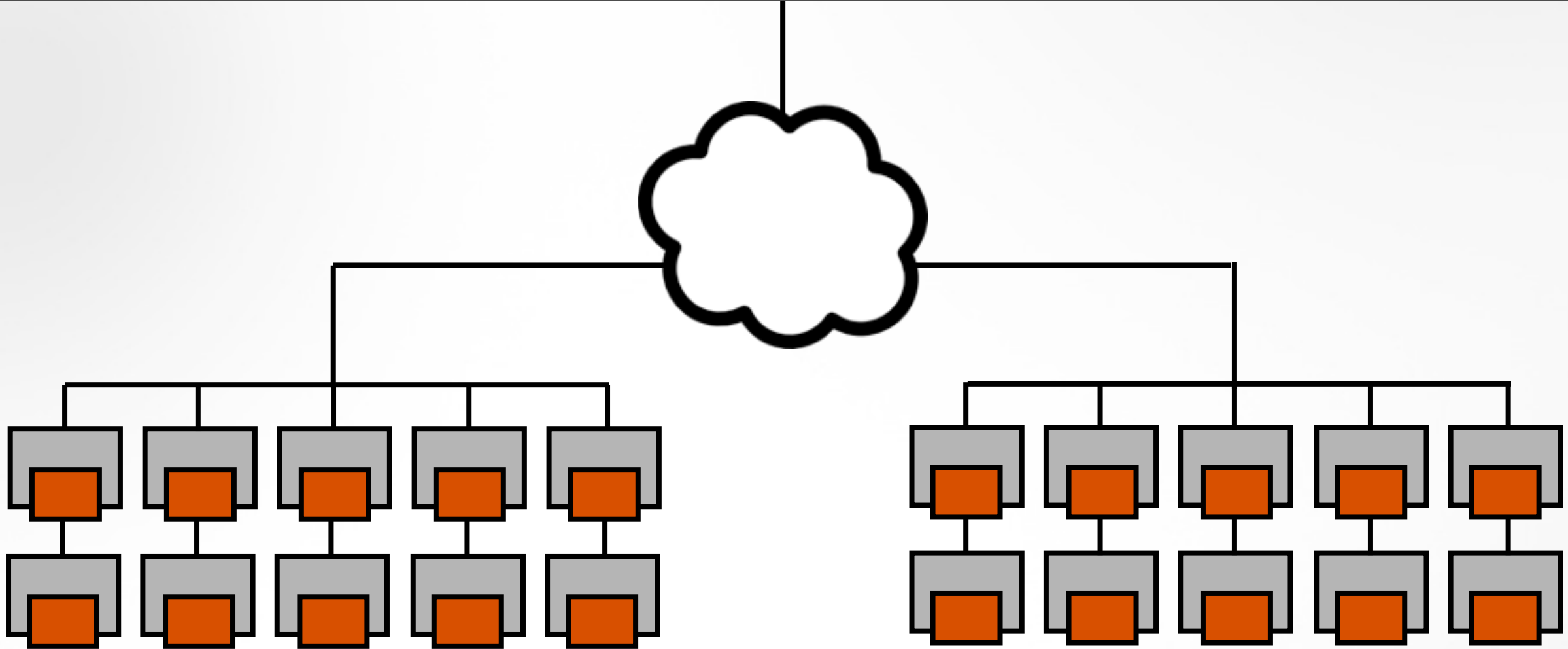


Async Messaging

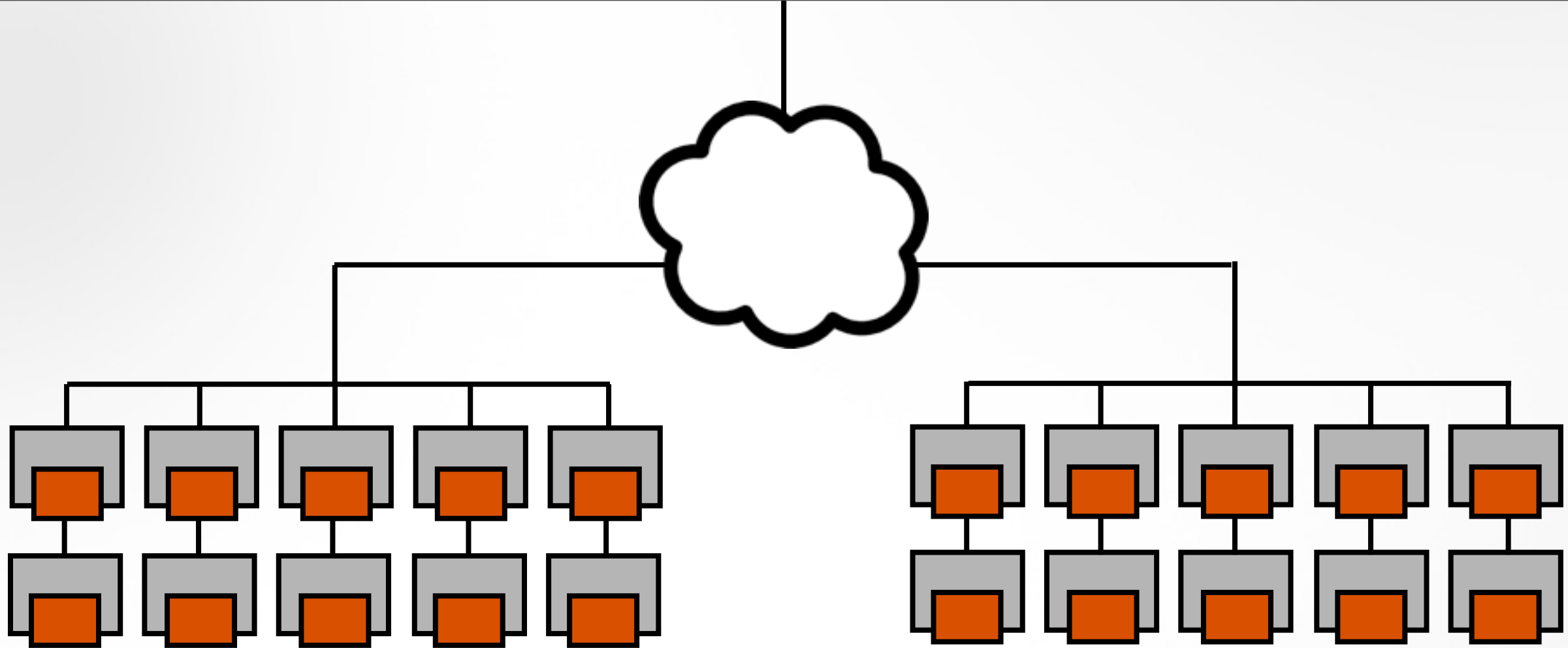
State [Fault] Containment



Async Messaging
State [Fault] Containment
Fault Monitoring



 **riak**



Yes, write-conflicts do occur.
Deal with it!

Shared Medicine Card



Shared Medicine Card



Shared Medicine Card



Who Else Uses Erlang?

facebook



riak

T-Mobile

klarna



RabbitMQ
Open Source Enterprise Messaging

Java is still important



Java is still important

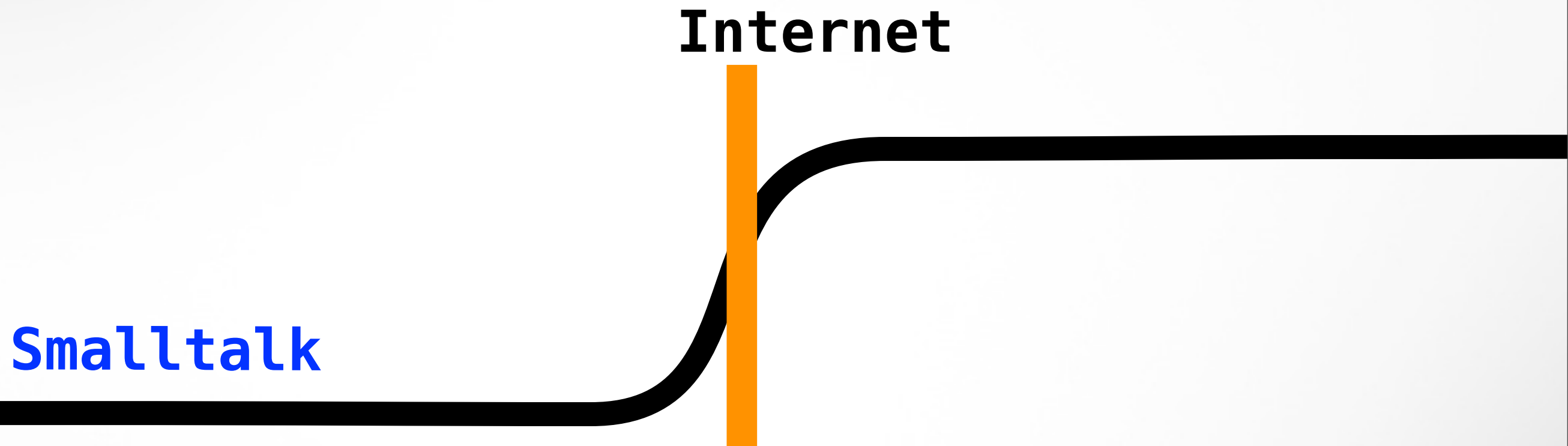


The Actor Revolution?

Smalltalk



The Actor Revolution?





Erlang

Cloud, Multi-Core
Integration, Coordination
Fault Tolerance, Availability, QoS

Cloud

Erlang

Cloud, Multi-Core
Integration, Coordination
Fault Tolerance, Availability, QoS

Thank You.

@drkrab



*Java either steps up,
or something else will.*
—Cameron Purdy

Thank You.

@drkrab



*Java either steps up,
or something else will.*
—Cameron Purdy

You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。

Thank You.

@drkrab



