What is Research in Programming Languages?

Prof. Dr. Guido Salvaneschi

School of Computer Science

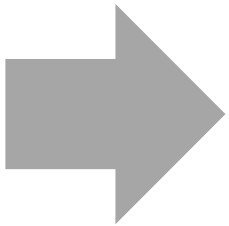
University of St.Gallen

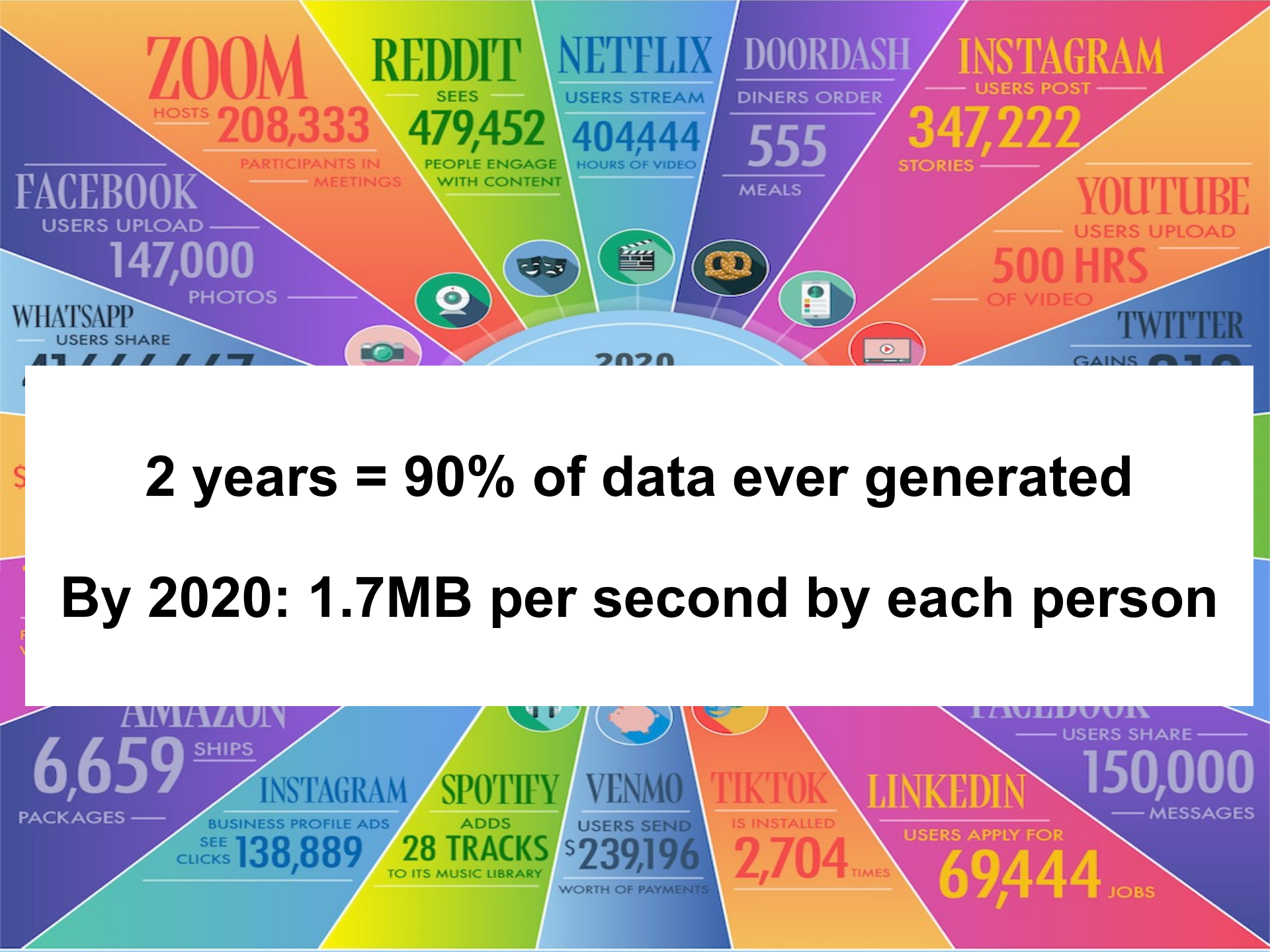
The World as we Knew it

```
use AdventureWorksDW2008R2

SELECT e.DepartmentName, COUNT(*)
FROM [dbo].[DimEmployee] AS e
WHERE e.Gender = 'F' and e.SickLea
GROUP BY e.DepartmentName
HAVING COUNT(*) > 3
ORDER BY EmployeeCount DESC
```

Results | Messages

	DepartmentName	EmployeeCount
1	Production	29
2	Purchasing	6
3	Finance	5
4	Information Services	4





2 years = 90% of data ever generated

By 2020: 1.7MB per second by each person

Real Time Processing

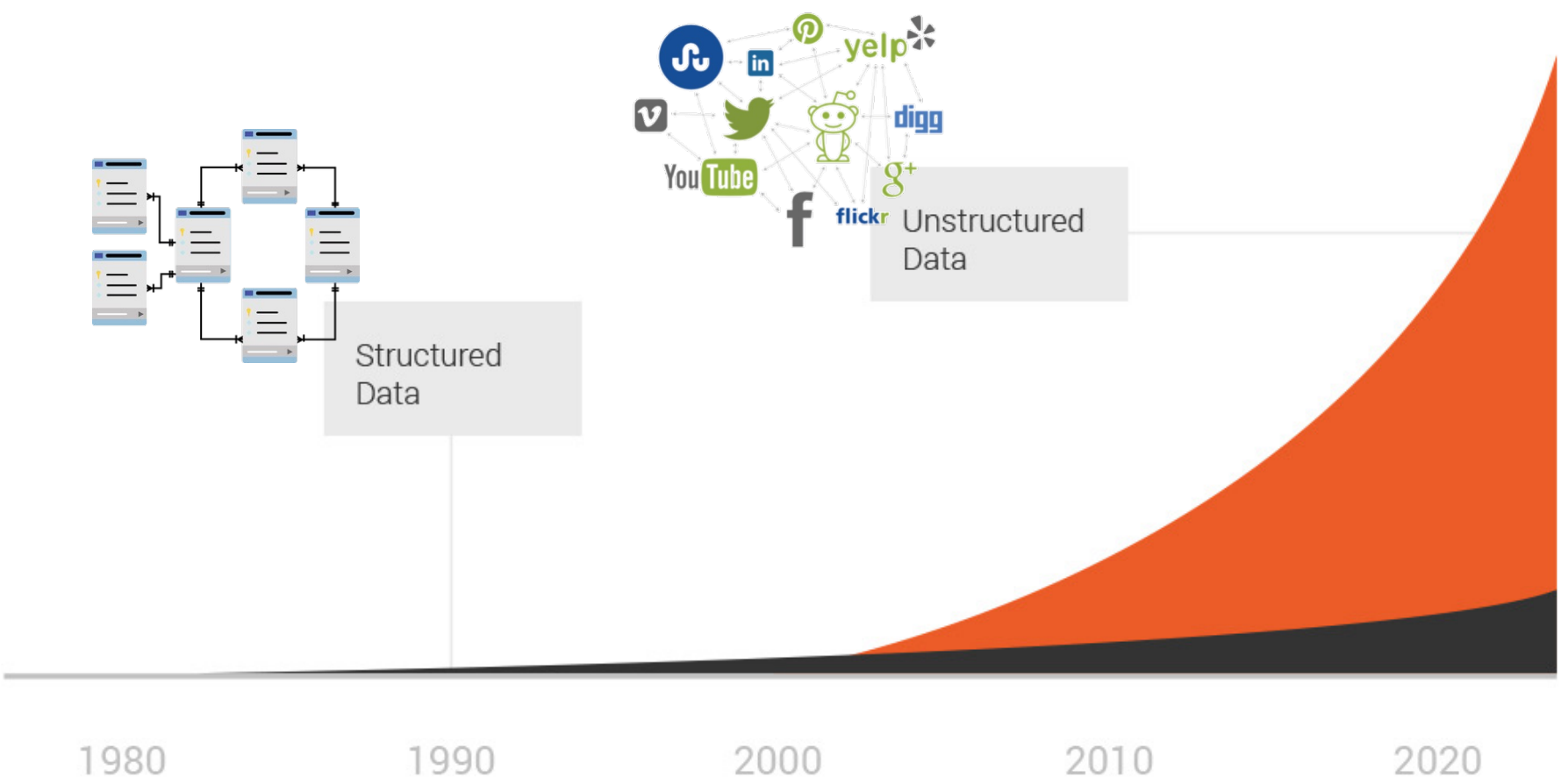
Fraud Detection

Real Time Business Intelligence

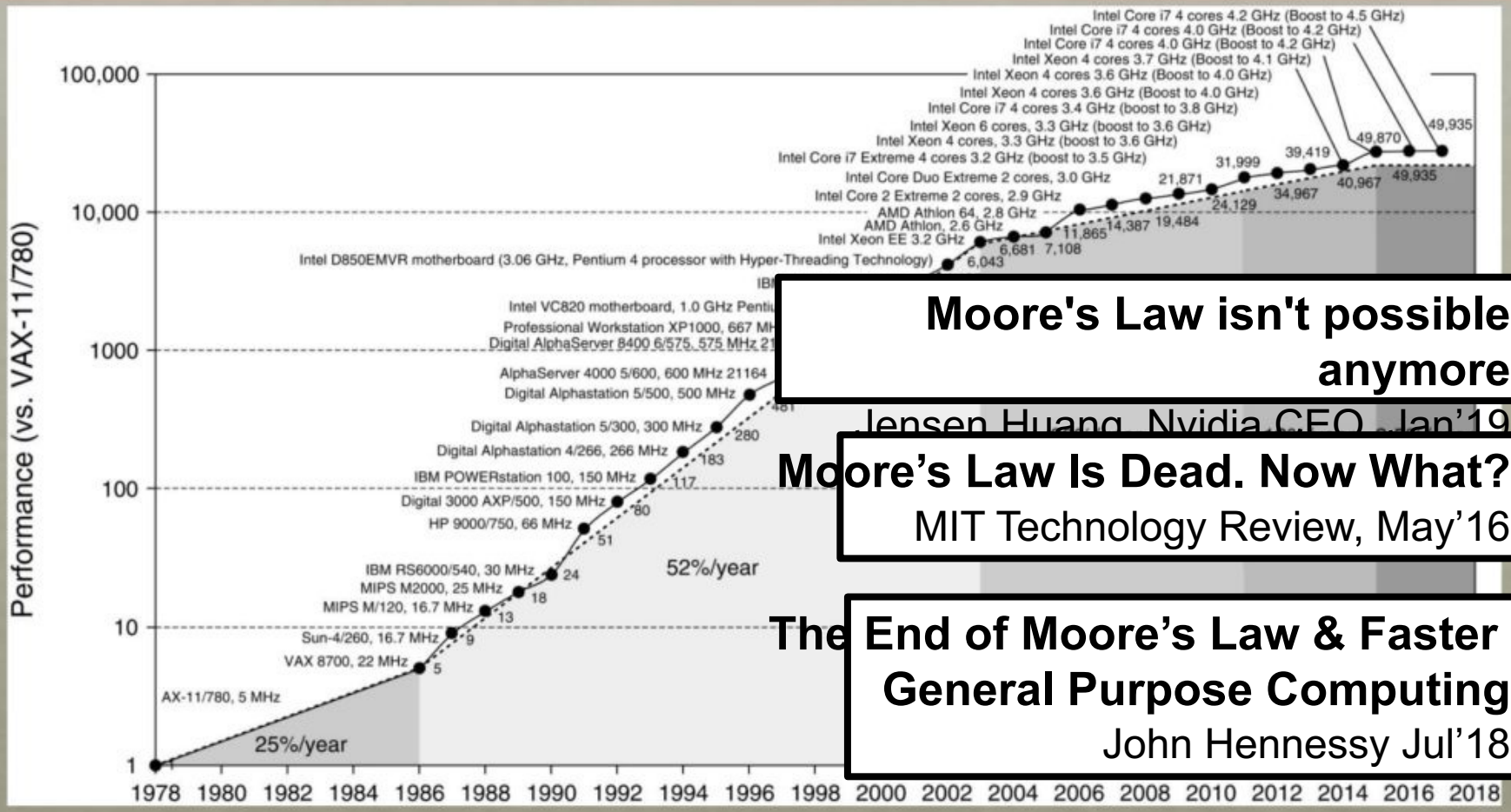
Cloud Monitoring

Structured and Unstructured Data

IDC and EMC project that data will grow
to 40 ZB by 2020



UNIPROCESSOR PERFORMANCE (SINGLE CORE)



Moore's Law isn't possible anymore

Jensen Huang, Nvidia CEO, Jan'19

Moore's Law Is Dead. Now What?

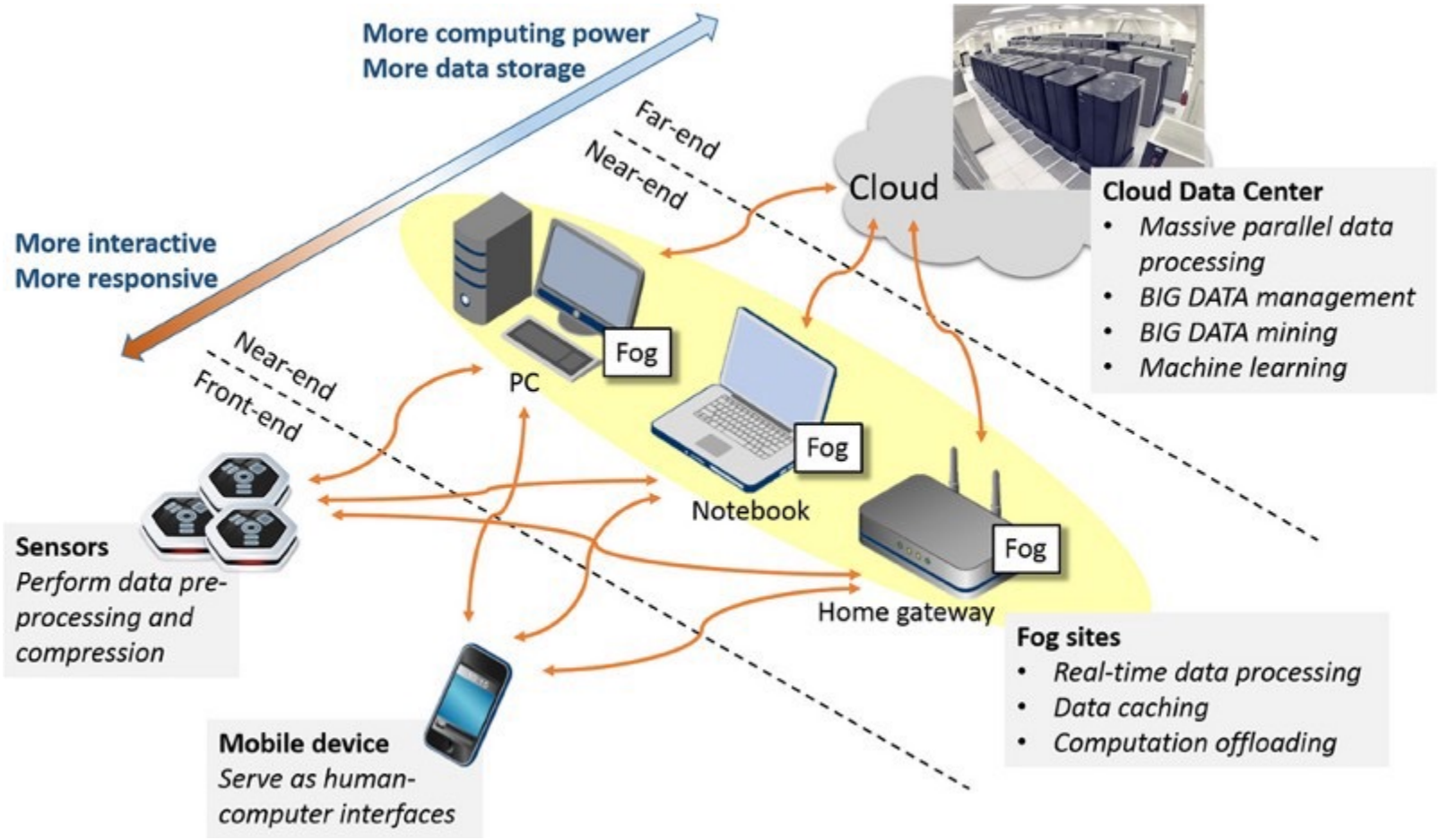
MIT Technology Review, May'16

The End of Moore's Law & Faster General Purpose Computing

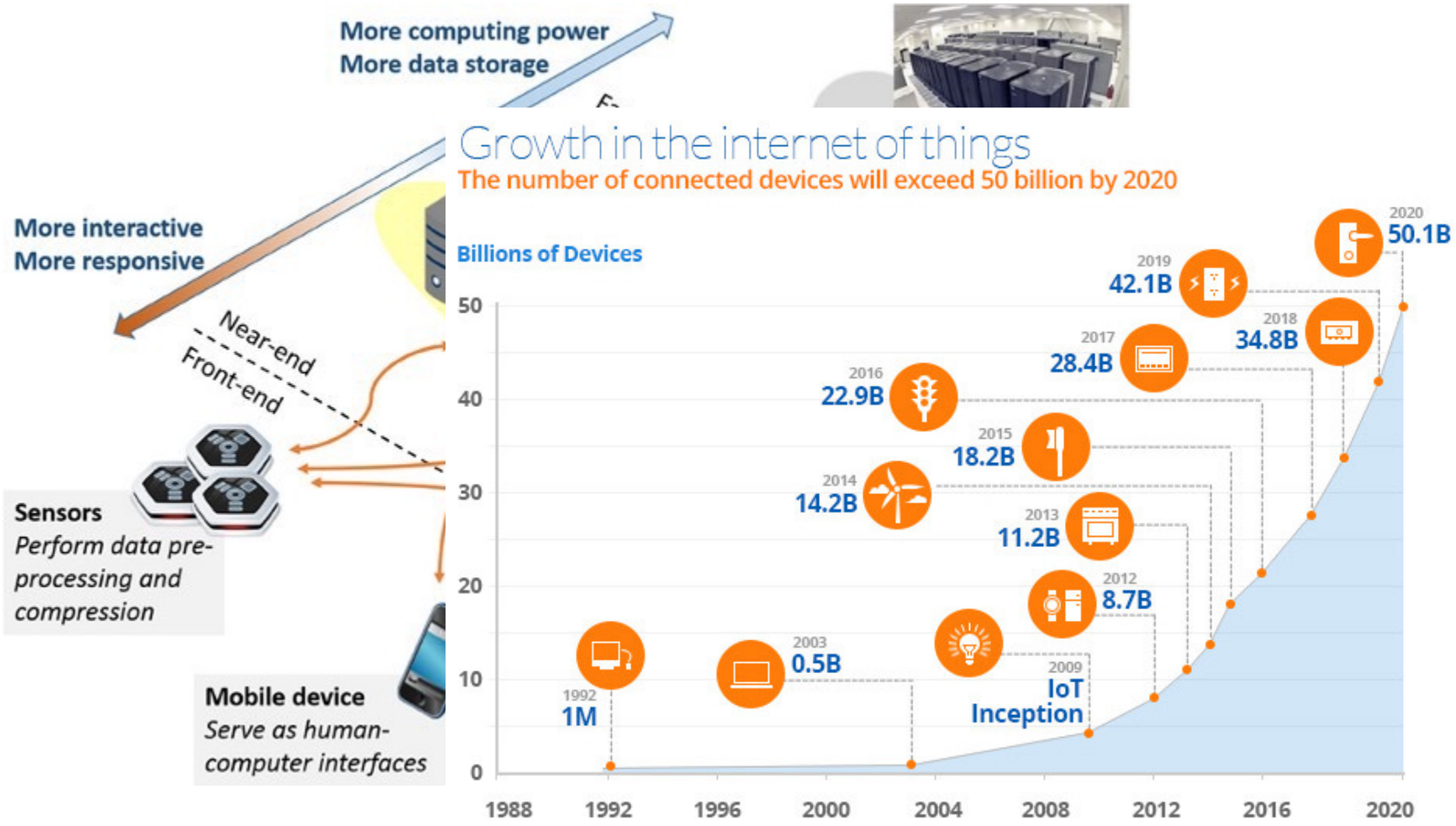
John Hennessy Jul'18

https://www.nextbigfuture.com/2019/02/the-end-of-moores-law-in-detail-and-starting-a-new-golden-age.html

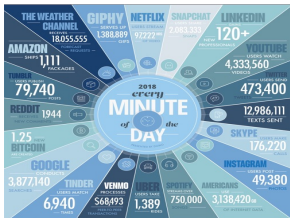
Processing at the Edge



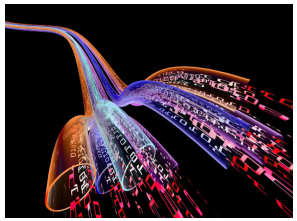
Processing at the Edge



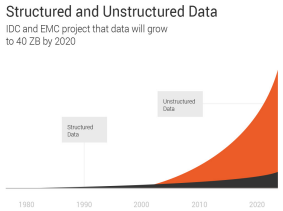
https://www.iotonlinestore.com/



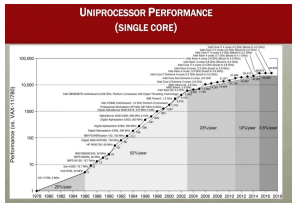
Big Data



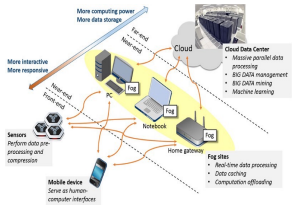
Real Time Requirements



Unstructured Data



No Moore's Law



Edge

DATA-INTENSIVE DISTRIBUTED APPLICATIONS

SCALABLE

LOW LATENCY

EVENT BASED

DISTRIBUTED

HETEROGENEOUS

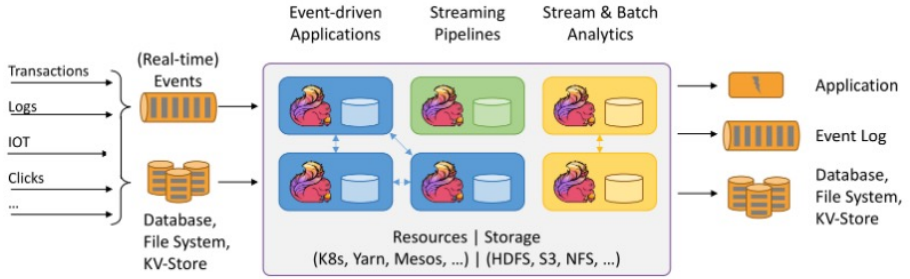
SOFTWARE DESIGN

Flink

- What is Apache Flink?
- Use Cases
- Powered By
- FAQ
- Downloads
- Tutorials
- Documentation
- Getting Help
- Flink Blog
- Community & Project Info
- Roadmap
- How to Contribute
- Flink on GitHub
- 中文版
- @ApacheFlink
- Plan Visualizer

Apache Software Foundation
License Security
Donate Thanks

Apache Flink® — Stateful Computations over Data Streams



All streaming use cases

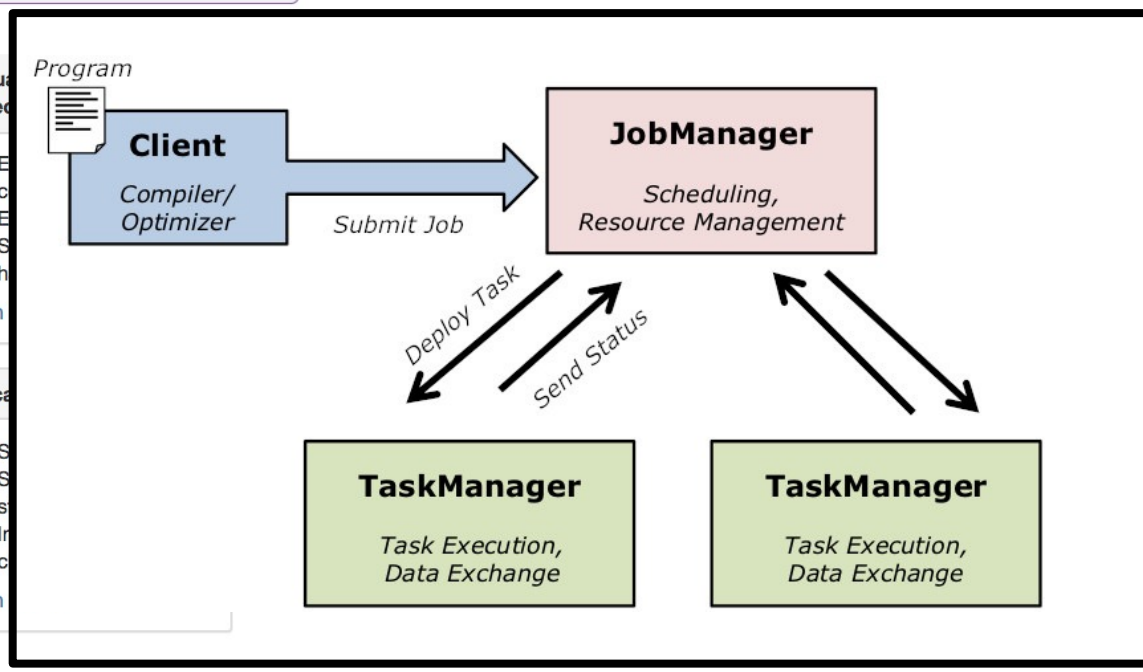
- Event-driven Applications
- Stream & Batch Analytics
- Data Pipelines & ETL

Learn more

Operational Focus

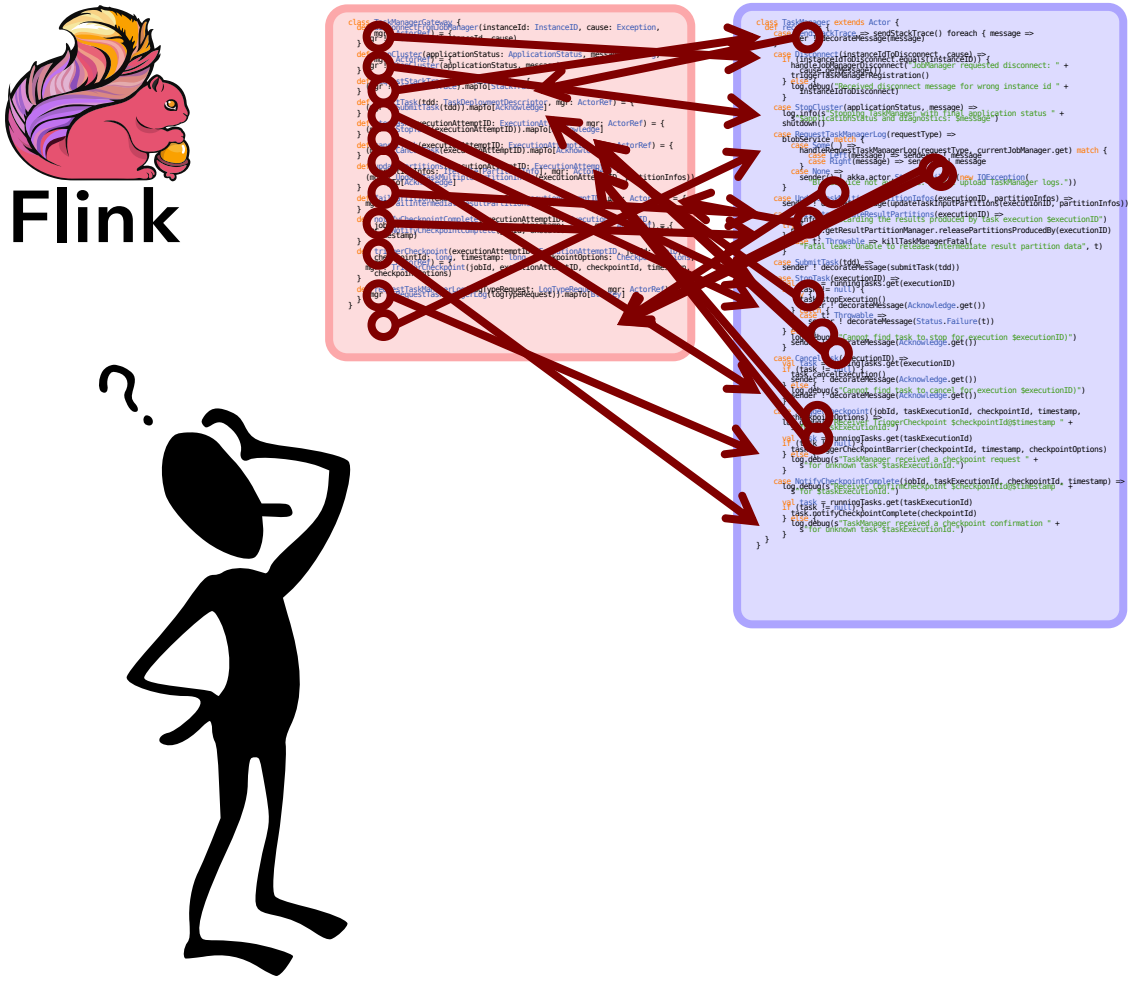
- Flexible deployment
- High-availability setup
- Savepoints

Learn more

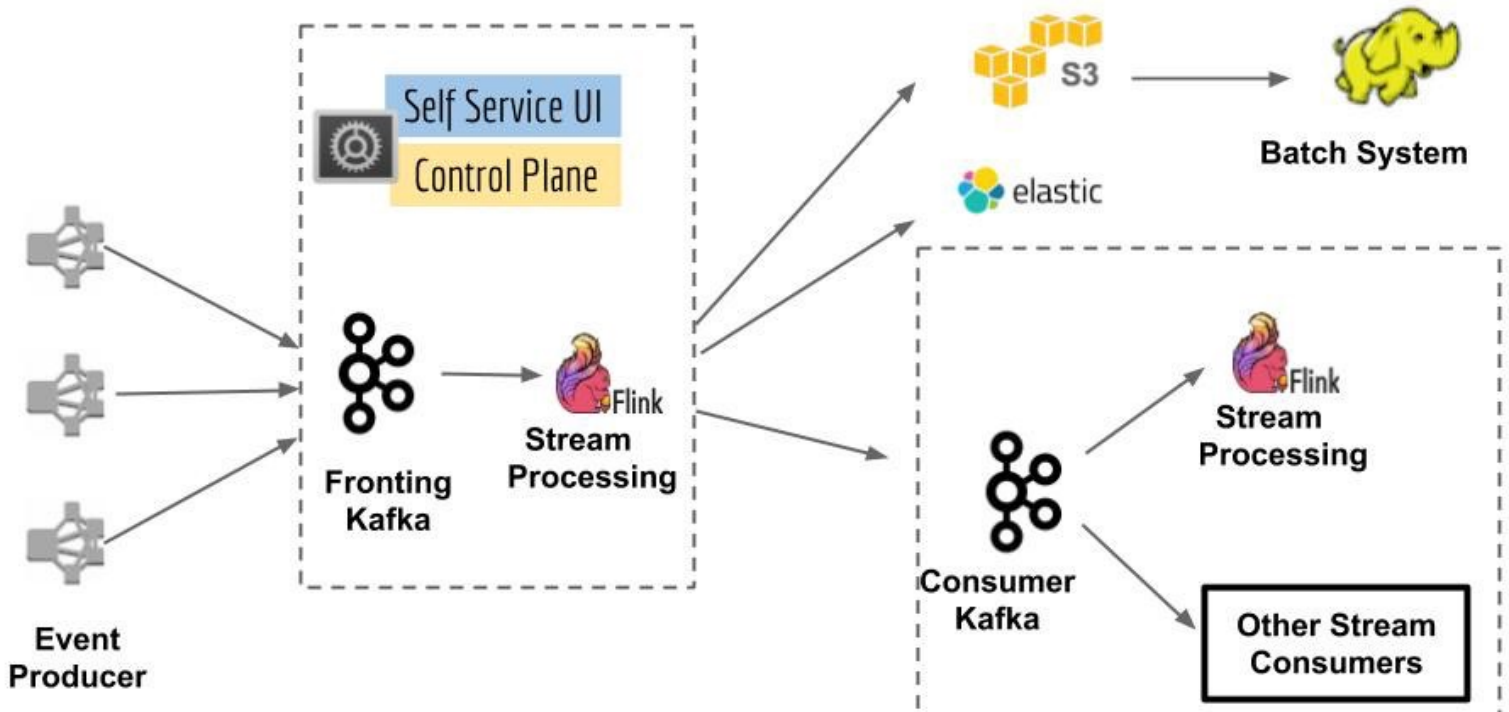


Powered by Flink

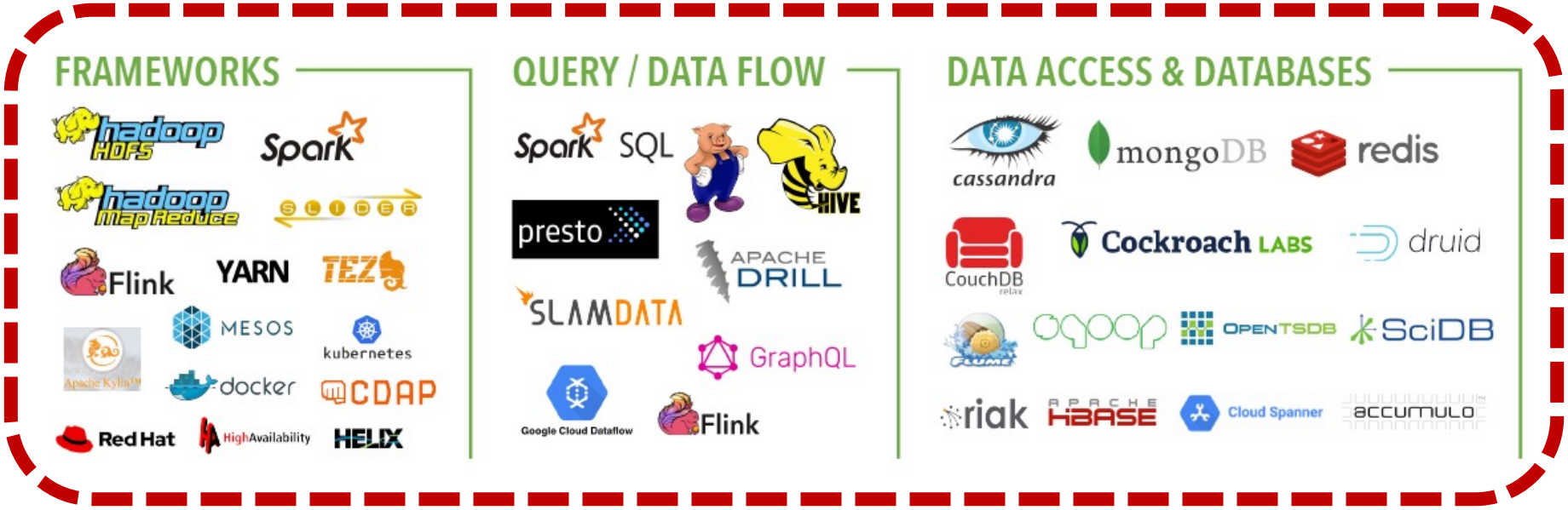
yelp zalando Alibaba.com aws


Flink

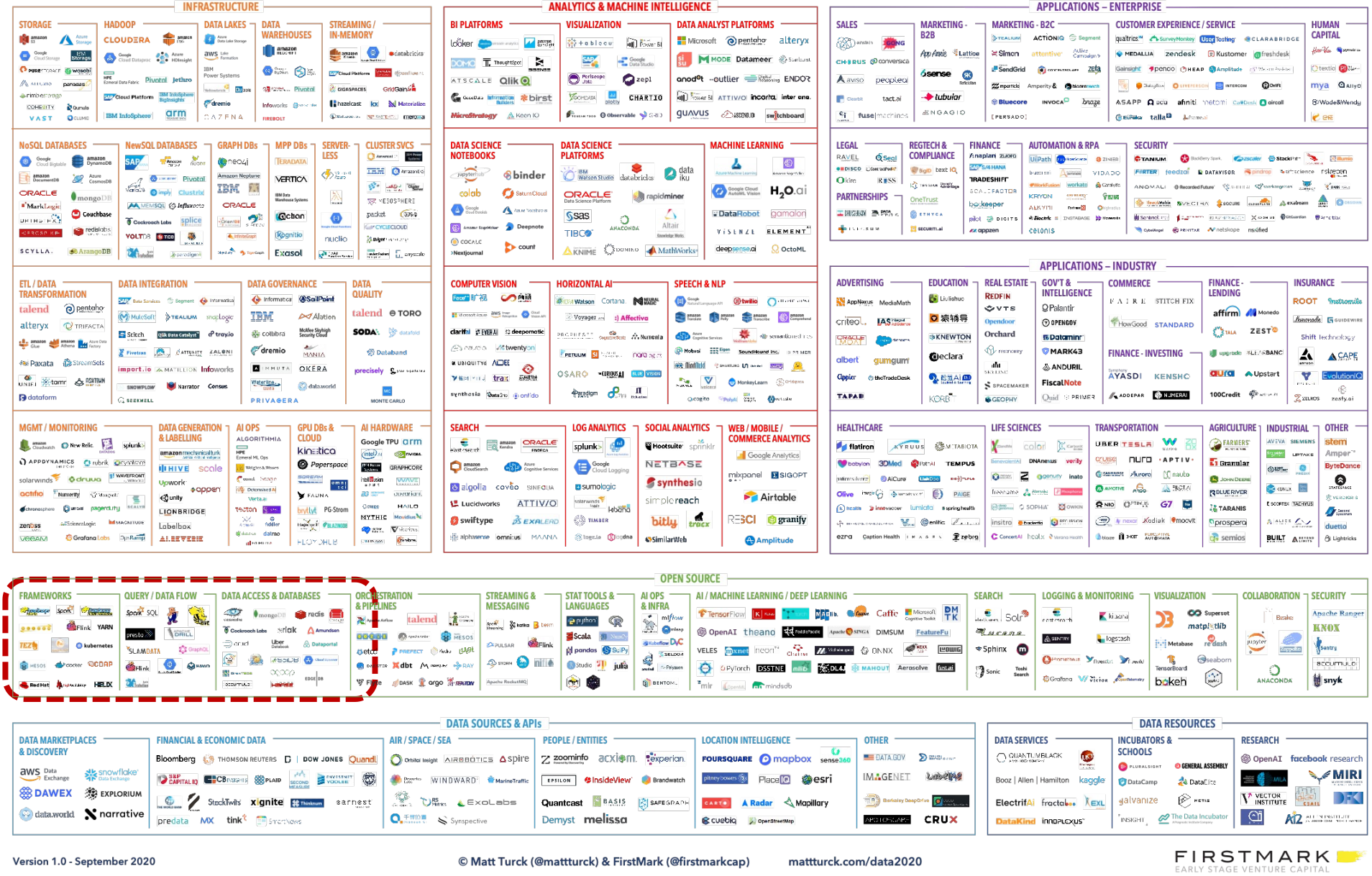
Inca: Netflix's real time message tracing system



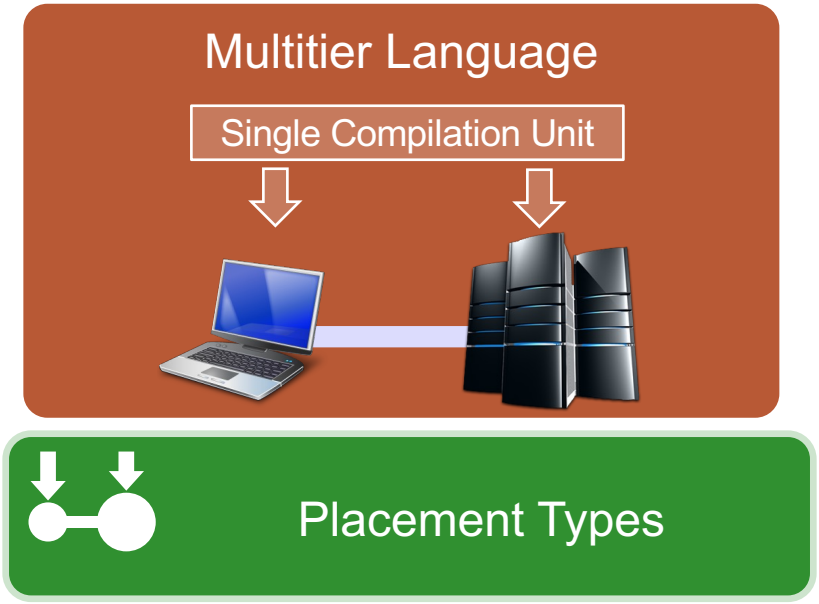
https://medium.com/@NetflixTechBlog/inca-message-tracing-and-loss-detection-for-streaming-data-netflix-de4836fc38c9


FRAMEWORKS
hadoop HDFS
Spark
hadoop MapReduce
SLIDER
Flink
YARN
TEZ
MESOS
kubernetes
Apache Kylin
docker
CDAP
Red Hat
HighAvailability
HELIX
QUERY / DATA FLOW
Spark SQL
HIVE
presto
APACHE DRILL
SLAMDATA
GraphQL
Google Cloud Dataflow
Flink
DATA ACCESS & DATABASES
cassandra
mongoDB
redis
CouchDB
Cockroach LABS
druid
OPENTSDB
SciDB
riak
APACHE HBASE
Cloud Spanner
accumulo

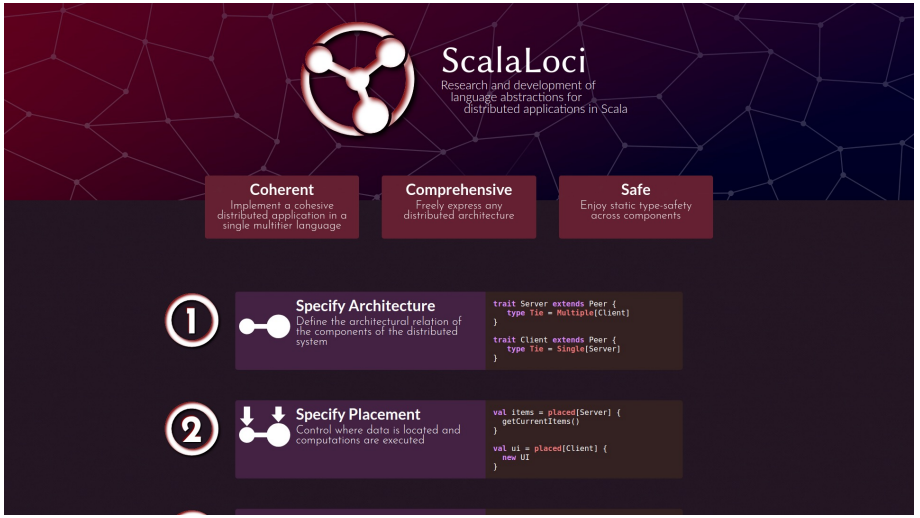
DATA & AI LANDSCAPE 2020



ScalaLoci Programming Framework



www.scala-loci.github.io



[P.Weisenbureger, M.Koehler, G.Salvaneschi, **Distributed System Development with ScalaLoci,** OOPSLA'18]

[P.Weisenbureger, G.Salvaneschi, **Multitier Modules,** ECOOP'19]

Placement Types

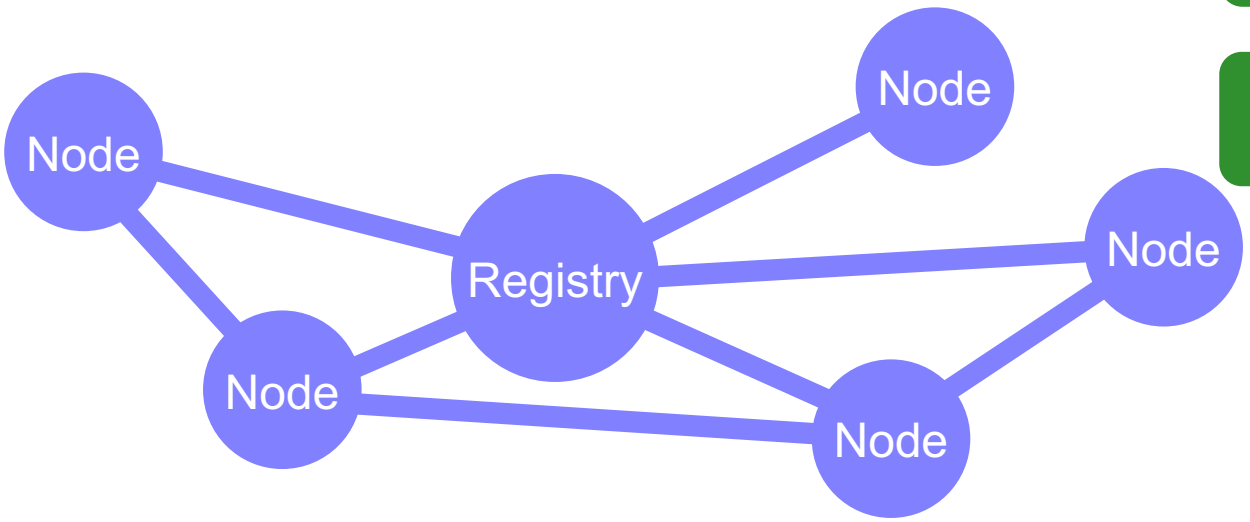
```
trait Registry extends Peer
trait Node extends Peer

val message: Event[String] on Registry
```

Peers

Placement Types

Architecture

```
trait Registry extends Peer { type Tie = Multiple[Node] }
trait Node extends Peer { type Tie = Single[Registry] with Multiple[Node] }
```

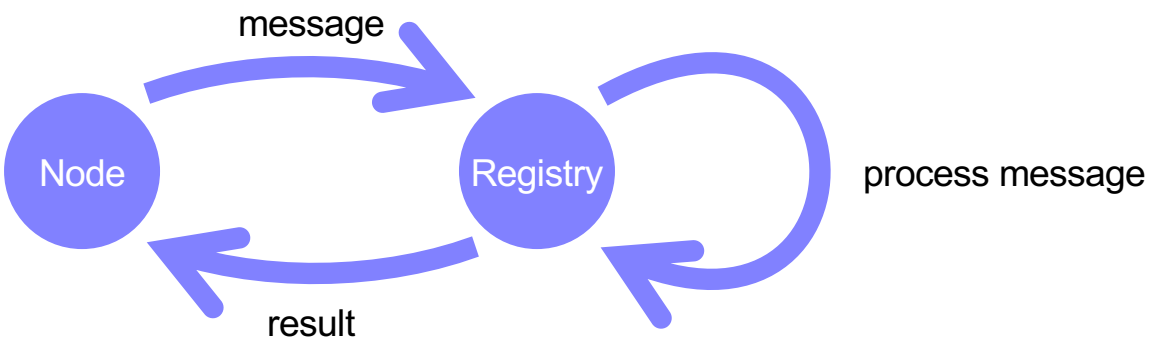


Architecture Specification through Peer Types

Architecture-Based Remote Access

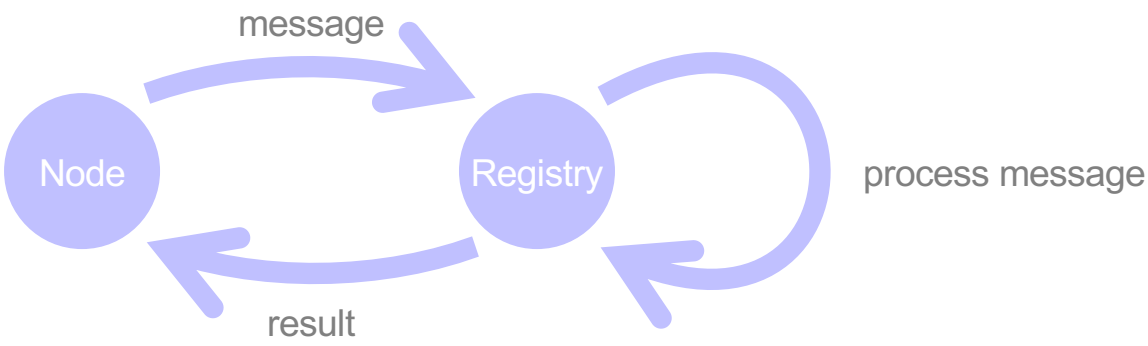
Data Flow

```
val message = Event[String]()
val result = message map processMessage
val ui = new UI(result)
```



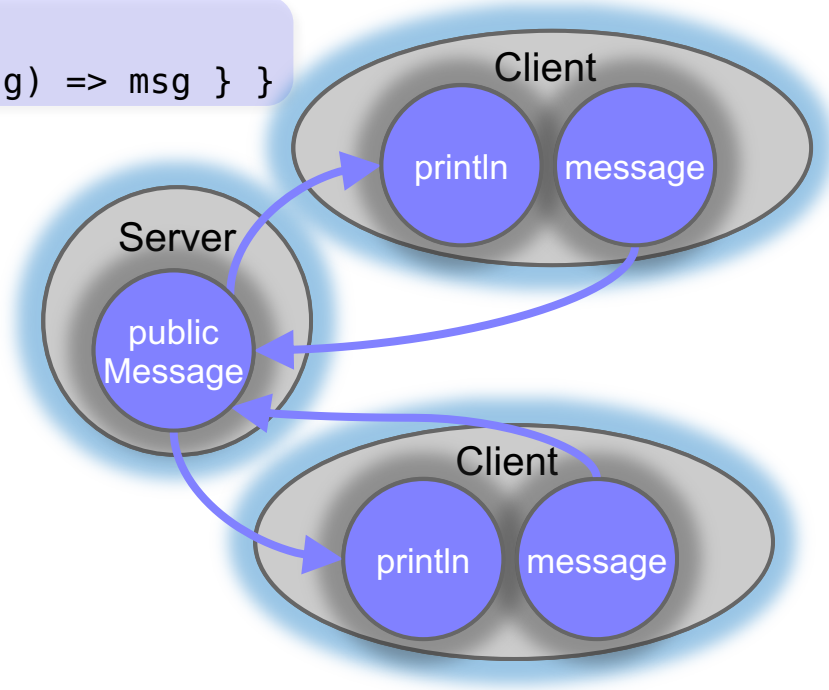
Distributed Data Flow

```
val message: Event[String] on Node = placed[Node] { Event[String]() }
val result = placed[Registry] { message.asLocal map processMessage }
val ui = placed[Node] { new UI(result.asLocal) }
```



Complete Distributed Chat

```
@multitier object Chat {
  trait Server extends Peer { type Tie = Multiple[Client] }
  trait Client extends Peer { type Tie = Single[Server] }

  val message = placed[Client] { Evt[String] }

  val publicMessage = placed[Server] {
    message.asLocalFromAllSeq map { case (_, msg) => msg } }

  placed[Client].main {
    publicMessage.asLocal observe println
    for (line <- io.Source.stdin.getLines)
      message fire line } }
```



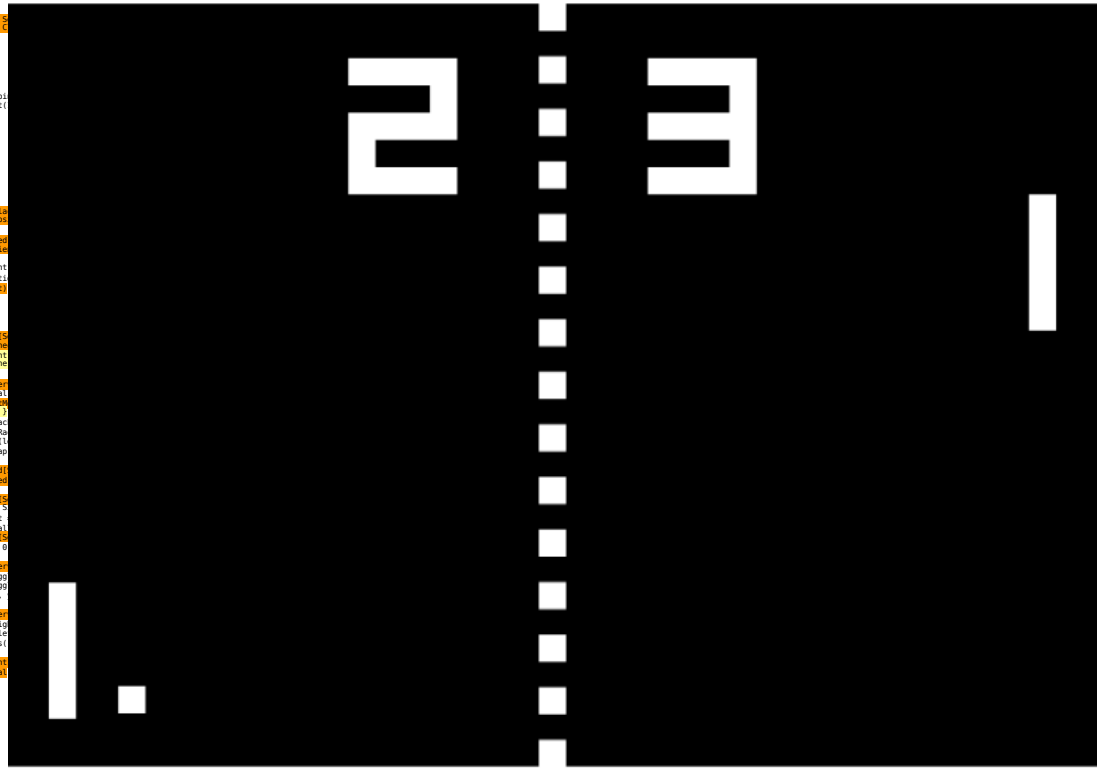
Porting to Distribution

Local

```
val ballSize = 20
val maxX = 800
val maxY = 400
val leftPos = 30
val rightPos = 770
val initPosition = Point(400, 200)
val initSpeed = Point(10, 8)

val ball: Signal[Point] = tick.fold(initPosition) {
  (ball, _) => ball + speed.get }

val areas = {
  val racketY = Seq(
    Signal { UI.mousePosition().y },
    Signal { ball().y })
  val leftRacket = Racket(leftRacketPos, racketY(0))
  val rightRacket = Racket(rightRacketPos, racketY(1))
  val rackets = List(leftRacket, rightRacket)
  Signal { rackets map { _.area() } } }

val leftWall = ball.changed && { _.x < 0 }
val rightWall = ball.changed && { _.x > maxX }

val xBounce = {
  val ballInRacket = Signal { areas() exists { _ contains ball() } }
  val collisionRacket = ballInRacket changedTo true
  leftWall || rightWall || collisionRacket }
val yBounce = ball.changed &&
  { ball => ball.y < 0 || ball.y > maxY }

val speed = {
  val x = xBounce toggle (initSpeed.x, -initSpeed.x)
  val y = yBounce toggle (initSpeed.y, -initSpeed.y)
  Signal { Point(x(), y()) } }

val score = {
  val leftPoints = rightWall.iterate(0) { _ + 1 }
  val rightPoints = leftWall.iterate(0) { _ + 1 }
  Signal { leftPoints() + " : " + rightPoints() } }

val ui = new UI(areas, ball, score)
```

ScalaLoci

```
trait Server extends S
trait Client extends C

val ballSize = 20
val maxX = 800
val maxY = 400
val leftPos = 30
val rightPos = 770
val initPosition = Poi
val initSpeed = Point(

val clientMouseY = pla
  Signal { UI.mousePos

val isPlaying = placed
  Signal { remote[Clie

val ball: Signal[Point
  tick.fold(initPositi
    if (isPlaying.get)

val players = placed[S
  remote[Client].conne
    case left :: right
    case _ => Seq(None

val areas = placed[Ser
  val racketY = Signal
    client => (clientM
      initPosition.y ) }
  val leftRacket = Rac
  val rightRacket = Ra
  val rackets = List(l
  Signal { rackets map

val leftWall = placed[
val rightWall = placed

val xBounce = placed[S
  val ballInRacket = S
  val collisionRacket
  leftWall || rightWal
val yBounce = placed[S
  { ball => ball.y < 0

val speed = placed[Ser
  val x = xBounce togg
  val y = yBounce togg
  Signal { Point(x(),

val score = placed[Ser
  val leftPoints = rig
  val rightPoints = le
  Signal { leftPoints(

val ui = placed[Client
  new UI(areas.asLocal
```



```
    case UpdateScore(score) => this.score set score }

  server ! AddPlayer
}
```

Akka

```
    clients foreach { _ updateScore score }
}

@remote trait Client {
  def updateAreas(areas: List[Area]): Unit
  def updateBall(ball: Point): Unit
  def updateScore(score: String): Unit }

class ClientImpl(server: Server) extends Client {
  val self = makeStub[Client](this)

  val areas = Var(List.empty[Area])
  val ball = Var(Point(0, 0))
  val score = Var("0 : 0")

  UI.mousePosition observe { pos =>
    server mouseYChanged (self, pos.y) }

  val ui = new UI(areas, ball, score)

  def updateAreas(areas: List[Area]) = synchronized { this.areas() = areas }
  def updateBall(ball: Point) = synchronized { this.ball() = ball }
  def updateScore(score: String) = synchronized { this.score() = score }

  server addPlayer self
}
```

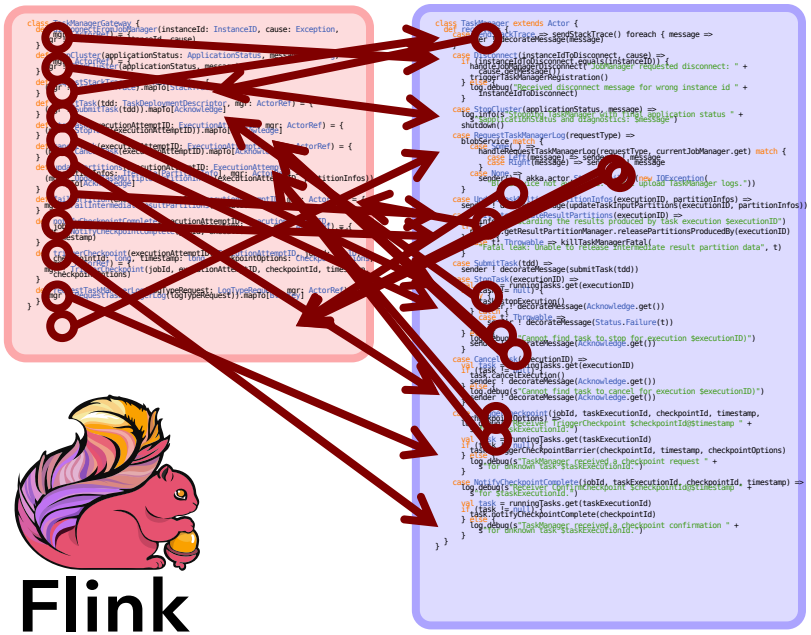
RMI

multi-user support

distribution

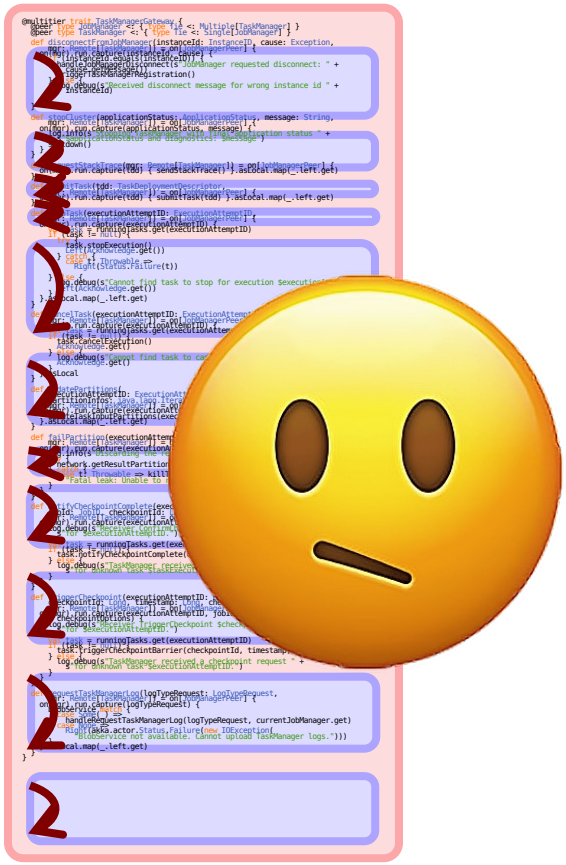

Flink

Crosscutting functionality separated among compilation units



Flink

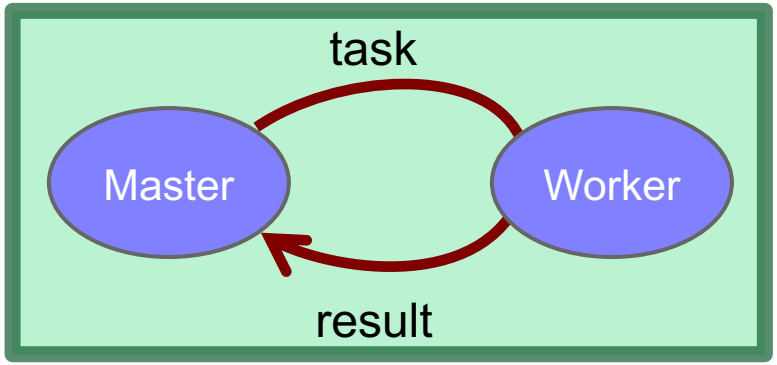
Developers are **not** forced to modularize **along network boundaries**



Modelling Distributed Software: Multitier Modules

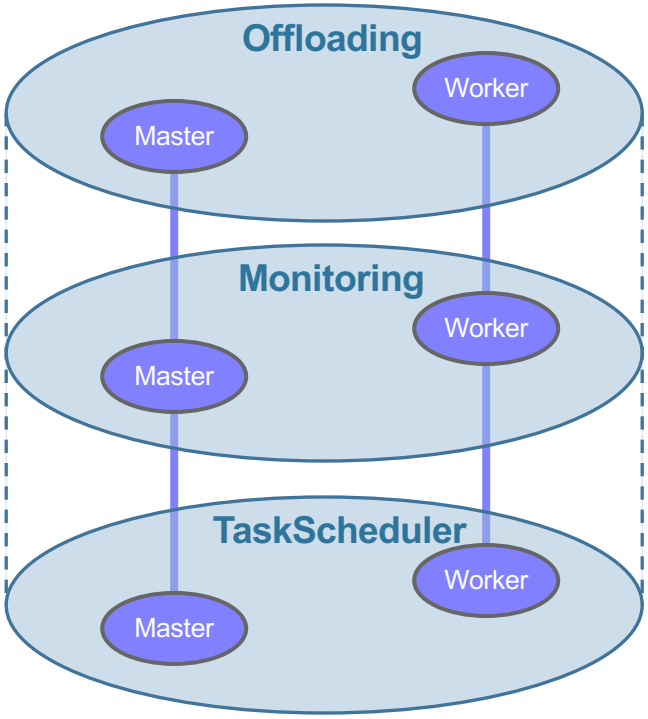
Distributed functionality = Module

Composing modules = Composing subsystems



MasterWorker<M,W>

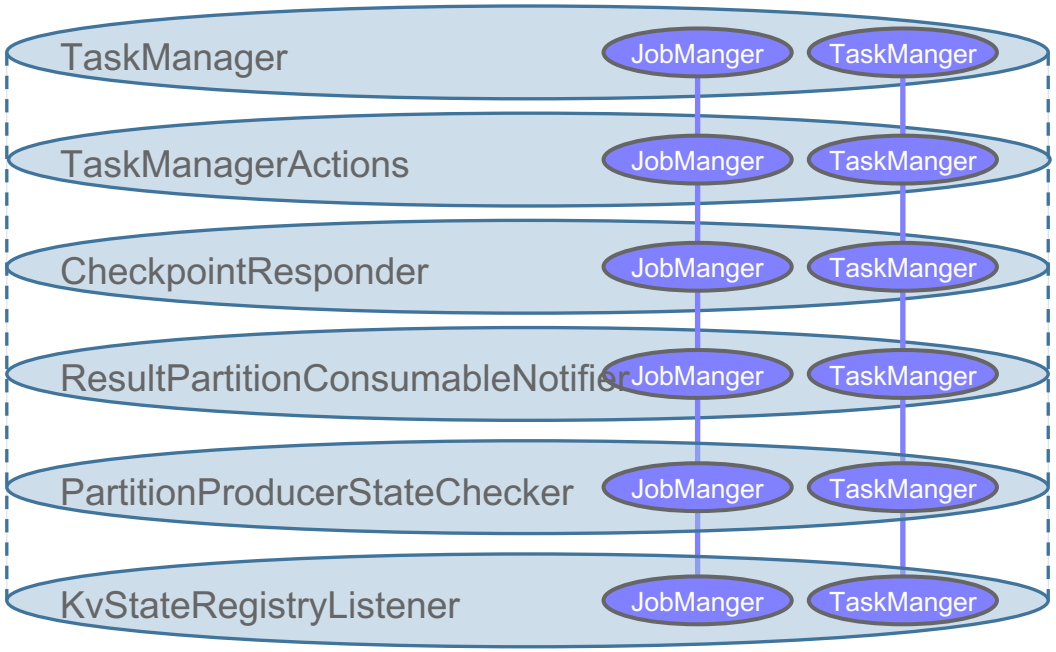
Stacking Multitier Modules




```
@multitier trait Offloading[T] {
  @peer type Master <: { type Tie <: Multiple[Worker] }
  @peer type Worker <: { type Tie <: Single[Master] }
  def run(task: Task[T]): Future[T] on Master =
    placed { (remote(selectWorker()) call execute(task)).asLocal }
  private def execute(task: Task[T]): T on Worker =
    placed { task.process() }
}

@multitier trait Monitoring {
  @peer type Master <: { type Tie <: Multiple[Worker] }
  @peer type Worker <: { type Tie <: Single[Master] }
  def monitoredTimedOut(monitored: Remote[Worker]): Unit on Master
}

@multitier trait TaskScheduler[T] extends
  Offloading[T] with
  Monitoring
```

Flink Case Study

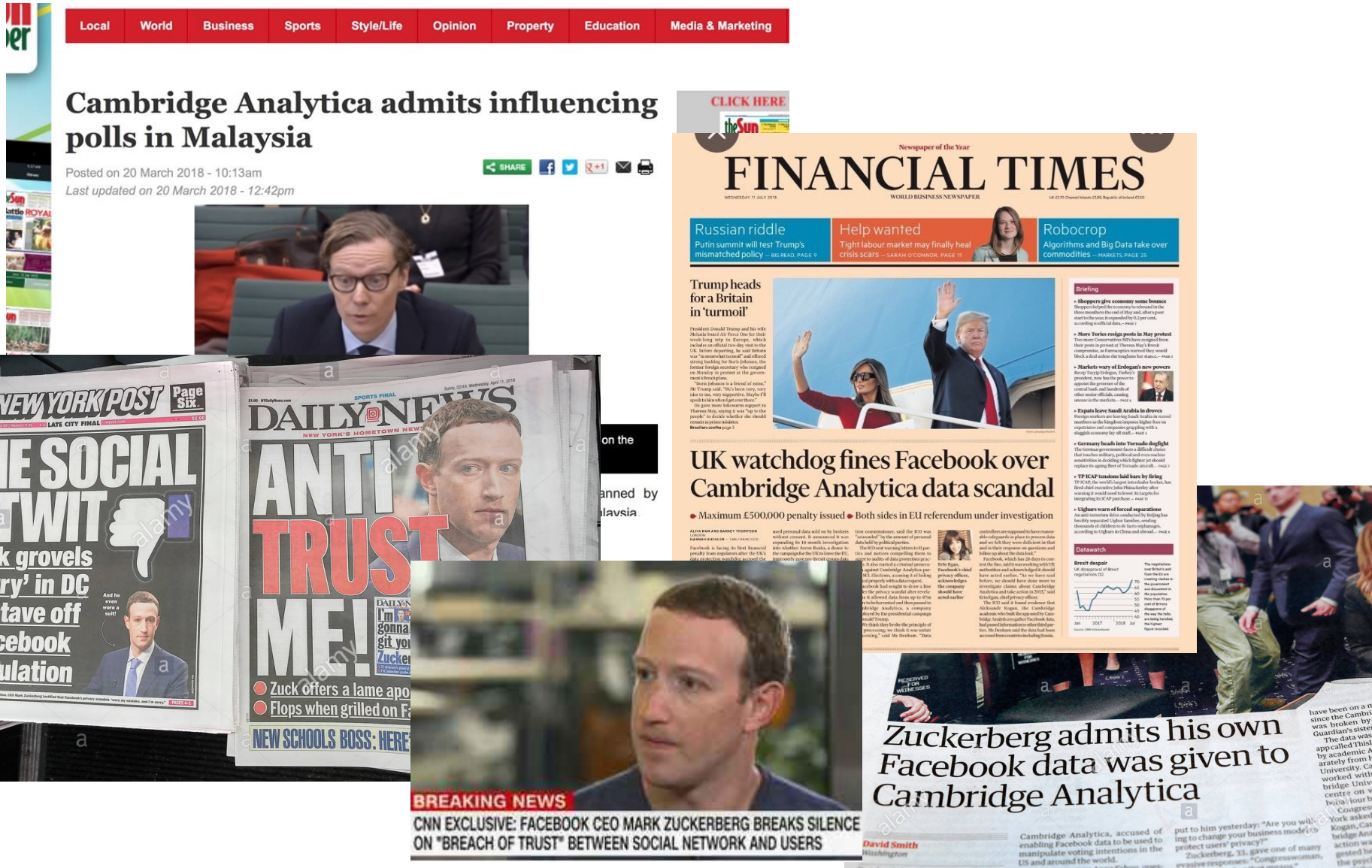
```
@multitier object TaskDistributionSystem extends
  TaskManager with
  TaskManagerActions with
  CheckpointResponder with
  ResultPartitionConsumableNotifier with
  PartitionProducerSt
  KvStateRegistryList
```




Flink

PRIVACY

The Case of Cambridge Analytica

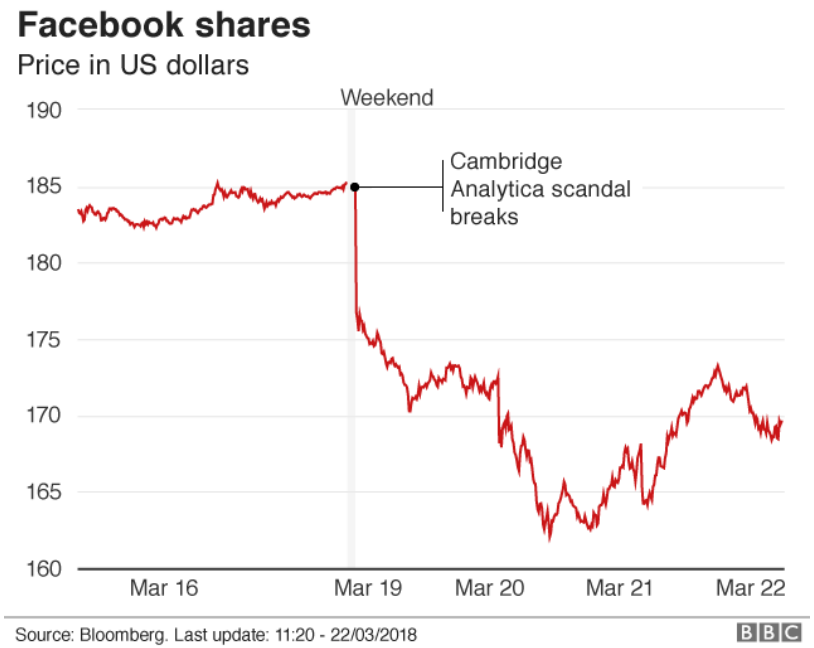


The Case of Cambridge Analytica

87 Million users affected (FB estimation)

100 billion loss, FB market cap.

Zuckerberg testified in front of the US Congress.



Turns out Cambridge Analytica assisted with Trump and Brexit campaign

Lesson Learned

We need to engineer software

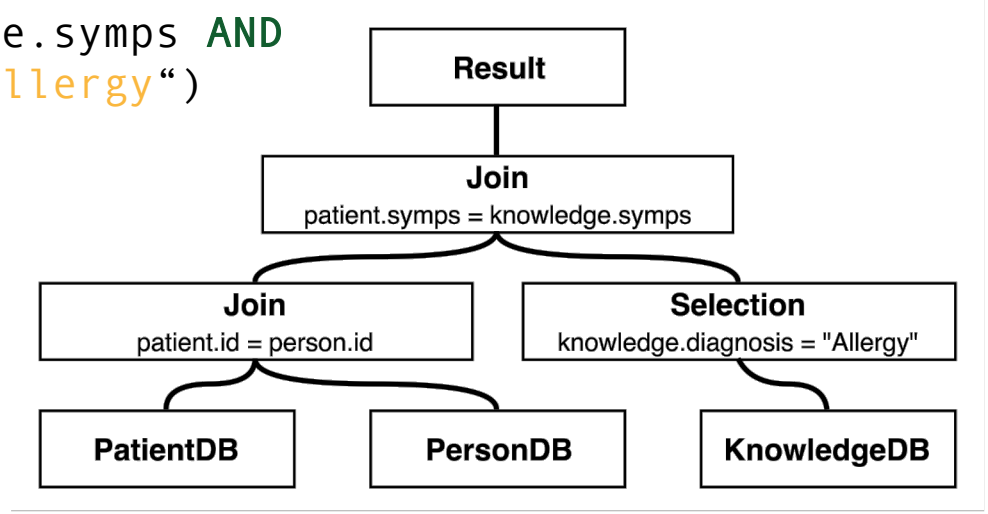
to ***precisely*** control

where (private) data go!

https://newsela.com/read/lib-what-is-river-delta/id/34818/quiz/0/

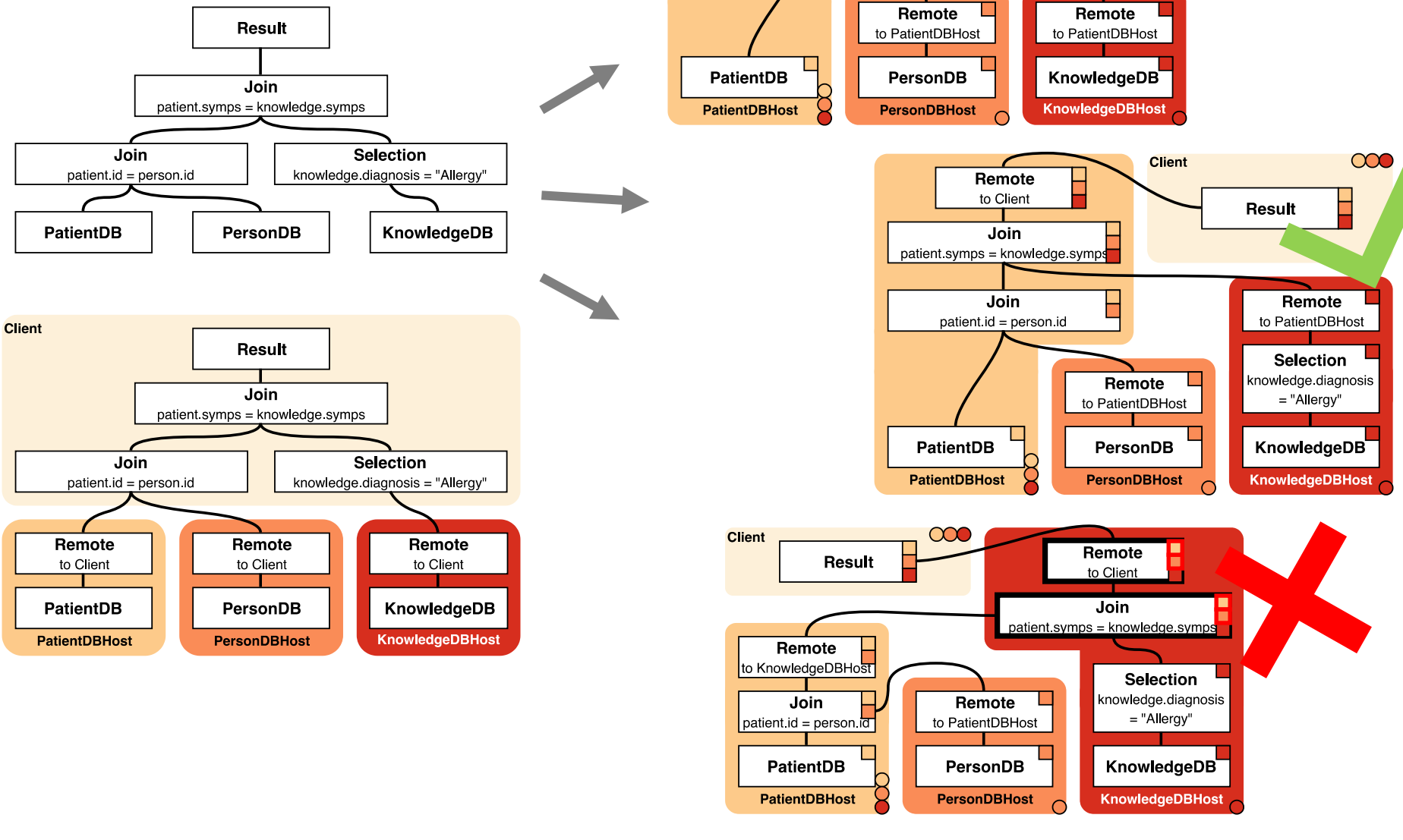
SecQL: Privacy Preserving Queries

```
val result = SELECT (*)
    FROM (personDB,patientDB,knowledgeDB)
    WHERE ((person,patient,knowledge) =>
        person.id == patient.id AND
        patient.symps == knowledge.symps AND
        knowledge.diagnosis == “Allergy“)
```



[G.Salvaneschi, M.Köhler, D.Sokolowski, P.Haller, S.Erdweg, M.Mezini, **Language-Integrated Privacy-Aware Distributed Queries**, OOPSLA’19]

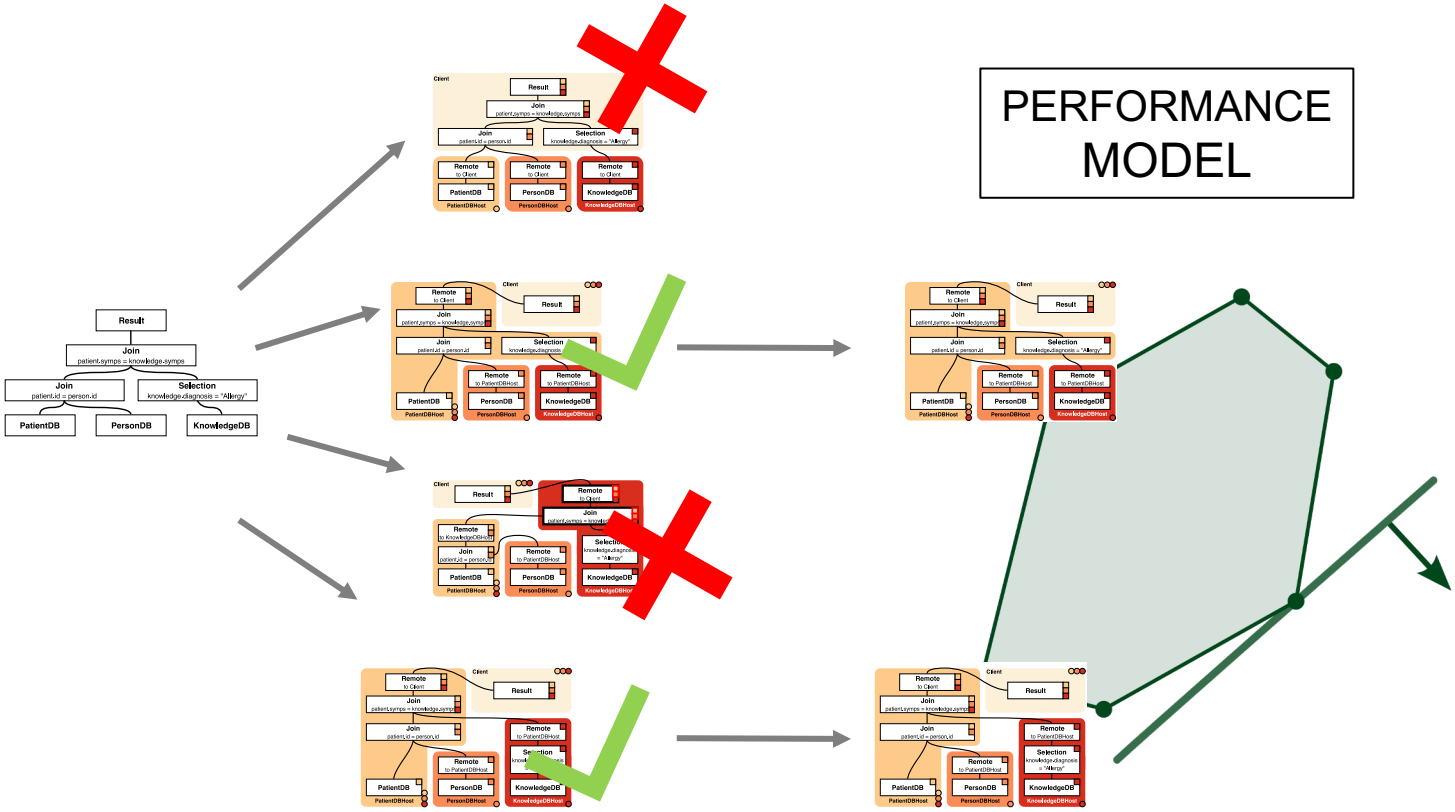
[Ralf Mitschke, Sebastian Erdweg, Mira Mezini, Mirko Kohler, Guido Salvaneschi, **i3QL: Language-Integrated Live Data Views,** OOPSLA’14.]

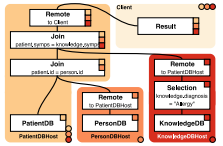

Result
Join
patient.symps = knowledge.symps
Join
patient.id = person.id
Selection
knowledge.diagnosis = "Allergy"
PatientDB
PersonDB
KnowledgeDB
Client
Result
Join
patient.symps = knowledge.symps
Join
patient.id = person.id
Selection
knowledge.diagnosis = "Allergy"
Remote
to Client
PatientDB
PatientDBHost
Remote
to Client
PersonDB
PersonDBHost
Remote
to Client
KnowledgeDB
KnowledgeDBHost
Client
Result
Remote
to Client
Join
patient.symps = knowledge.symps
Join
patient.id = person.id
Remote
to Client
PatientDB
PatientDBHost
Remote
to PatientDBHost
PersonDB
PersonDBHost
Remote
to PatientDBHost
Selection
knowledge.diagnosis
= "Allergy"
KnowledgeDB
KnowledgeDBHost
Client
Result
Remote
to Client
Join
patient.symps = knowledge.symps
Remote
to KnowledgeDBHost
Join
patient.id = person.id
PatientDB
PatientDBHost
Remote
to PatientDBHost
PersonDB
PersonDBHost
Selection
knowledge.diagnosis
= "Allergy"
KnowledgeDB
KnowledgeDBHost

SecQL: 2 Steps Placement Strategy

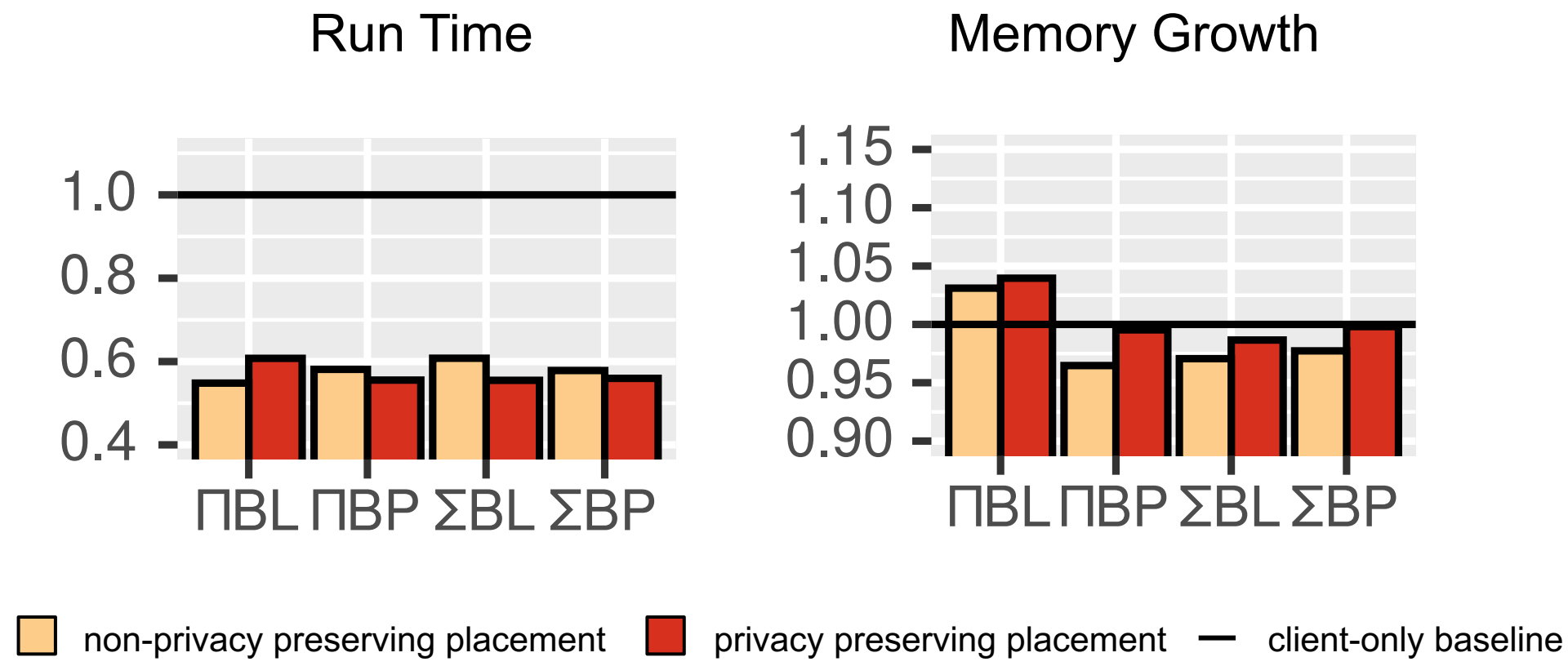
Placement Space Reduction

Cost Model Optimization



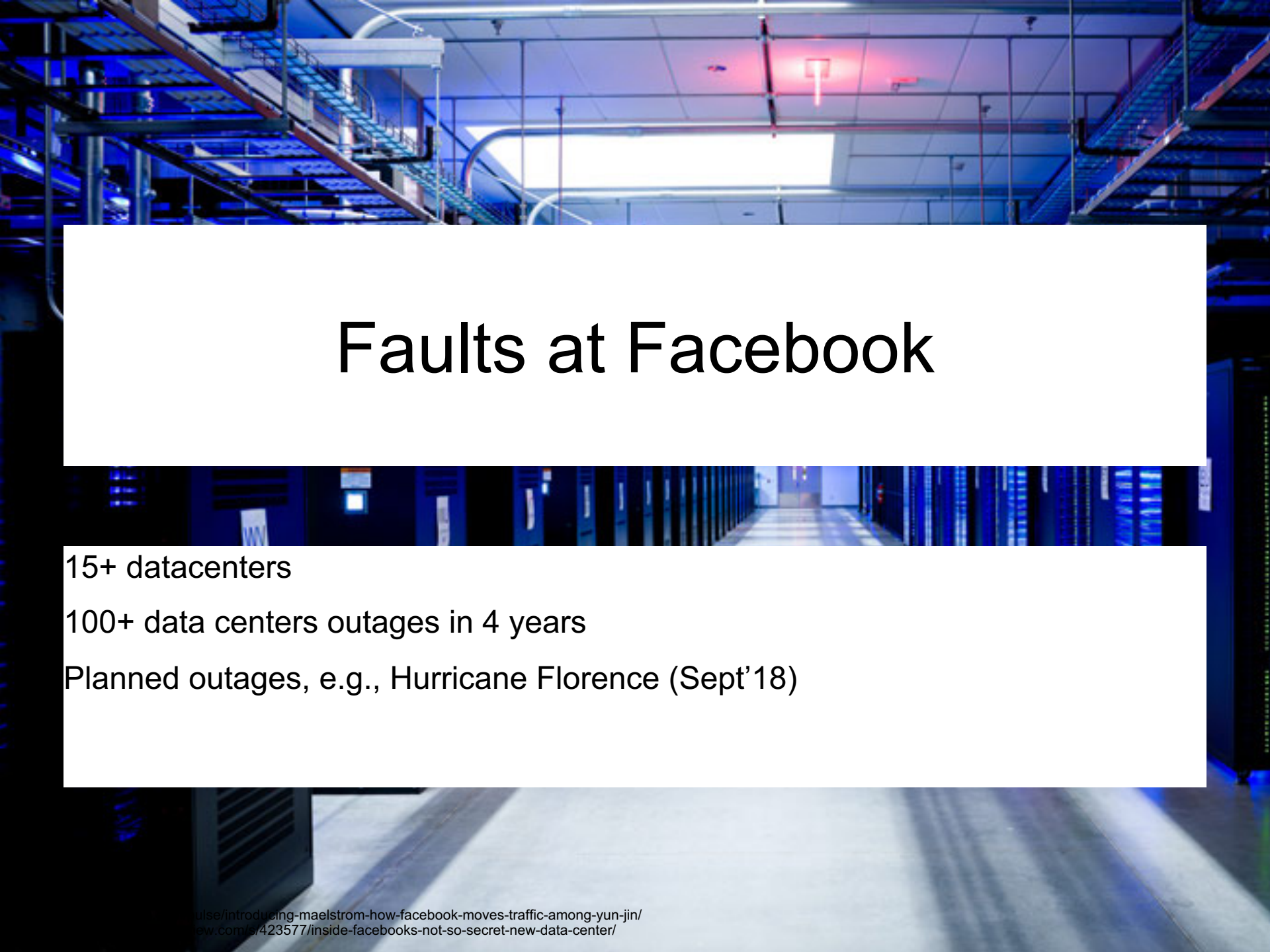


Secure Optimal Plecement



Privacy preserving placement has performance comparable to the optimal placement

FAULT TOLERANCE



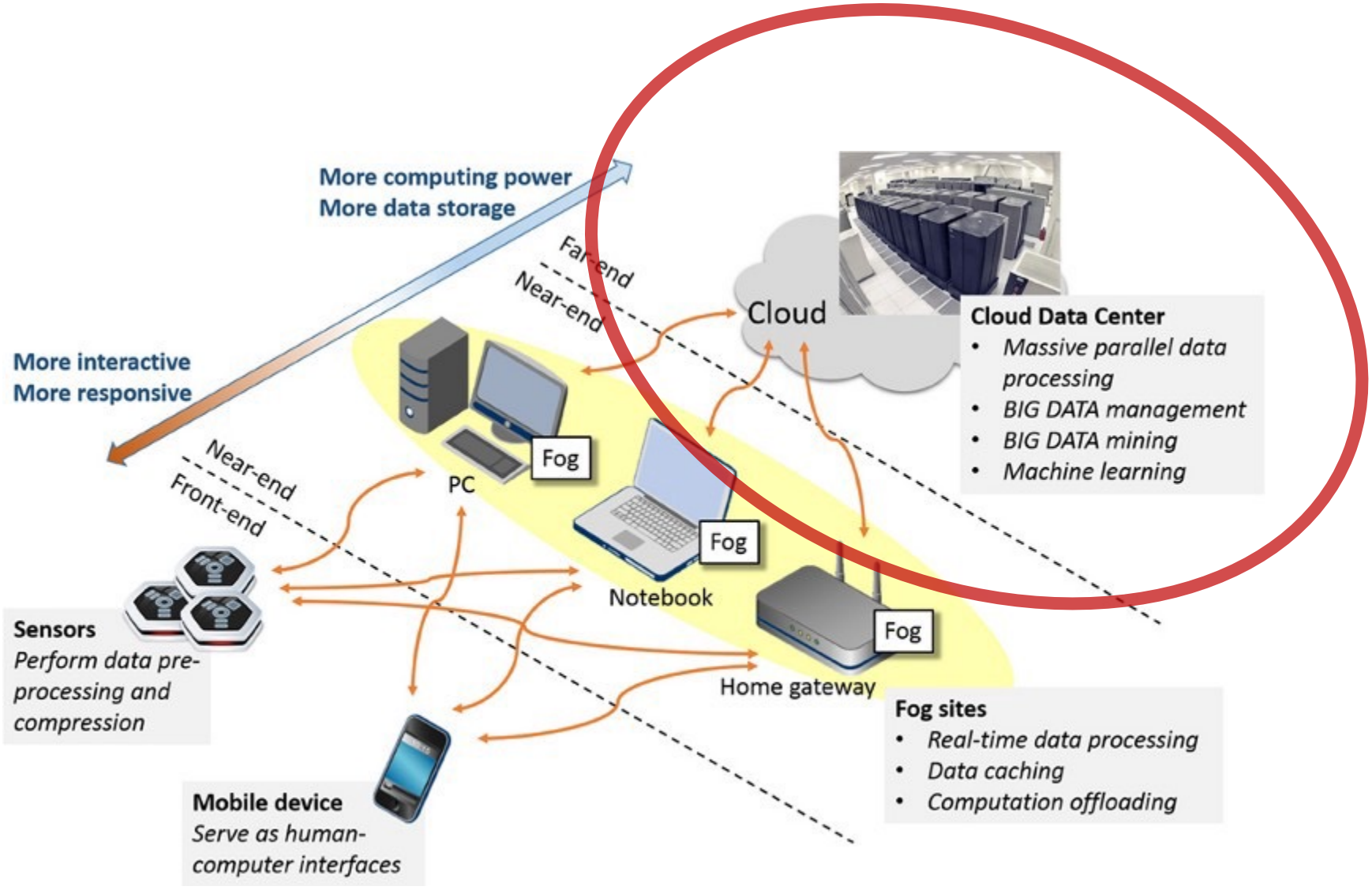
Faults at Facebook

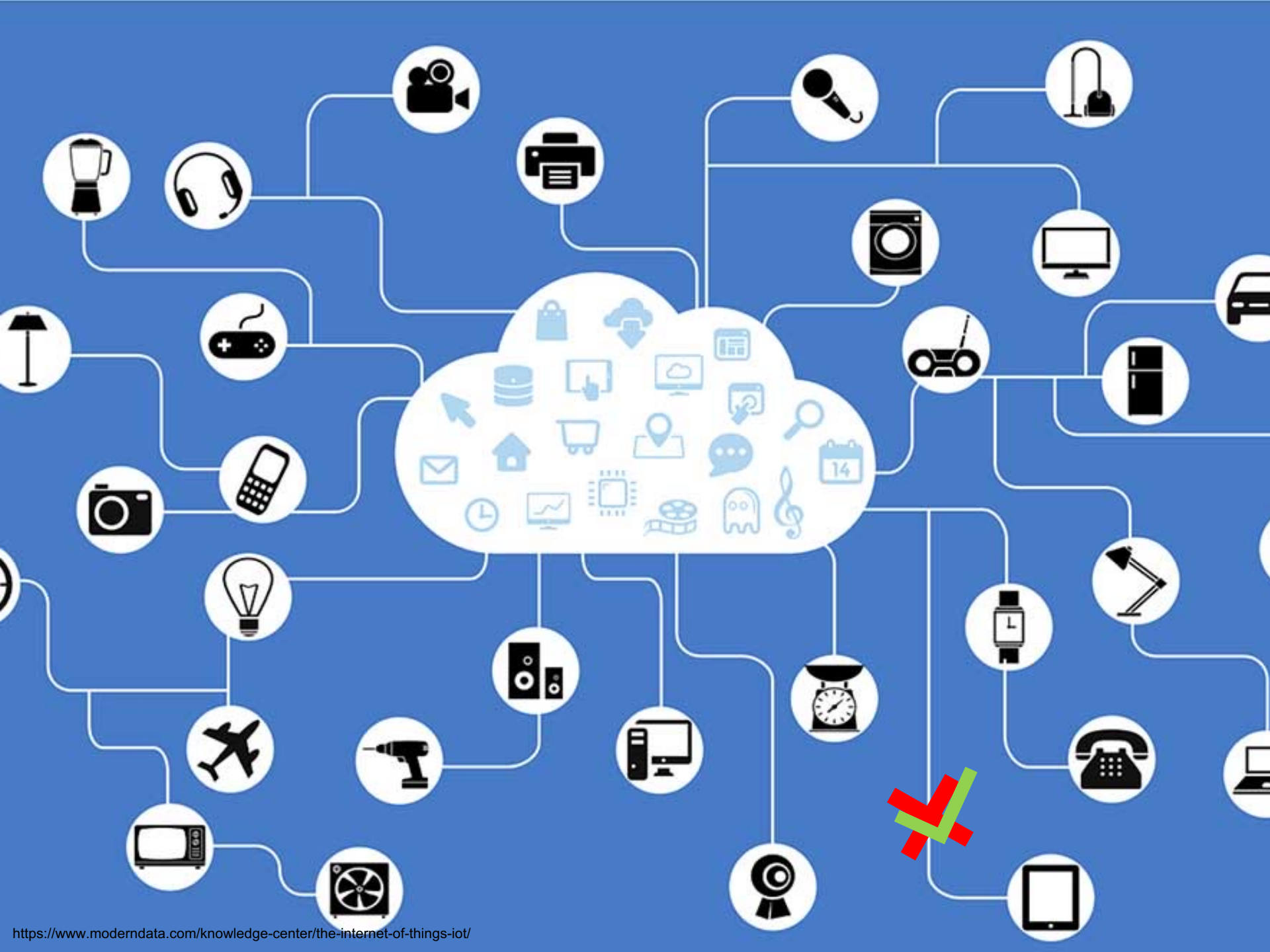
15+ datacenters

100+ data centers outages in 4 years

Planned outages, e.g., Hurricane Florence (Sept'18)

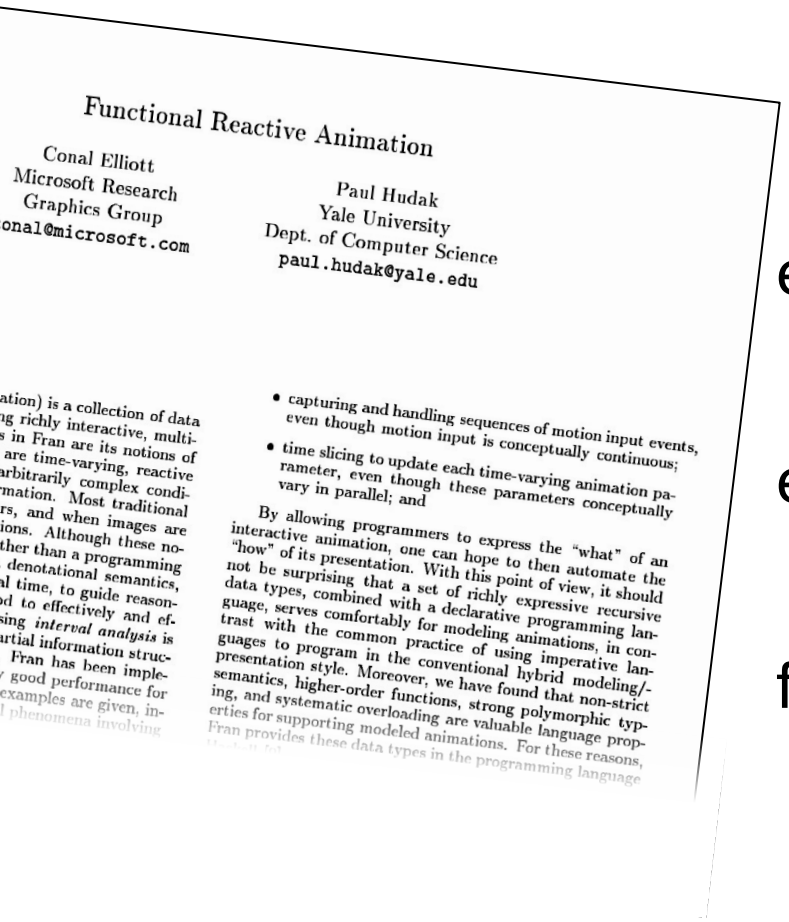
uise/introducing-maelstrom-how-facebook-moves-traffic-among-yun-jin/
ew.com/s/423577/inside-facebooks-not-so-secret-new-data-center/


More computing power
More data storage
More interactive
More responsive
Far-end
Near-end
Near-end
Front-end
Cloud
Cloud Data Center
• Massive parallel data processing
• BIG DATA management
• BIG DATA mining
• Machine learning
Fog
PC
Fog
Notebook
Fog
Home gateway
Sensors
Perform data pre-processing and compression
Mobile device
Serve as human-computer interfaces
Fog sites
• Real-time data processing
• Data caching
• Computation offloading



https://www.moderndata.com/knowledge-center/the-internet-of-things-iot/

Programming for Failures in Edge Computing



Functional Reactive Animation

Conal Elliott
Microsoft Research
Graphics Group
onal@microsoft.com

Paul Hudak
Yale University
Dept. of Computer Science
paul.hudak@yale.edu

ation) is a collection of data
g richly interactive, multi-
s in Fran are its notions of
are time-varying, reactive
arbitrarily complex condi-
mation. Most traditional
rs, and when images are
ions. Although these no-
ther than a programming
denotational semantics,
l time, to guide reason-
d to effectively and ef-
sing *interval analysis* is
artial information struc-
Fran has been imple-
good performance for
examples are given, in-
phenomena involving

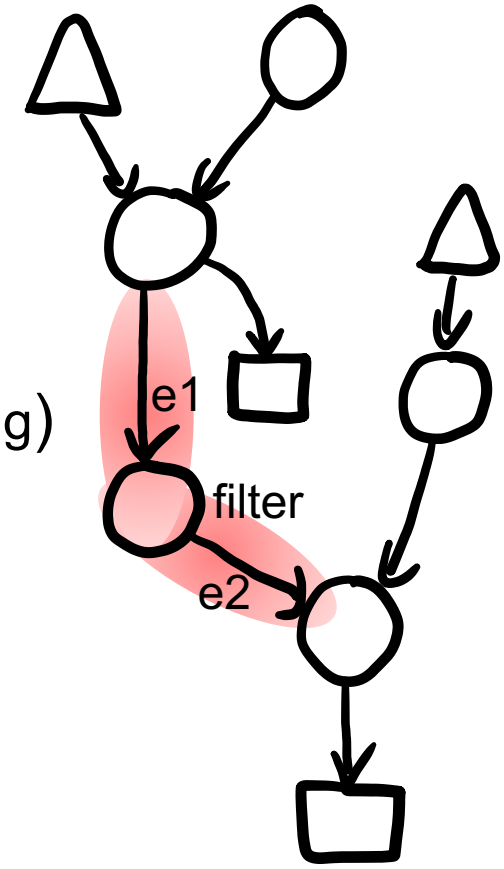
- capturing and handling sequences of motion input events, even though motion input is conceptually continuous;
- time slicing to update each time-varying animation parameter, even though these parameters conceptually vary in parallel; and

By allowing programmers to express the "what" of an interactive animation, one can hope to then automate the "how" of its presentation. With this point of view, it should not be surprising that a set of richly expressive recursive data types, combined with a declarative programming language, serves comfortably for modeling animations, in contrast with the common practice of using imperative languages to program in the conventional hybrid modeling/-presentation style. Moreover, we have found that non-strict semantics, higher-order functions, strong polymorphic typing, and systematic overloading are valuable language properties for supporting modeled animations. For these reasons, Fran provides these data types in the programming language

e2 = e1.filter(x >10)

e3 = e2.map(x.toString)

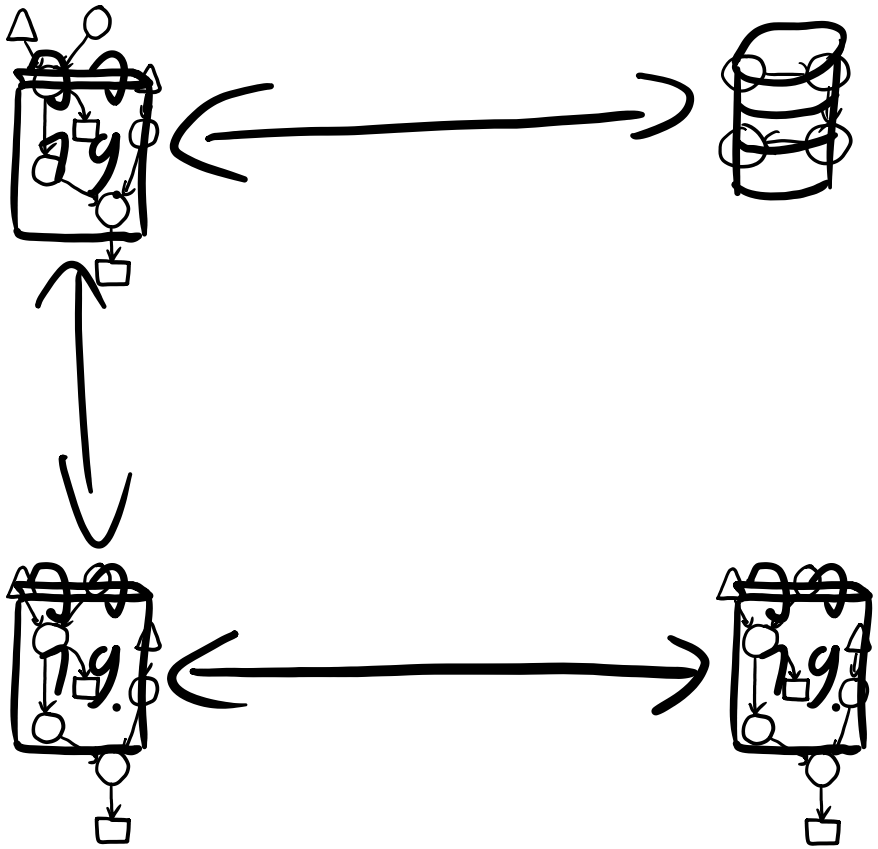
fire e1(10)



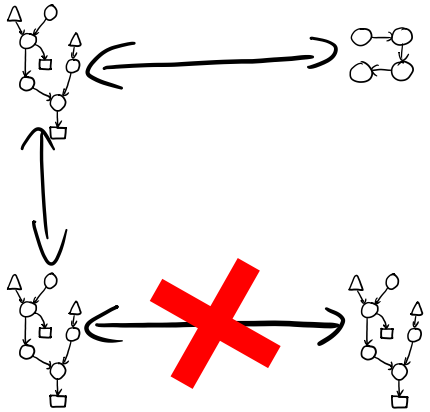
[R.Mogk, J.Drechler, G.Salvaneschi, M.Mezini, **A Fault-tolerant Programming Model for Distributed Interactive Applications**, OOPSLA 2019]

[R.Mogk, L.Baumgaertner, G.Salvaneschi, B.Freisleben, M.Mezini, **Fault-tolerant disttributed reactive Programming,** ECOOP 2018]

Distributed Graphs

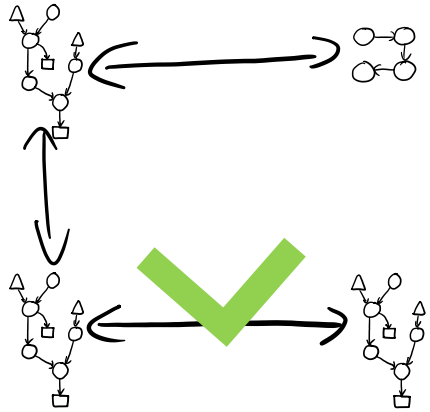


Fault Tolerance „for Free“!



Automatic restore after a crash.

- **Incremental** snapshots of stateful nodes
- Stateless nodes simply ‘**replayed**’



What if your code is not in reactive style?

Automated Refactoring to Reactive Programming

Mirko Köhler
Reactive Programming Technology
Technische Universität Darmstadt
Darmstadt, Germany
koehler@cs.tu-darmstadt.de

Guido Salvaneschi
Reactive Programming Technology
Technische Universität Darmstadt
Darmstadt, Germany
salvaneschi@cs.tu-darmstadt.de

***Abstract*—**
Reactive programming languages and libraries, such as ReactiveX, have been shown to significantly improve software design and have seen important industrial adoption over the last years. Asynchronous applications – which are notoriously error-prone to implement and to maintain – greatly benefit from reactive programming because they can be defined in a declarative style, which improves code clarity and extensibility.

In this paper, we tackle the problem of refactoring existing software that has been designed with traditional abstractions for asynchronous programming. We propose 2Rx, a refactoring approach to automatically convert asynchronous code to reactive programming. Our evaluation on top-starred GitHub projects shows that 2Rx is effective with common asynchronous constructs and it can provide a refactoring for 91.7% of their occurrences.

***Keywords*-refactoring; asynchronous programming; reactive programming; Java;**

over low level abstractions like threads, but come with their own limitations. For example, `AsyncTask` does not easily support composition, like sequencing multiple asynchronous computations.

Recently, Reactive Programming (RP) has emerged as a programming paradigm specifically addressing software that combines events [3]. Crucially, RP allows to easily express computations on event streams that can be chained and combined using high-order functional operators. This way, each operator can be scheduled independently, providing a convenient model for asynchronous programming. As a result, RP provides means to describe asynchronous programs in a declarative way. Previous research showed [illegible] software design of [illegible]

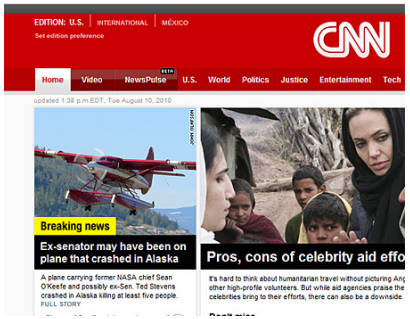
[ASE'19]

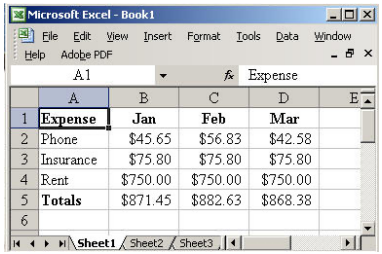
REACTIVE PROGRAMMING

Reactive Software is Ubiquitous

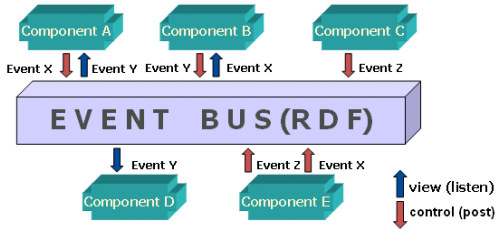
Web apps, IDEs, monitoring systems, …

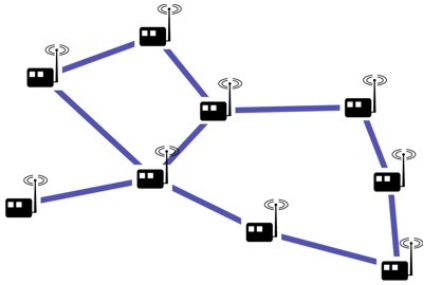
- Detect events
- Combine reactions
- Propagate changes
- Aggregate/filter





	A	B	C	D
1	Expense	Jan	Feb	Mar
2	Phone	$45.65	$56.83	$42.58
3	Insurance	$75.80	$75.80	$75.80
4	Rent	$750.00	$750.00	$750.00
5	Totals	$871.45	$882.63	$868.38
6				



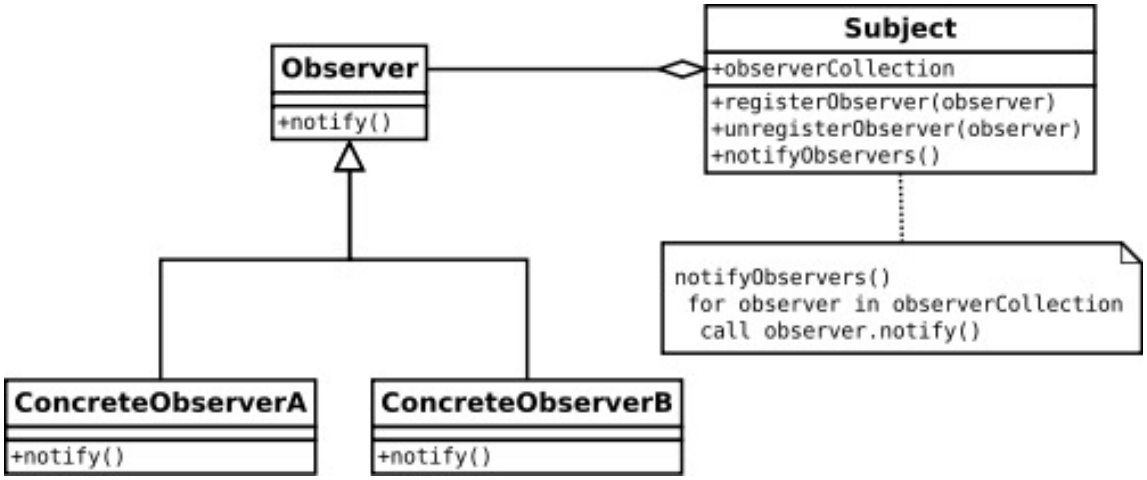


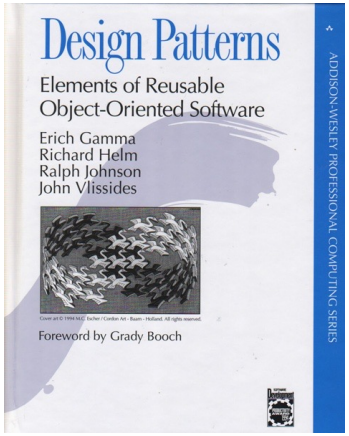
Reactive Applications are not Easy to Develop

Event handling:

- 30% of code in desktop applications
- 50% of bugs reported during production cycle

The (*good*? old) Observer Pattern

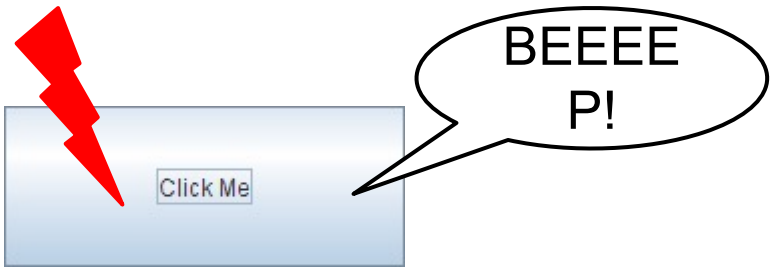




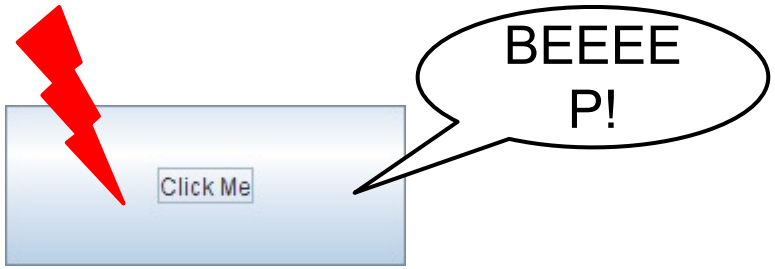
The Observer Pattern

What about Java Swing ?

- javax.swing



```
public class Beeper extends JPanel  implements ActionListener {
   JButton button;

   public Beeper() {
      super(new BorderLayout());
      button = new JButton("Click Me");
      button.setPreferredSize(new Dimension(200, 80));
      add(button, BorderLayout.CENTER);
      button.addActionListener(this);
   }
   public void actionPerformed(ActionEvent e) {
      Toolkit.getDefaultToolkit().beep();
   }
   private static void createAndShowGUI() { // Create the GUI and show it.
      JFrame frame = new JFrame("Beeper");         //Create and set up the window.
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JComponent newContentPane = new Beeper();  //Create and set up the content pane.
      newContentPane.setOpaque(true);
      frame.setContentPane(newContentPane);
      frame.pack();         //Display the window.
      frame.setVisible(true);
   }
   public static void main(String[] args) {
      javax.swing.SwingUtilities.invokeLater( new Runnable() { public void run() {createAndShowGUI();}});
   }
}
```



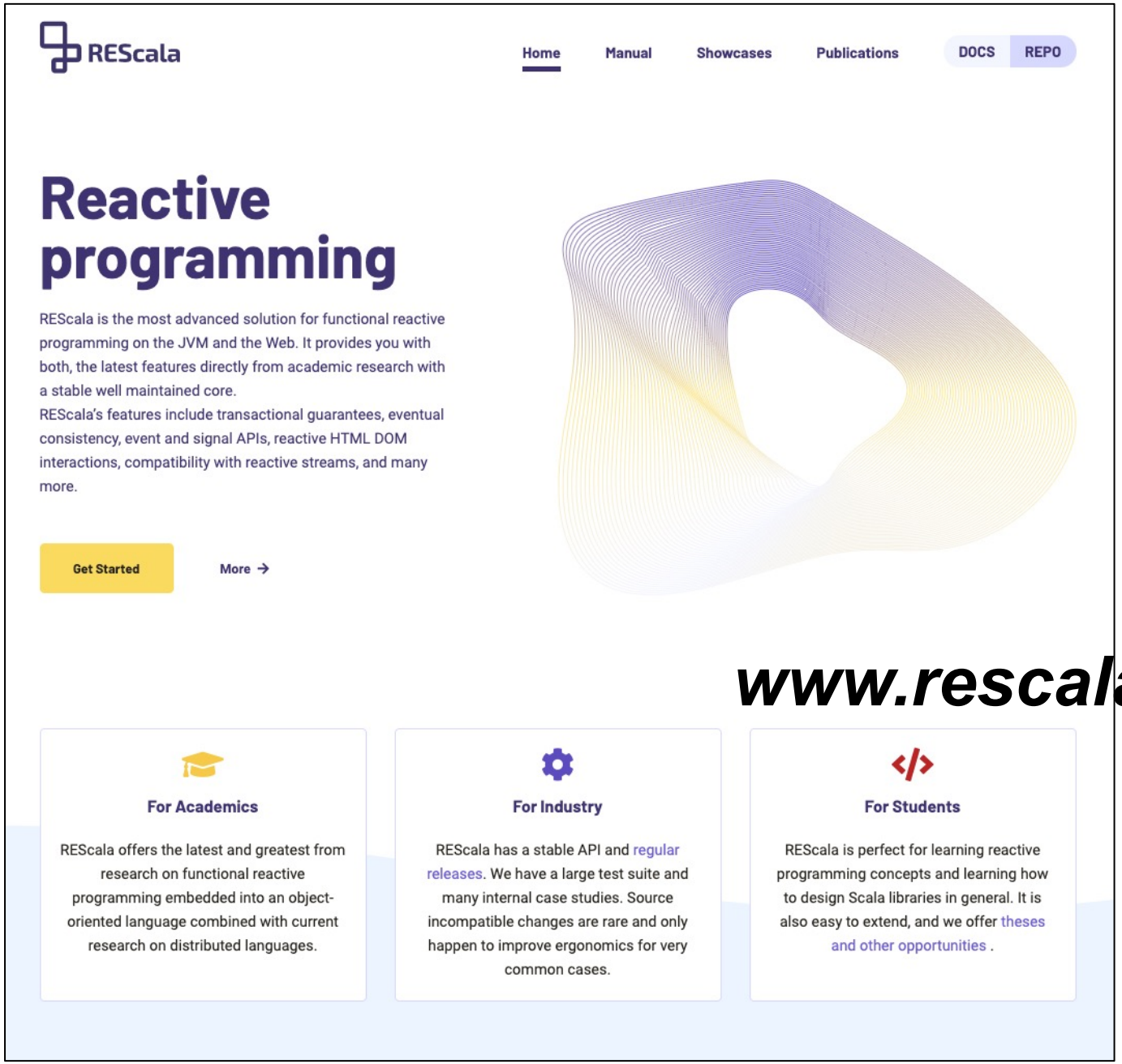
Reactive Programming

Can **language abstractions** help software design?

- Events
- Signals

Similar examples:

- fault-tolerance -> **exceptions**
- encapsulation -> visibility **modifiers**
- extensibility -> **inheritance**



www.rescala-lang.com

Events

Registered handlers are executed when the event fires

- The actual parameter is provided to the handler

```
val e = Evt()
e += { x => println("The value of the event is: " + x) }
e(10)
e(11)

-- output ----
The value of the event is: 10
The value of the event is: 11
```

Constrains as Reactive Values

What about expressing *functional dependencies* as constraints ?

```
val a = 3
val b = 7
val c = a + b // Statement
println(c)
> 10
a= 4
println(c)
> 10
```

```
val a = 3
val b = 7
val c := a + b // Constraint
println(c)
> 10
a= 4
println(c)
> 11
```

Constrains as Reactive Values

Signals: (pure)reactive expressions

Constraints “automatically” enforced

.now returns the current value

Vars: primitive reactive values

Updated “manually”

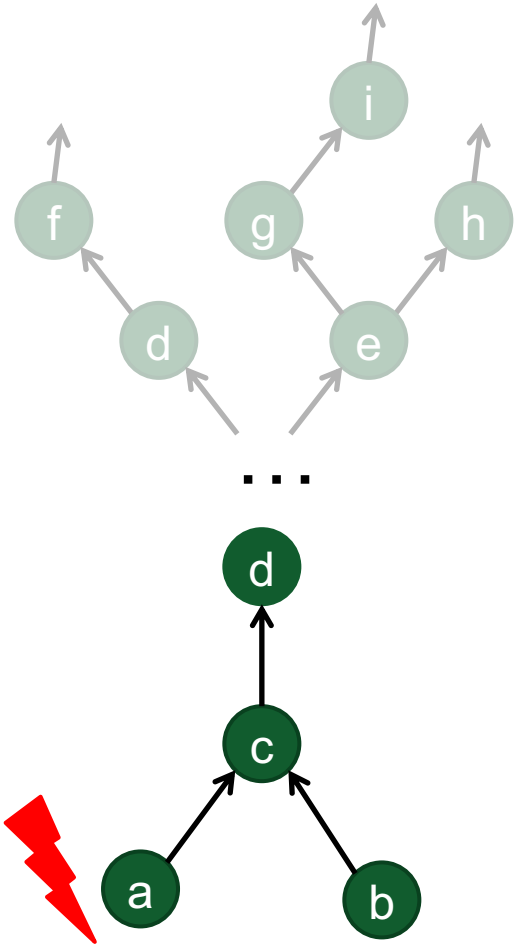
```
val a = Var(3)
val b = Var(7)
val c = Signal{ a() + b()
}

println(c.now)
> 10

a.set(4)
println(c.now)
> 11
```

Reference Model

Change propagation model

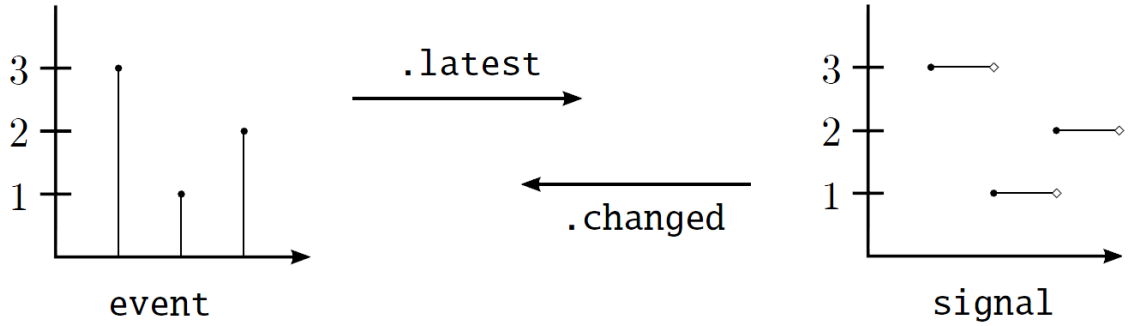
- Dependency graph
- Push-driven evaluation

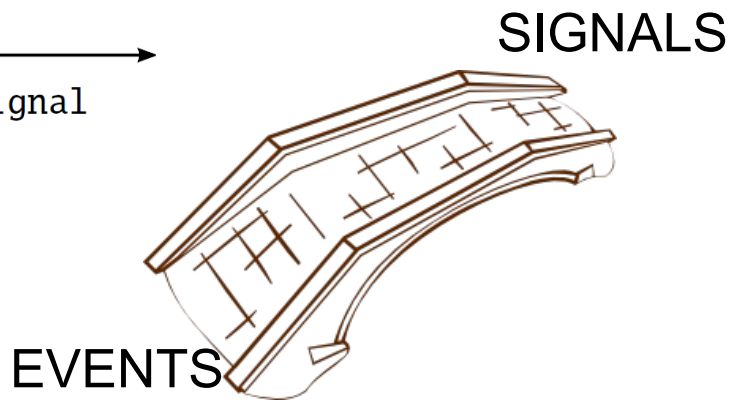
```
val a = Var(3)
val b = Var(7)
val c = Signal{ a() + b()
}
val d = Signal { 2 * c() }
 …
```



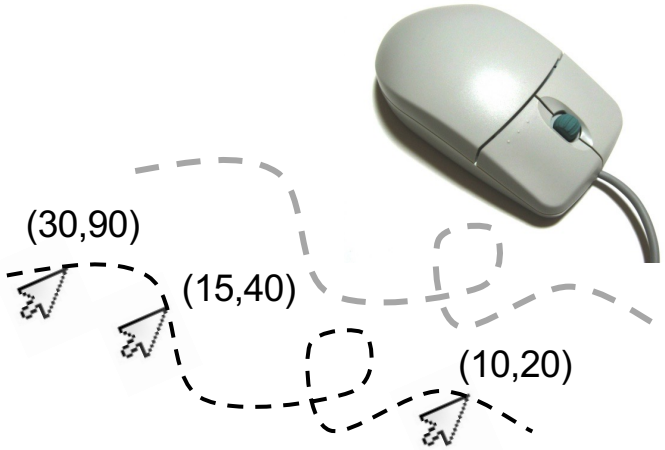
Basic Conversion Functions

```
Changed :: Signal -> Event

Latest :: Event -> Signal
```



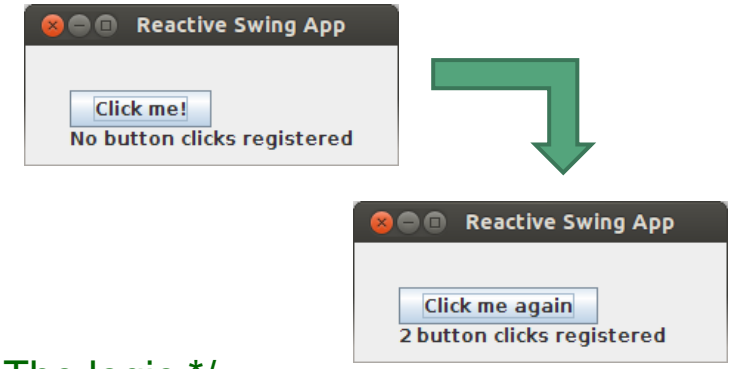


Mouse with RP




```
val position: Signal[(Int,Int)] = mouse.position
val shiftedPosition: Signal[(Int,Int)] = Signal{ mouse.position + (10,
10) }

evt clicked: Event[Unit] = mouse.clicked
val lastClick: Signal[(Int,Int)] = position snapshot clicked
```

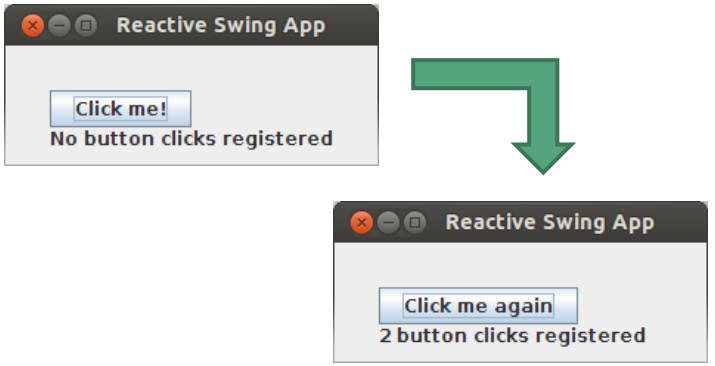
Example: Observer




```
/* Create the graphics */
title = "Reactive Swing App"
val button = new Button {
  text = "Click me!"
}
val label = new Label {
  text = "No button clicks registered"
}
contents = new BoxPanel(Orientation.Vertical) {
  contents += button
  contents += label
}
```

```
/* The logic */
listenTo(button)
var nClicks = 0
reactions += {
  case ButtonClicked(b) =>
    nClicks += 1
    label.text = "Number of button clicks: " +
nClicks
    if (nClicks > 0)
      button.text = "Click me again"
}
```

Example: Signals




```
title = "Reactive Swing App"
val label = new ReactiveLabel
val button = new ReactiveButton

val nClicks = button.clicked.fold(0) {(x, y) => x + 1}

label.text = Signal { ( if (nClicks() == 0) "No" else nClicks() ) + " button clicks registered" }

button.text = Signal { "Click me" + (if (nClicks() == 0) "!" else " again " )}

contents = new BoxPanel(Orientation.Vertical) {
    contents += button
    contents += label
}
```

Claim: RP beats OO (Observer)

Easier to compose

Declarative style

Easier program comprehension

State management not explicit

Automatic memory management

Flapjax: A Programming Language for Ajax Applications

"Obviously, the Flapjax code may not appear any 'easier' to a first-time reader"

[Leo A. Meyerovich, Arjun Guha, Jacob Baskin, Gregory H. Cooper, Michael Greenberg, Aleks Bromfield, Shriram Krishnamurthi, *Flapjax: A Programming Language for Ajax Applications, OOPSLA'09*]

This paper presents Flapjax, a language designed for contemporary Web applications. These applications communicate with servers and have rich, interactive interfaces. Flapjax provides two key features that simplify writing these applications. First, it provides *event streams*, a uniform abstraction for communication within a program as well as with ex-

Keywords JavaScript, Web Programming, Functional Reactive Programming

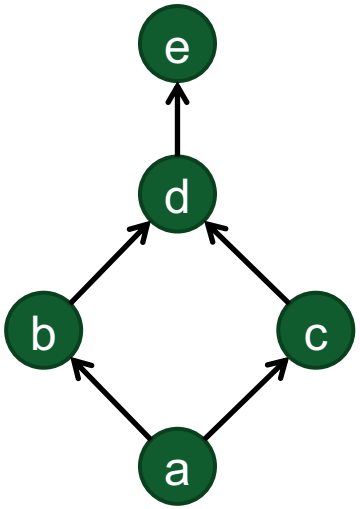
1. Introduction

The advent of broadband has changed the structure of application software. Increasingly, desktop applications are mi-

The study

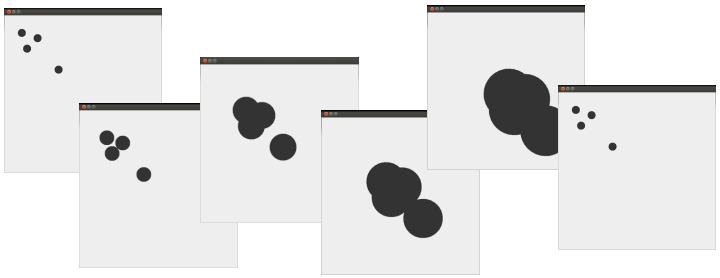
10 applications, ~130 subjects

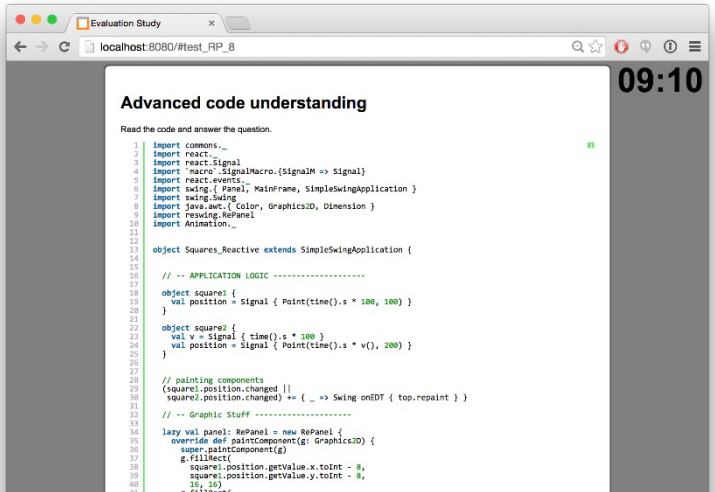
- RP and OO group (**between** subj.)
- Questions for comprehension



What to measure?

- **Time** to answer a question
- Amount of **correct answers**

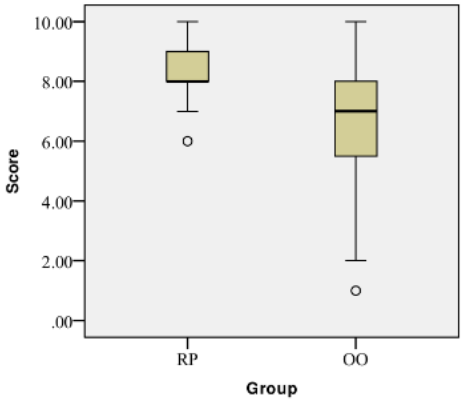


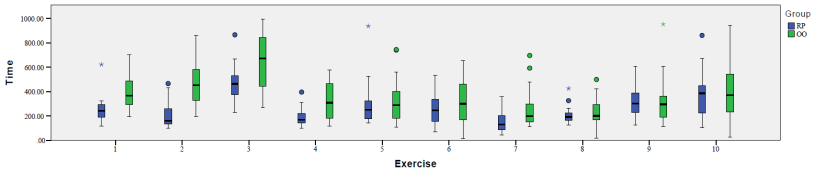


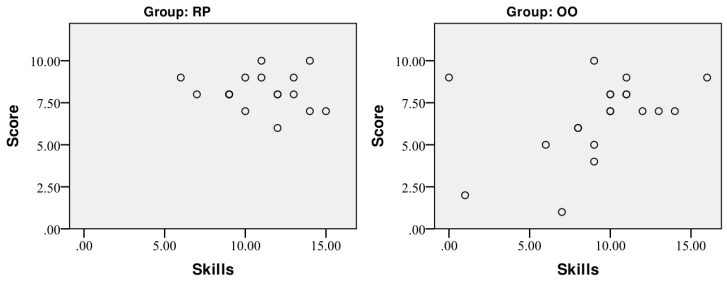
Results

REScala **increases correctness** of program comprehension

In REScala, comprehension is **not more time-consuming**







[Guido Salvaneschi, Sven Amann, Sebastian Proksch, Mira Mezini, **An Empirical Study on Program Comprehension with Reactive Programming,** FSE'14.]

[G. Salvaneschi, S. Proksch, S. Amann, S. Nadi, M. Mezini, ***On the Positive Effect of Reactive Programming on Software Comprehension: An Empirical Study***, TSE'17]

Teaching Reactive Programming

Master course (9CP)

Software Engineering: Design & Construction

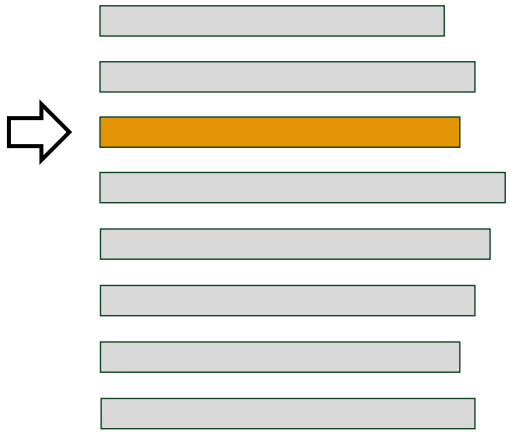
- Design patterns
- Domain specific languages
- Software architectures
- Reactive Programming with REScala
- ...

HOW TO DEBUG REACTIVE PROGRAMS !?

Debugging for Reactive Programing

Traditional debugging

(Imperative)



~~Step over statements~~

~~Inspect state~~

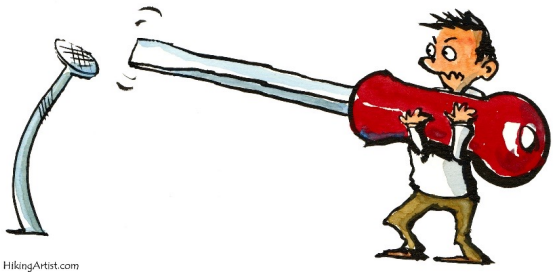
```
0x051DE590   a0 e5 1d 05 4c
0x051DE5A0   20 e6 1d 05 b1
0x051DE5B0   40 e6 1d 05 00
0x051DE5C0   bc c3 94 70 b4
```

Program Stack

Reactive Programming

(Declarative)

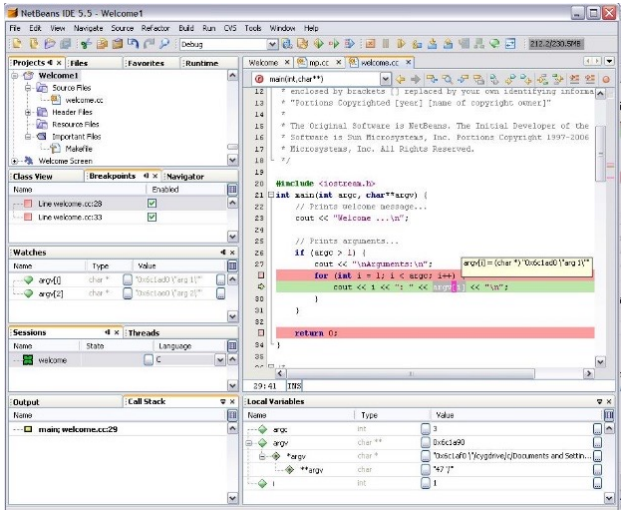
Signals

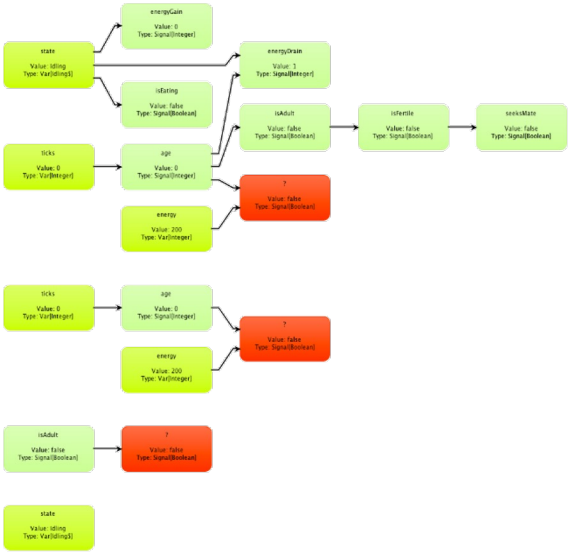


Abstract over state

!?

A Paradigm Shift

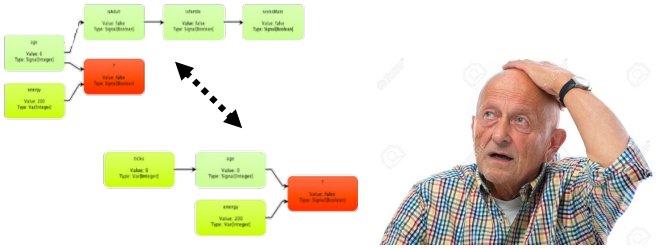




Traditional Debugging	RP Debugging
Stepping over statements	Stepping over the dependency graph
Breakpoint on line X	Breakpoint on node X
Inspect memory	Inspect values in the dependency graph
Navigate object references	Navigate signals in the graph

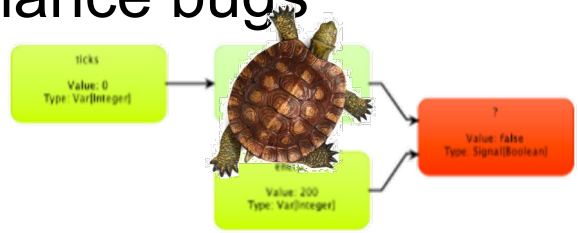
Bug Hunting with Reactive Debugging

Missing dependencies

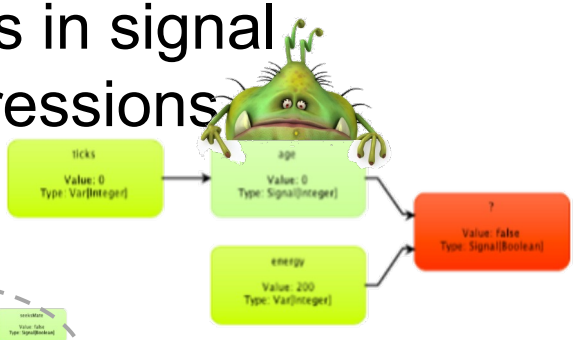


Understanding RP programs

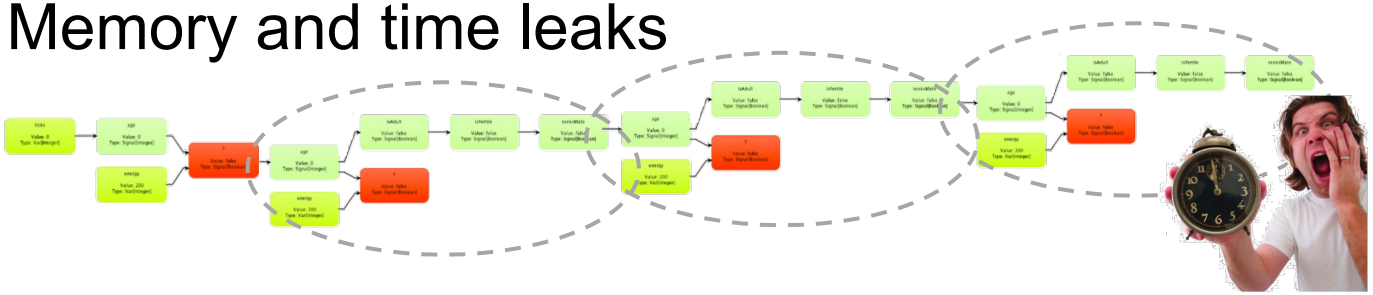
Performance bugs



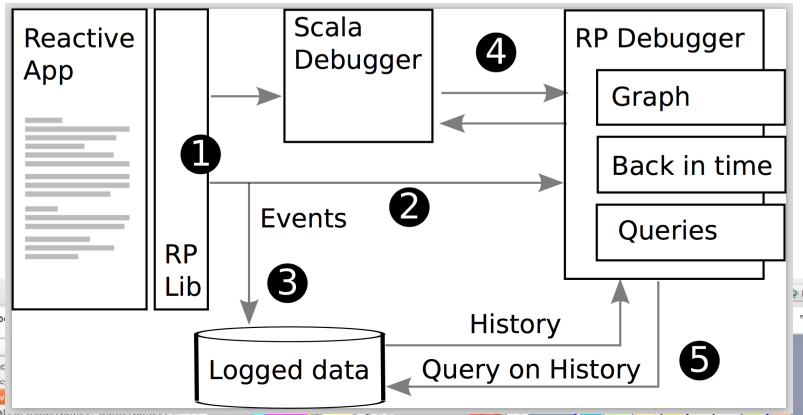
Bugs in signal expressions



Memory and time leaks



Reactive Inspector
(Eclipse plugin - Scala IDE)



NODE SEARCH

NODE BREAKPOINTS

TREE INSPECTION

NODE QUERIES

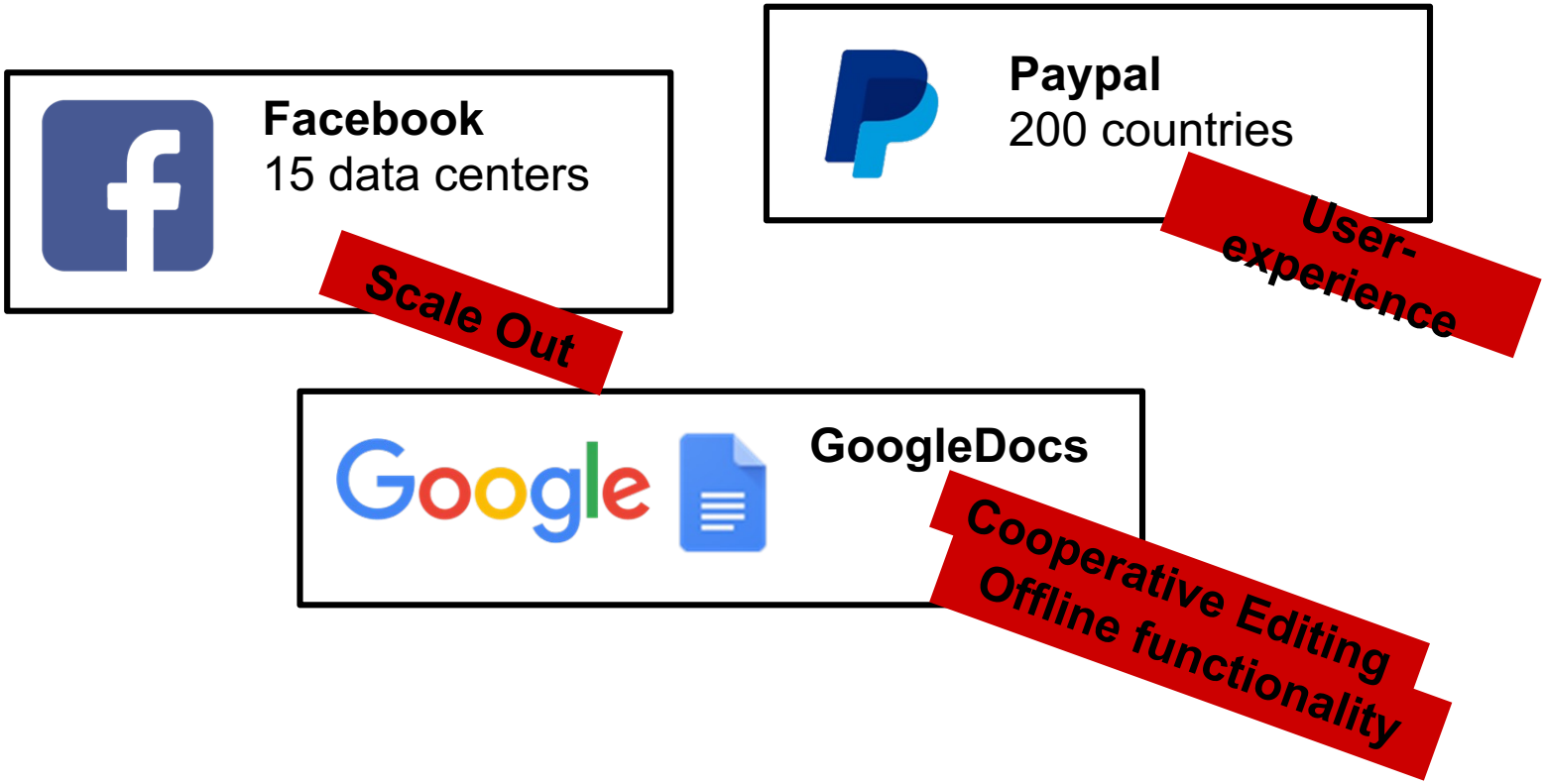
BACK-IN-TIME DEBUGGING

REACTIVE BREAKPOINTS

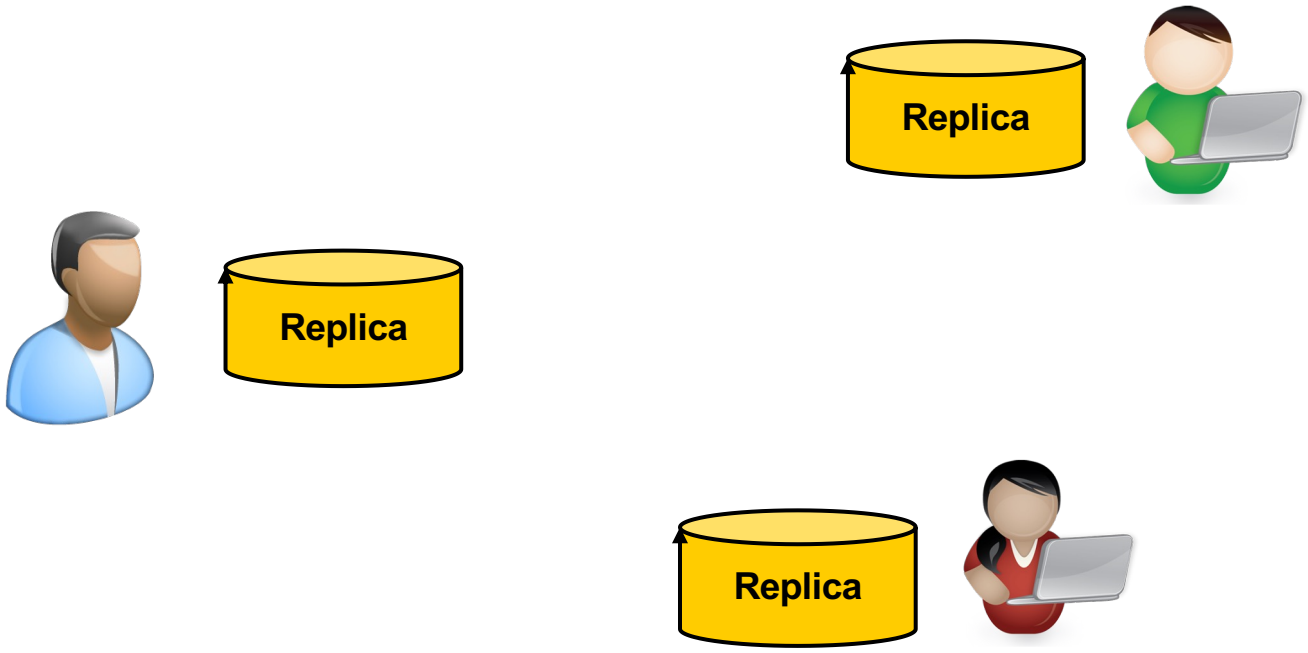
CONSISTENCY

Replicated Data

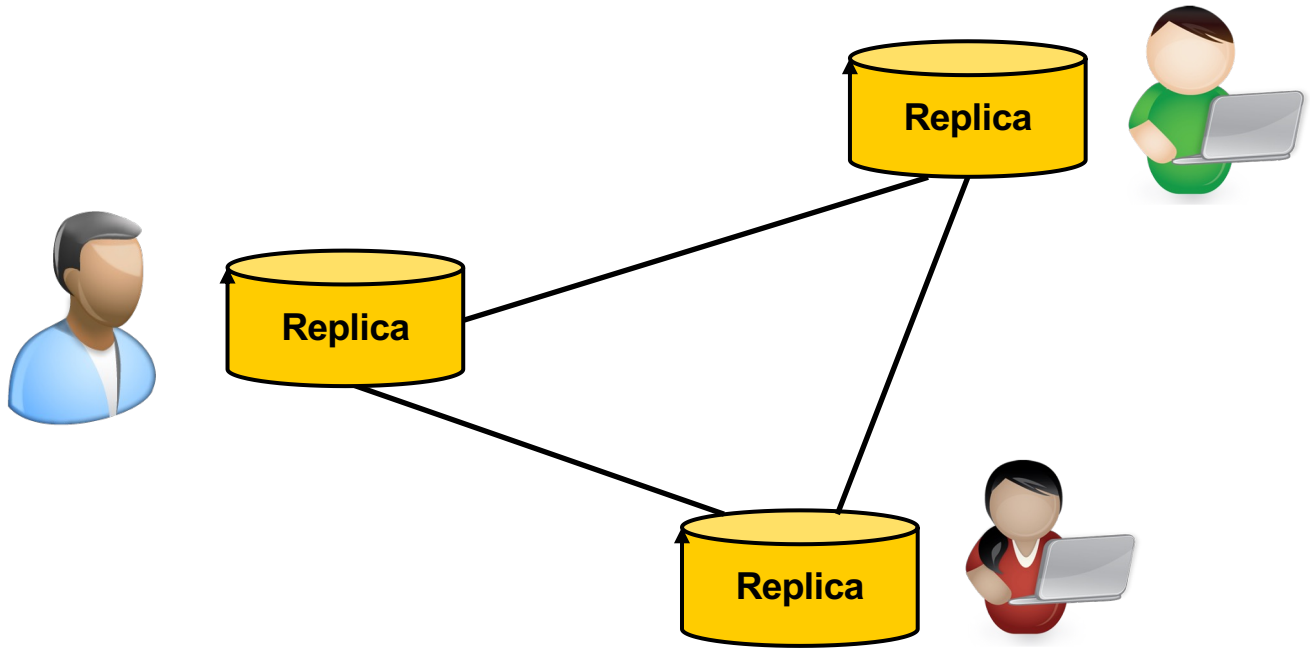
Replicated data is used to…



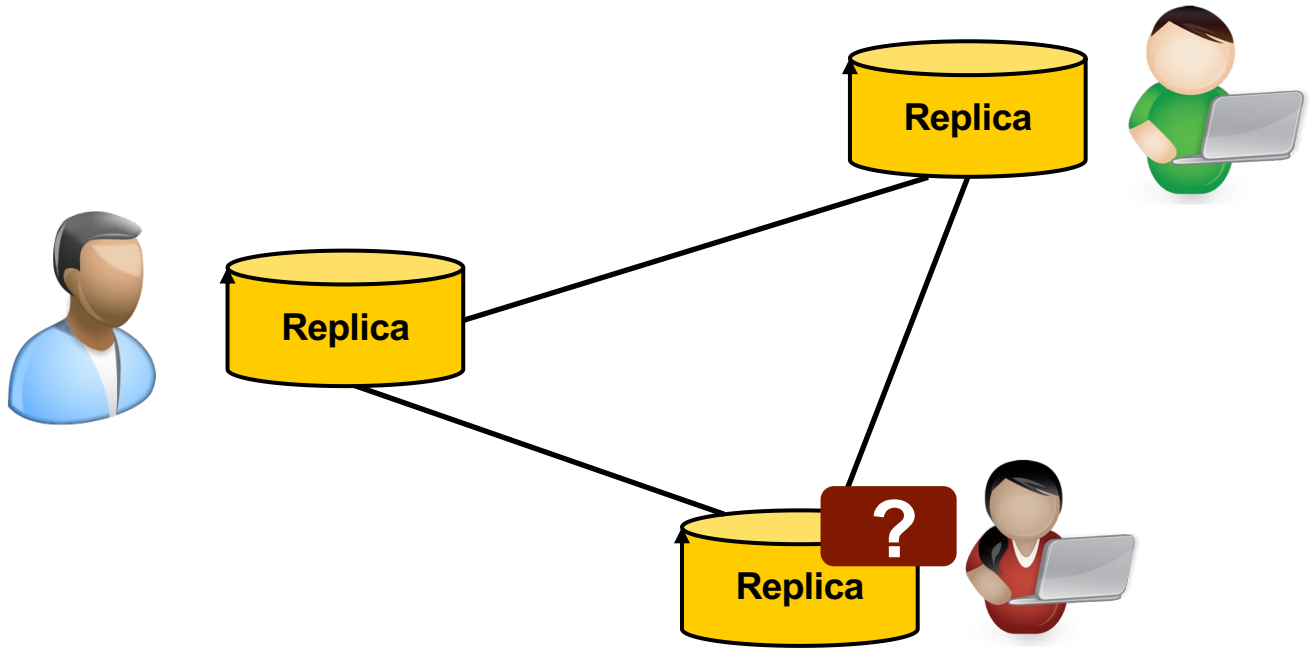
Example: Replicated Data



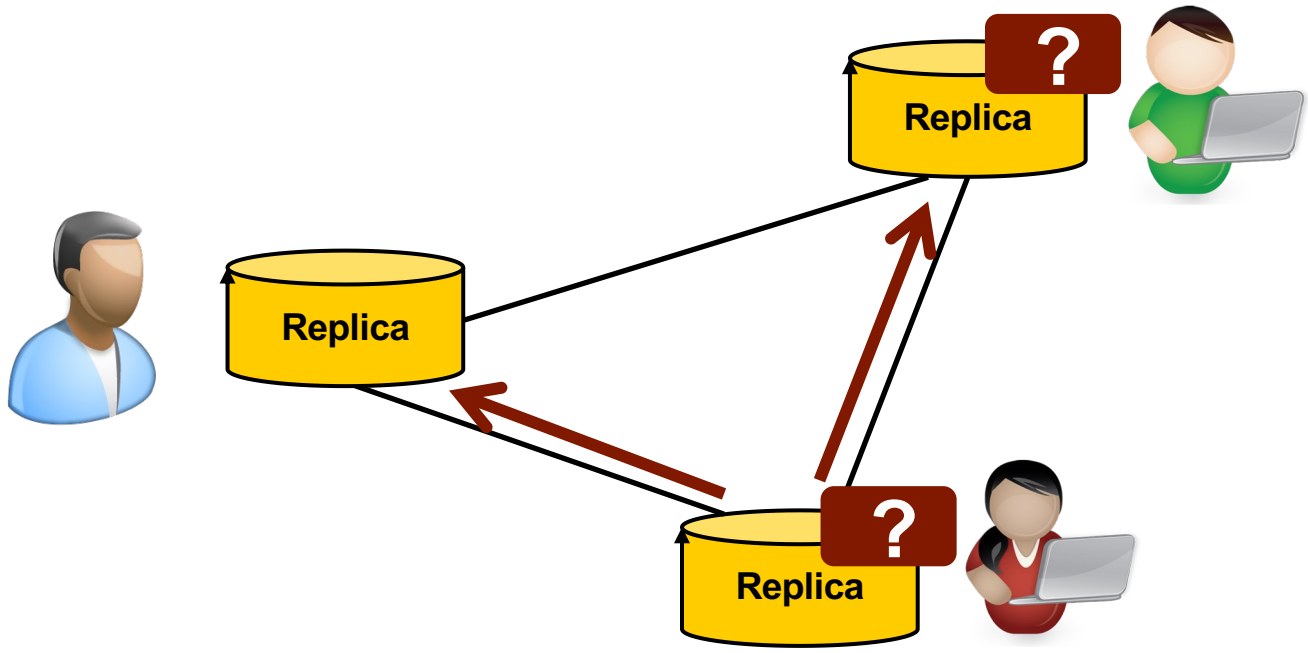
Example: Replicated Data



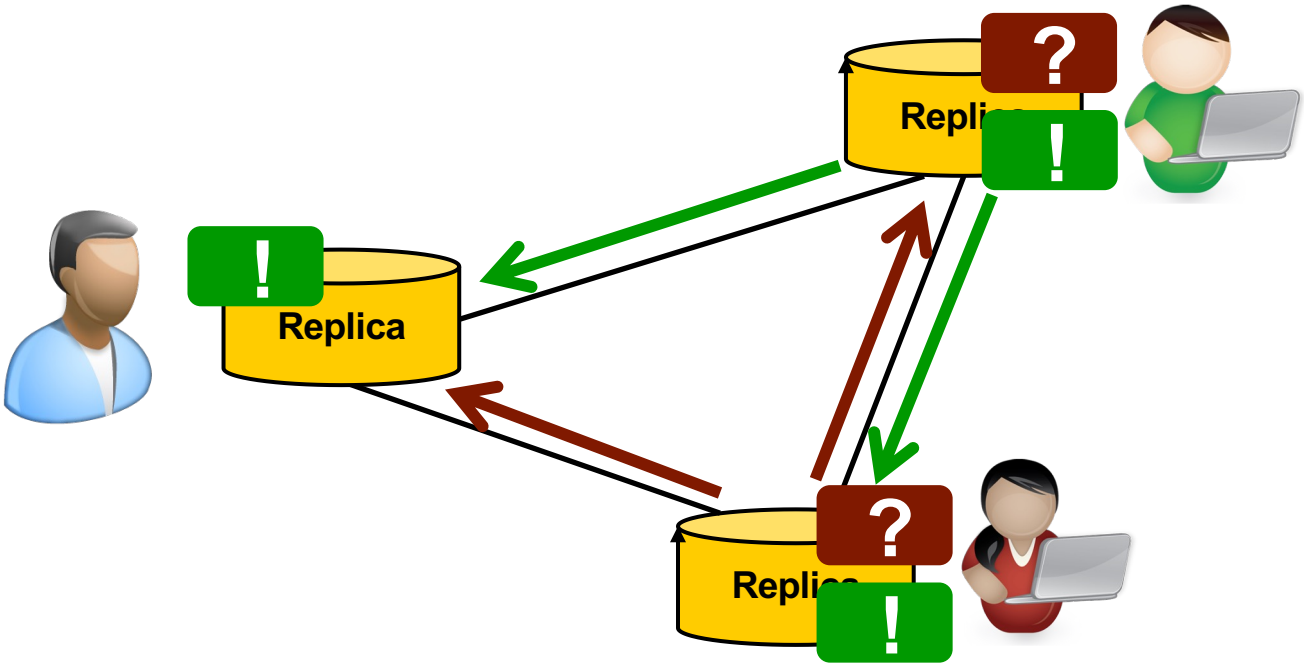
Example: Consistency



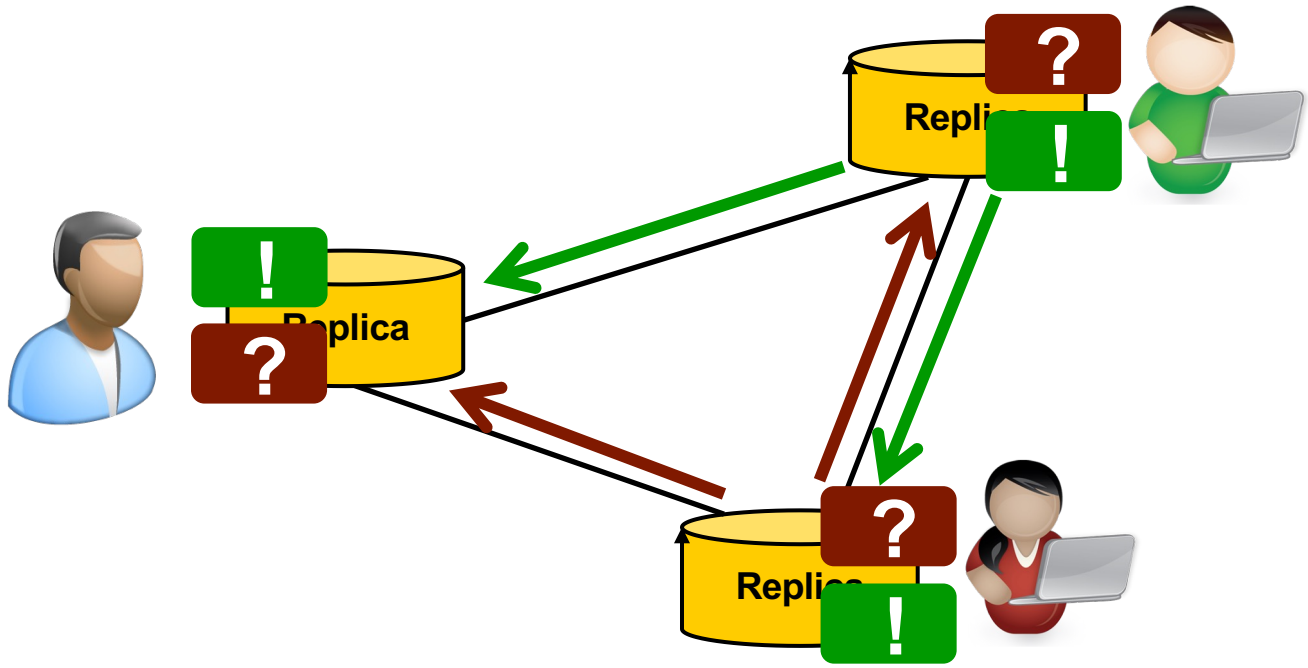
Example: Consistency



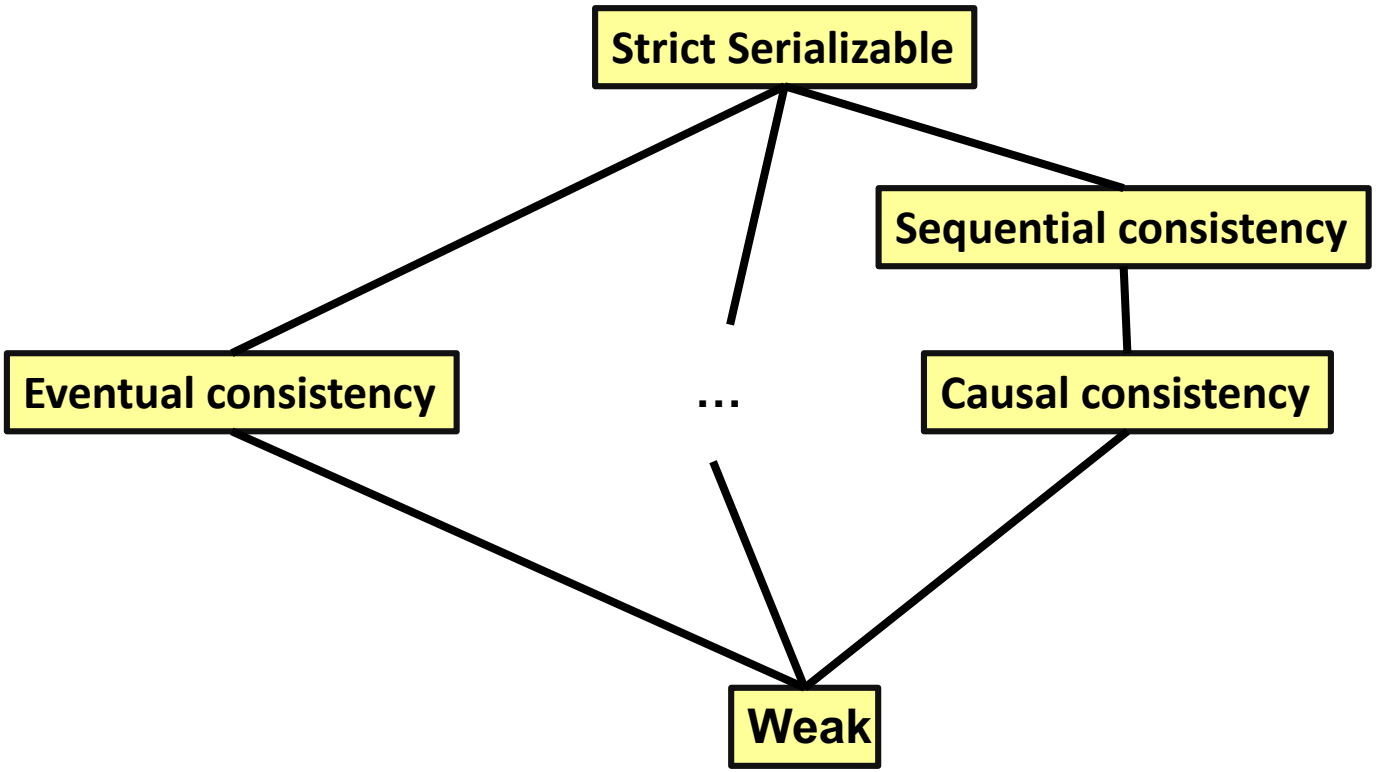
Example: Consistency



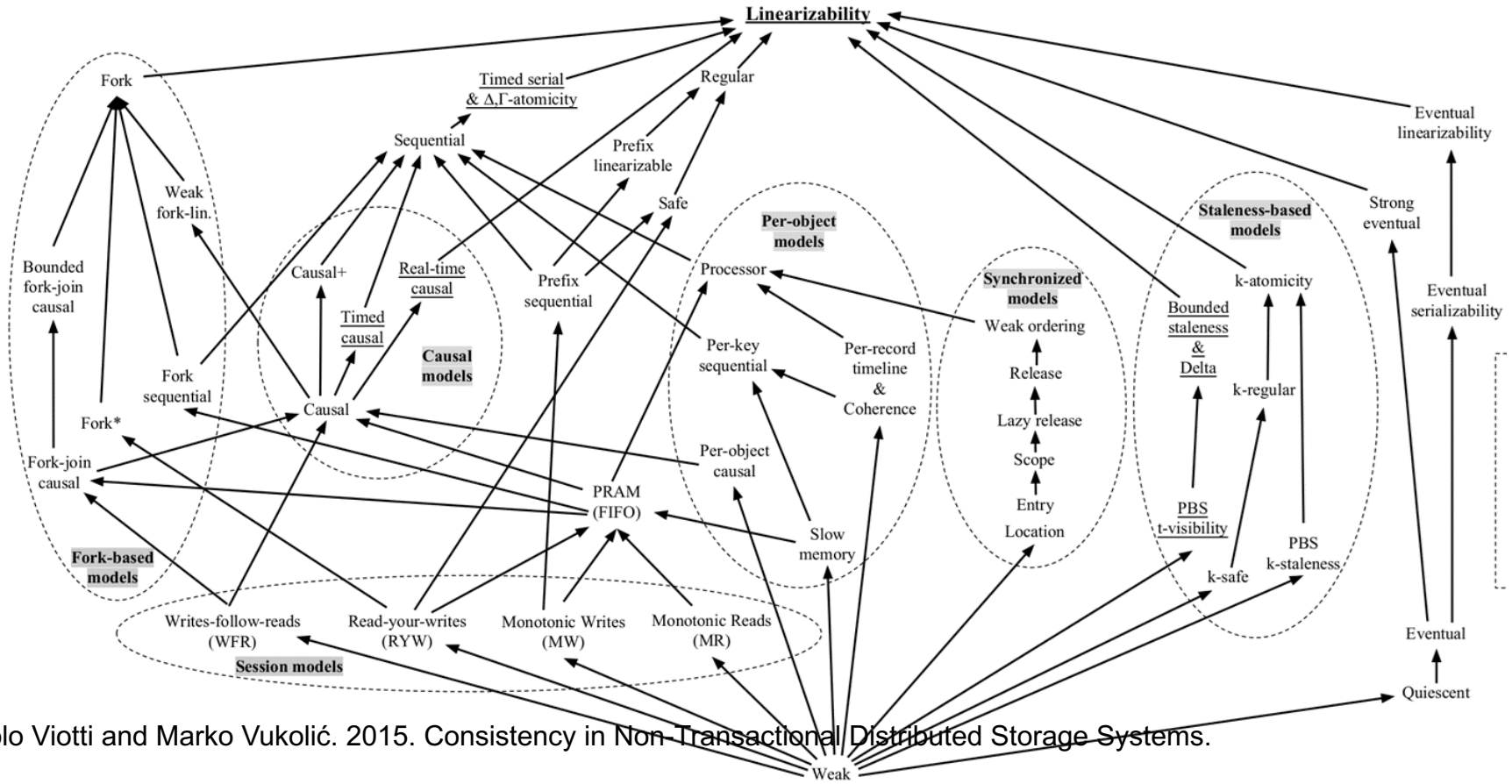
Example: Consistency



Consistency Levels

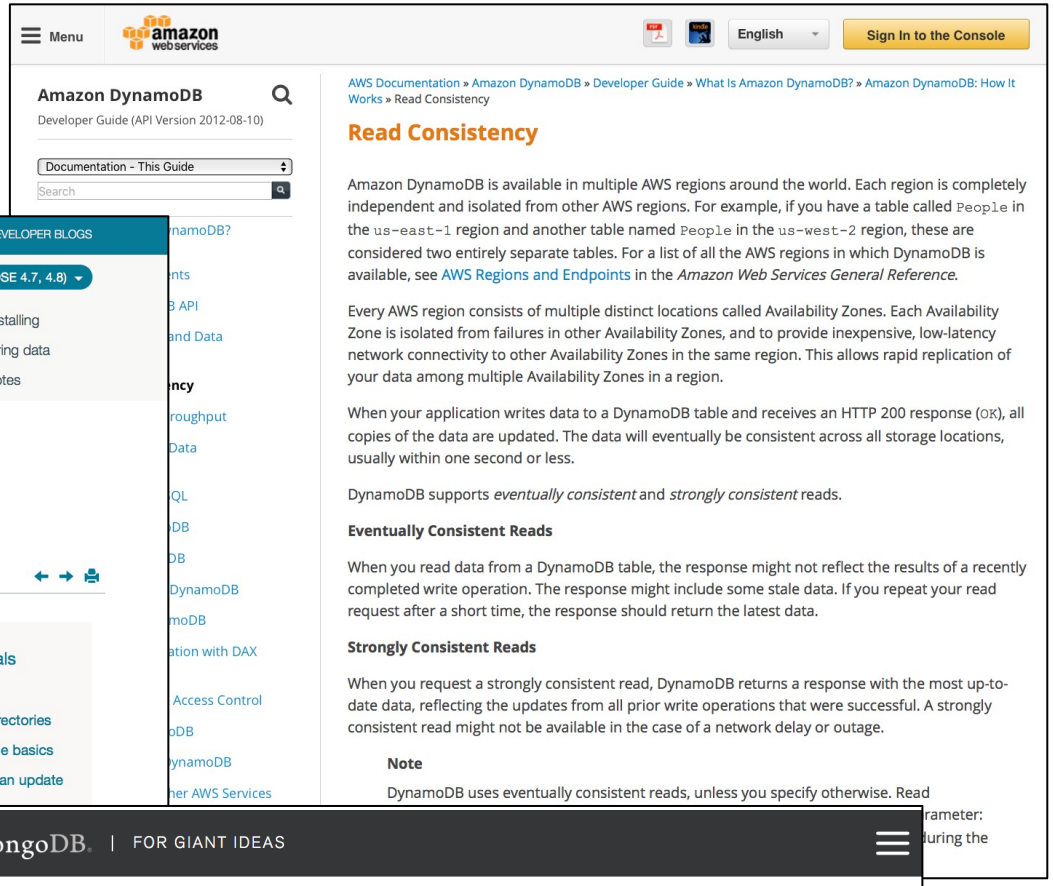


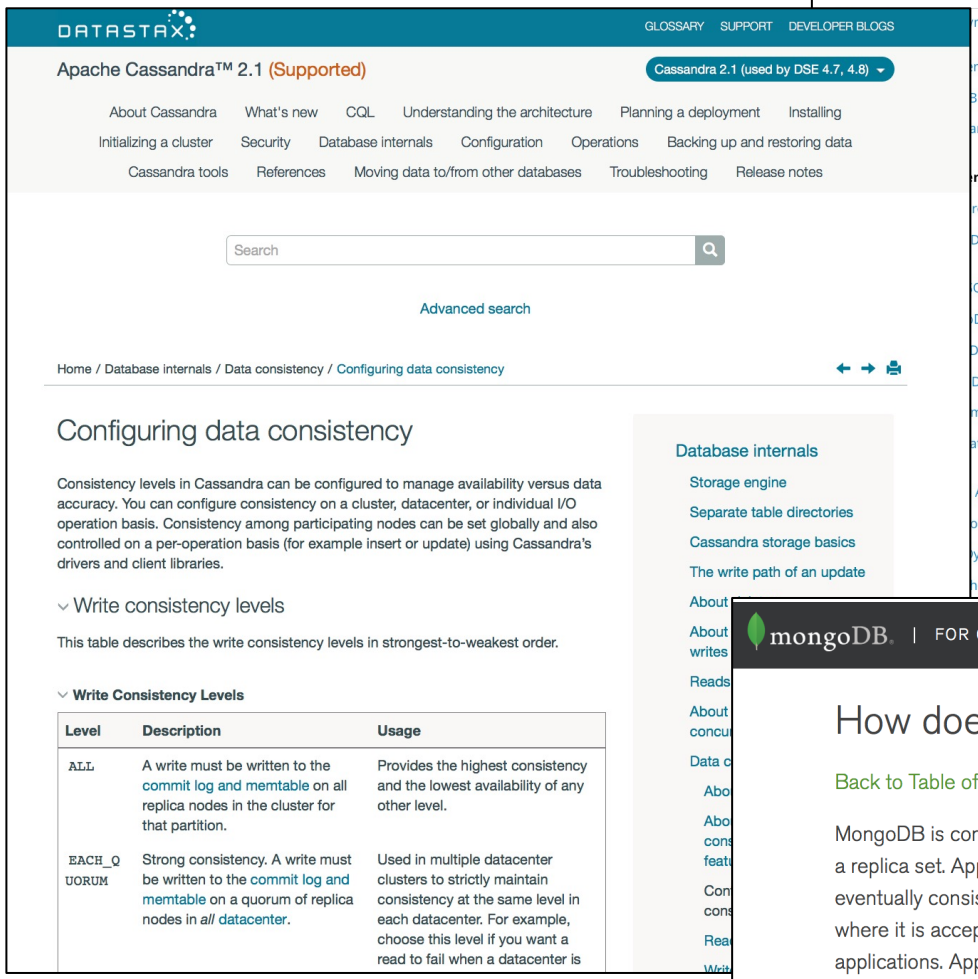
Consistency Levels

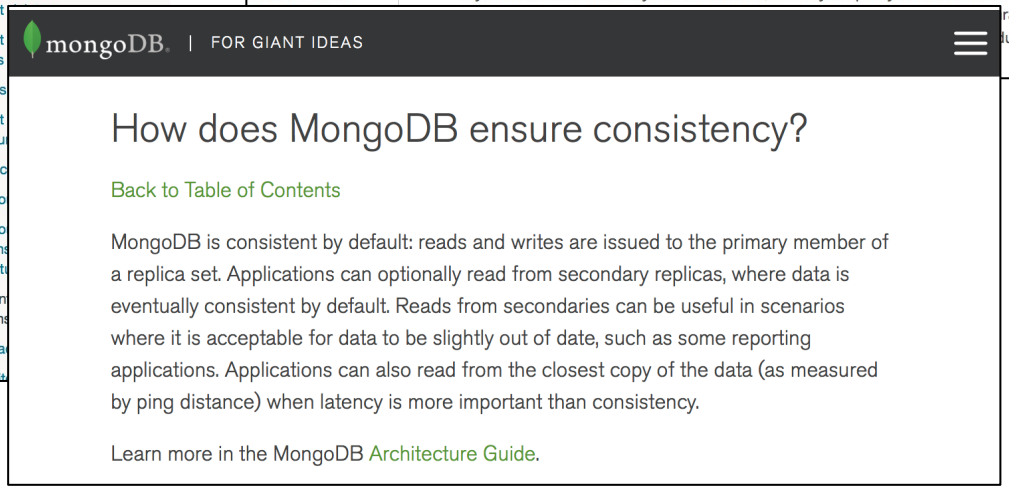


Paolo Viotti and Marko Vukolić. 2015. Consistency in Non-Transactional Distributed Storage Systems.

Eventual consistency
Causal consistency
Fifo consistency
….







ConSysT Programming Framework

Combine multiple consistency levels in the same application

High consistency (slow)
Payment, Security, …

Low consistency (fast)
Logging, messaging, …

ConSysT Programming Framework

```
class Concert {
    Date date;
    Ref<@Weak ConcertHall> hall;
    Ref<@Weak Band> band;
    Ref<@Strong Counter> soldTickets;

  …
}
```

Programming model supports
explicit consistency levels

Information flow static analysis
ensures safe mixing of consistency levels

[M. Köhler, N. Eskandani, P. Weisenburger, A. Margara, G. Salvaneschi,
Rethinking Safe Consistency in Distributed Object-Oriented Programming, OOPSLA'20]

Type System: Ensures Safety

```
Ref<@Weak Counter> weakCounter;
Ref<@Strong Counter> strongCounter;

weakCounter.ref.value = strongCounter.ref.value;
```

Allowed

Strong to Weak

```
Ref<@Weak Counter> weakCounter;
Ref<@Strong Counter> strongCounter;

strongCounter.ref.value = weakCounter.ref.value;
```

Disallowed

Weak to Strong

```
Ref<@Weak Counter> weakCounter;
Ref<@Strong Counter> strongCounter;

if (weakCounter.ref.value == 0) {
        strongCounter.ref.value = 0;
}
```

Disallowed

Weak to Strong (implicit)

View on GitHub

ConSysT

Tunable, safe consistency meets object–oriented programming.

https://consyst-project.github.io/

What is ConSysT?

ConSysT is an distributed object-oriented language. Objects can be replicated with different levels of consistency. The type system ensures that consistency levels are safely.

Multiple consistency levels

Each replicated object comes with its own consistency level.

Safe mixing of consistencies

The static type system ensures correct mixing of consistency levels.

Object–oriente programming

Consistency is full integrated with obje oriented abstractio

Overview

In **ConSysT**, the main abstraction are *replicated objects* that are fully integrated into *object-oriented* language. Replicated objects have a *consistency level* specified by t developer.

Distribution

Easily distribute your your data across your local network, datacenters or geo-replica devices. *Replicated objects* allow to distribute and perform operations on your data. ConSysT is implemented as a language extension to Java, you can create replicated

---

Rethinking Safe Consistency in Distributed Object-Oriented Programming

MIRKO KÖHLER, Technische Universität Darmstadt, Germany
NAFISE ESKANDANI, Technische Universität Darmstadt, Germany
PASCAL WEISENBURGER, Technische Universität Darmstadt, Germany
ALESSANDRO MARGARA, Politecnico di Milano, Italy
GUIDO SALVANESCHI, Universität St. Gallen, Switzerland

Large scale distributed systems require to embrace the trade off between consistency and availability, accepting lower levels of consistency to guarantee higher availability. Existing programming languages are, however, agnostic to this compromise, resulting in consistency guarantees that are the same for the whole application and are implicitly adopted from the middleware or hardcoded in configuration files.

In this paper, we propose to integrate availability in the design of an object-oriented language, allowing developers to specify different consistency and isolation constraints in the same application at the granularity of single objects. We investigate how availability levels interact with object structure and define a type system that preserves correct program behavior. Our evaluation shows that our solution performs efficiently and improves the design of distributed applications.

CCS Concepts: • **Software and its engineering → Distributed programming languages**; • **Information systems** → *Remote replication.*

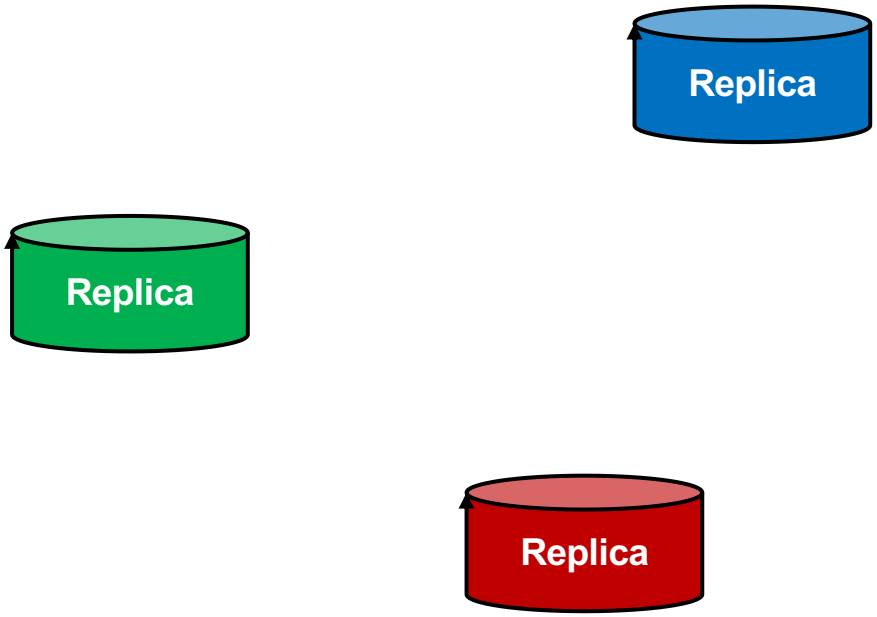
Additional Key Words and Phrases: replication, consistency, type systems, Java

ACM Reference Format:
Mirko Köhler, Nafise Eskandani, Pascal Weisenburger, Alessandro Margara, and Guido Salvaneschi. 2020. Rethinking Safe Consistency in Distributed Object-Oriented Programming. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 188 (November 2020), 30 pages. https://doi.org/10.1145/3428256

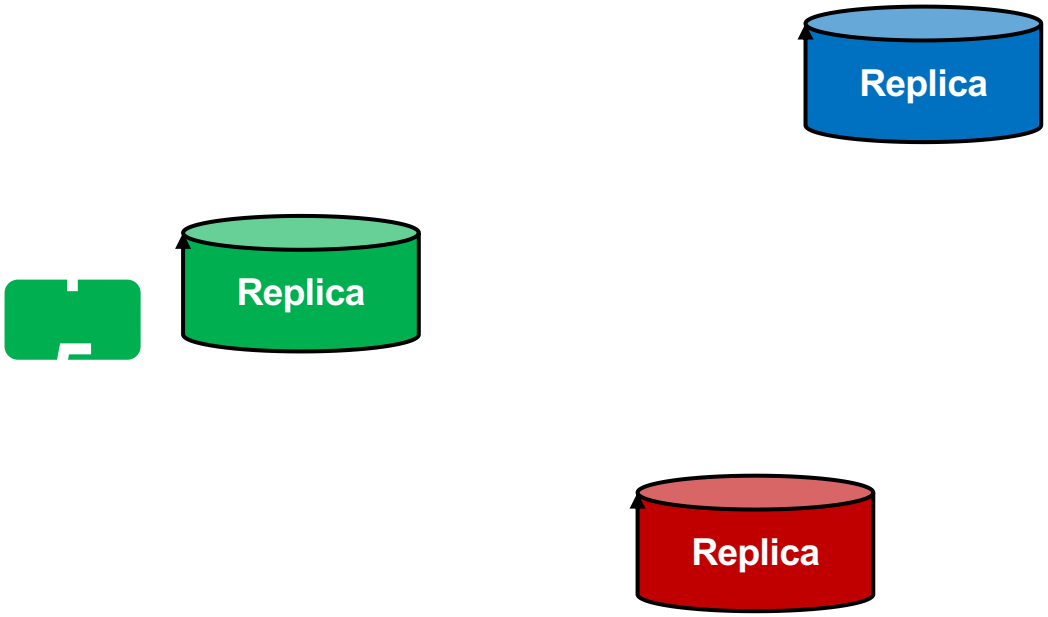
1 INTRODUCTION

Replication is fundamental in large-scale distributed systems to improve data availability and data access latency. By replicating data at multiple sites, distributed components can interact with the closest (e.g., local) replica without incurring the cost of remote data access. However, synchronizing replicas to ensure a globally consistent view implies onerous performance costs that can hamper rather than improve availability and latency. For this reason, weakly consistent models have been widely adopted to limit synchronization overhead at the expense of correctness [Vogels 2009].

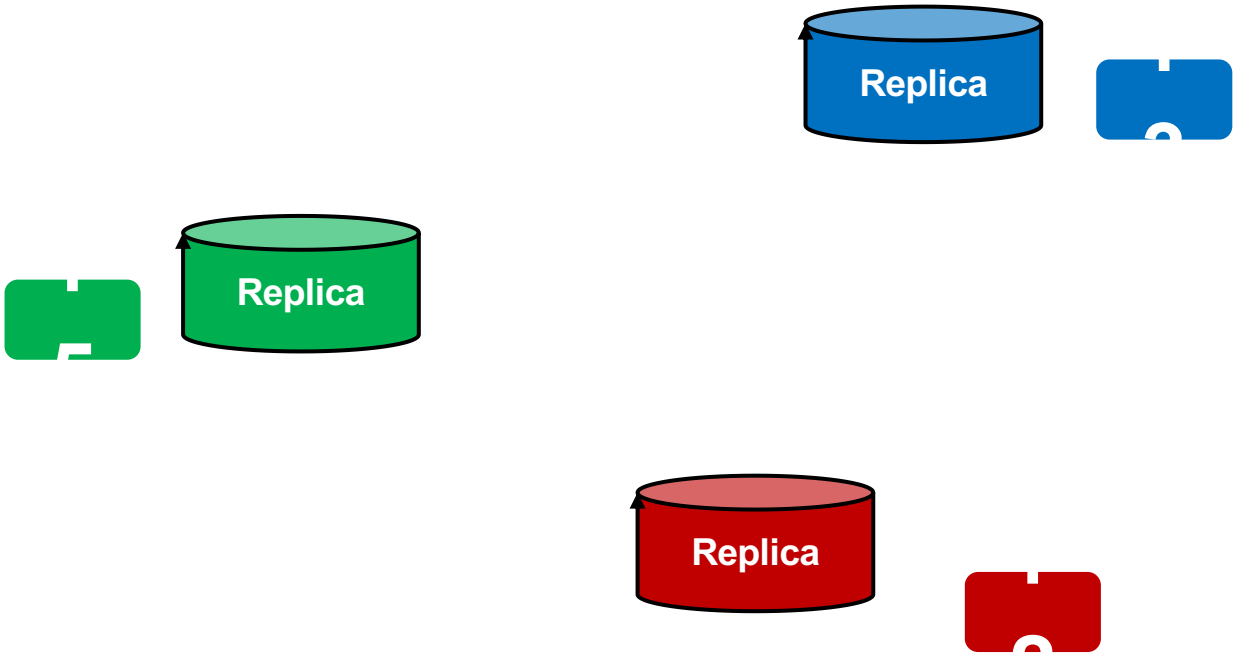
Eventual Consistency: CRDTs



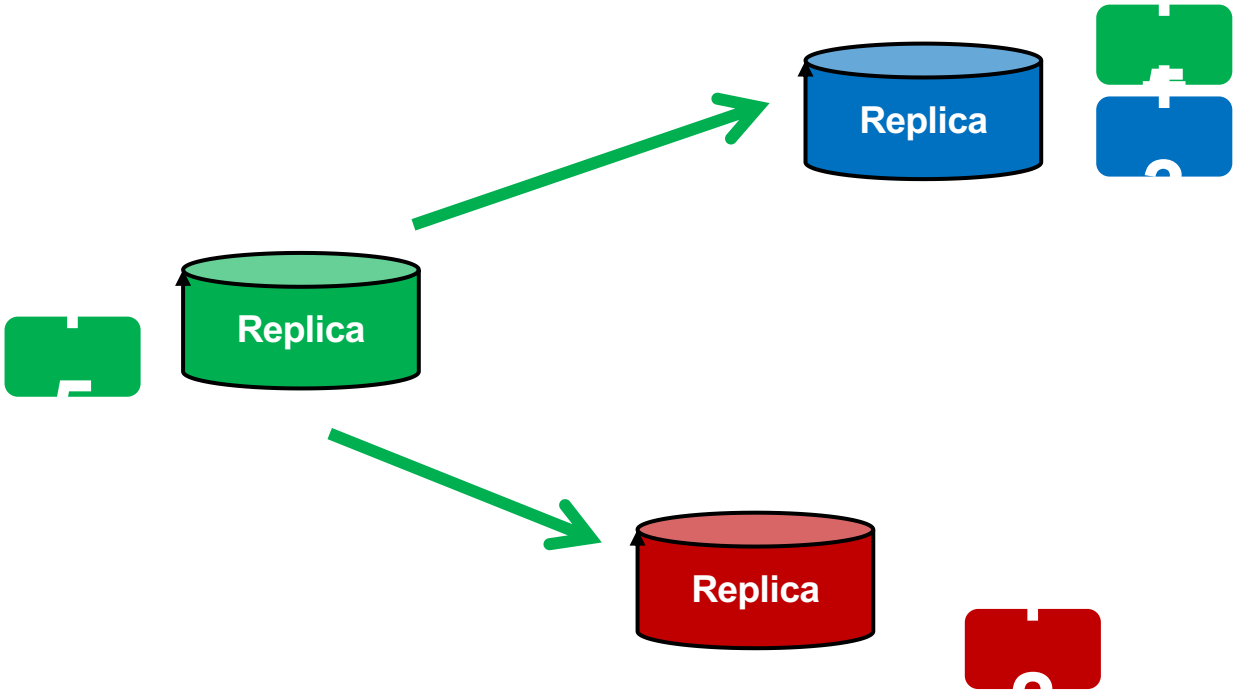
Eventual Consistency: CRDTs



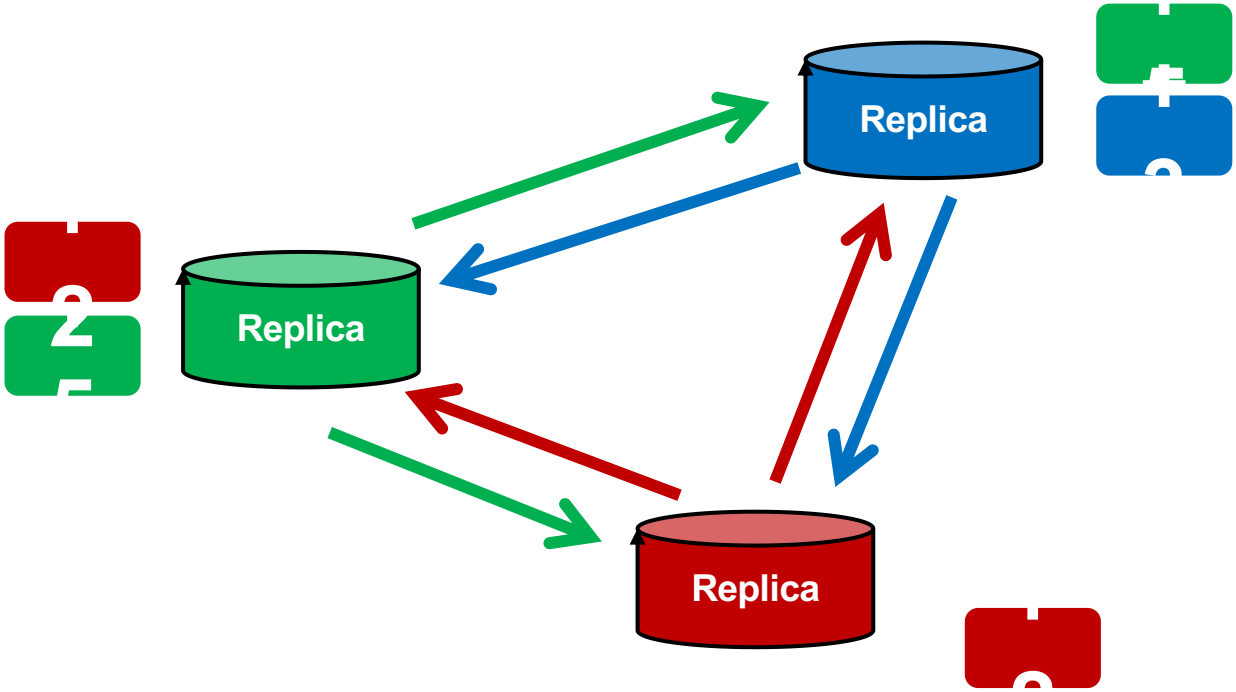
Eventual Consistency: CRDTs



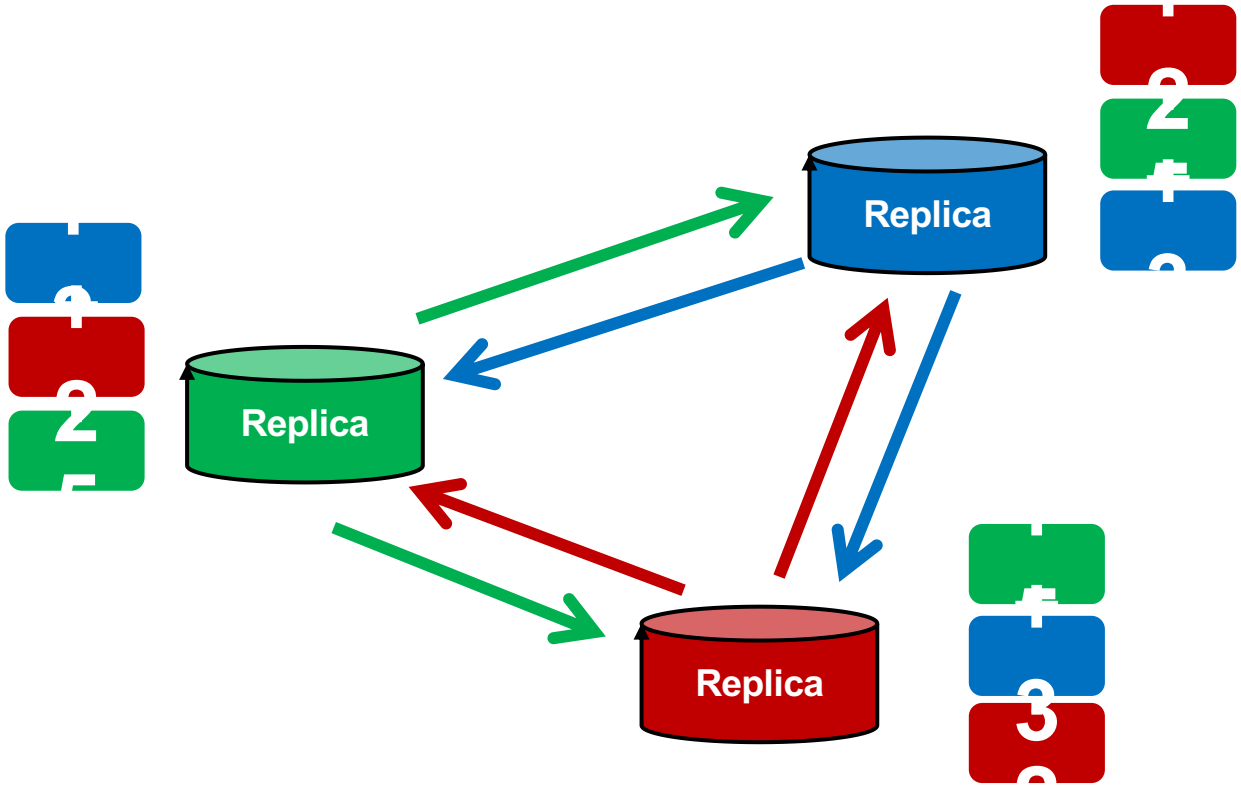
Eventual Consistency: CRDTs



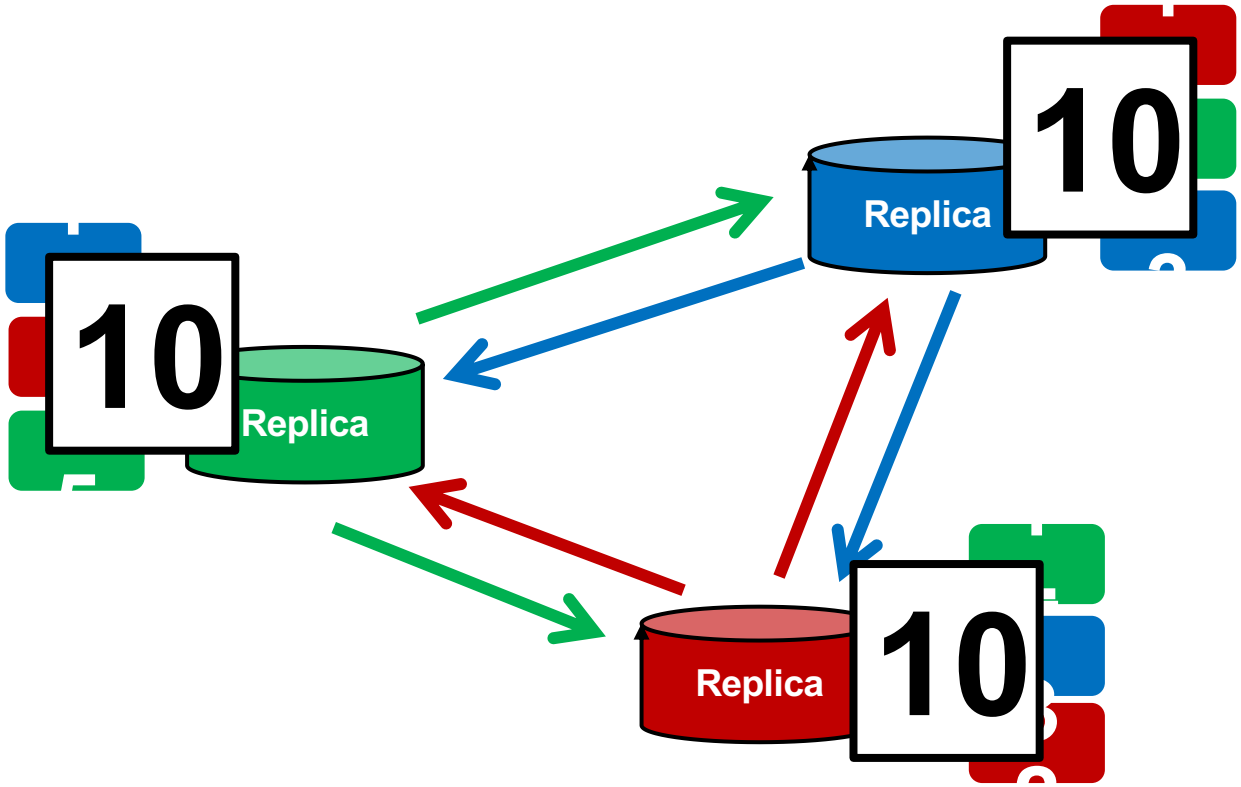
Eventual Consistency: CRDTs



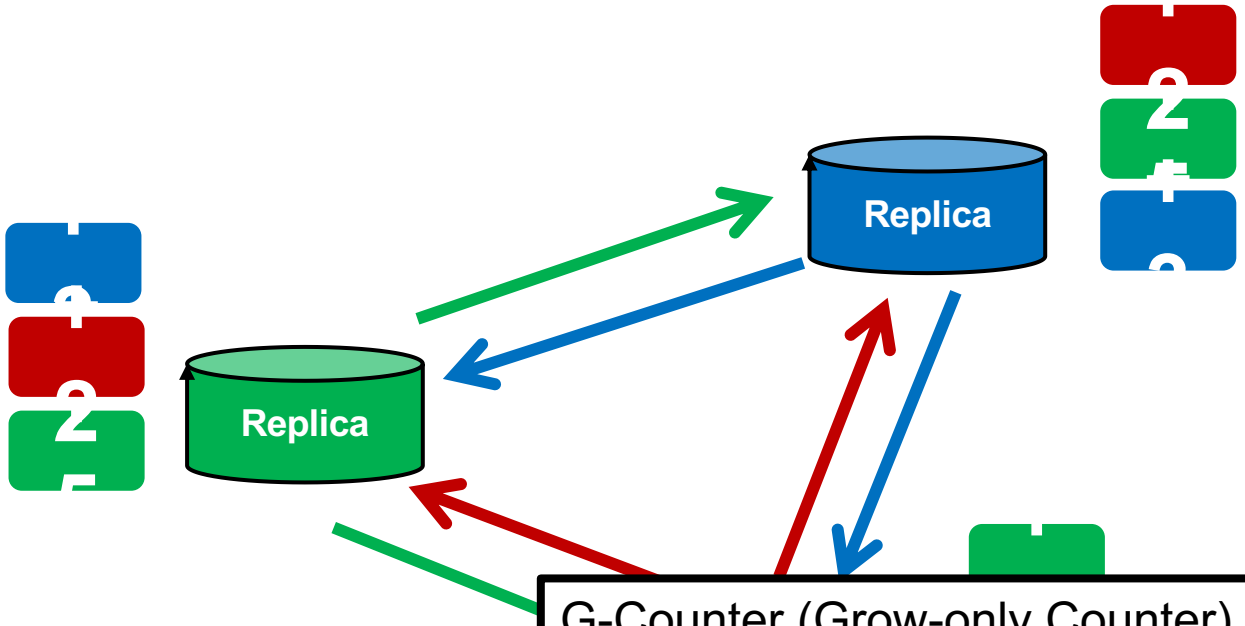
Eventual Consistency: CRDTs



Eventual Consistency: CRDTs

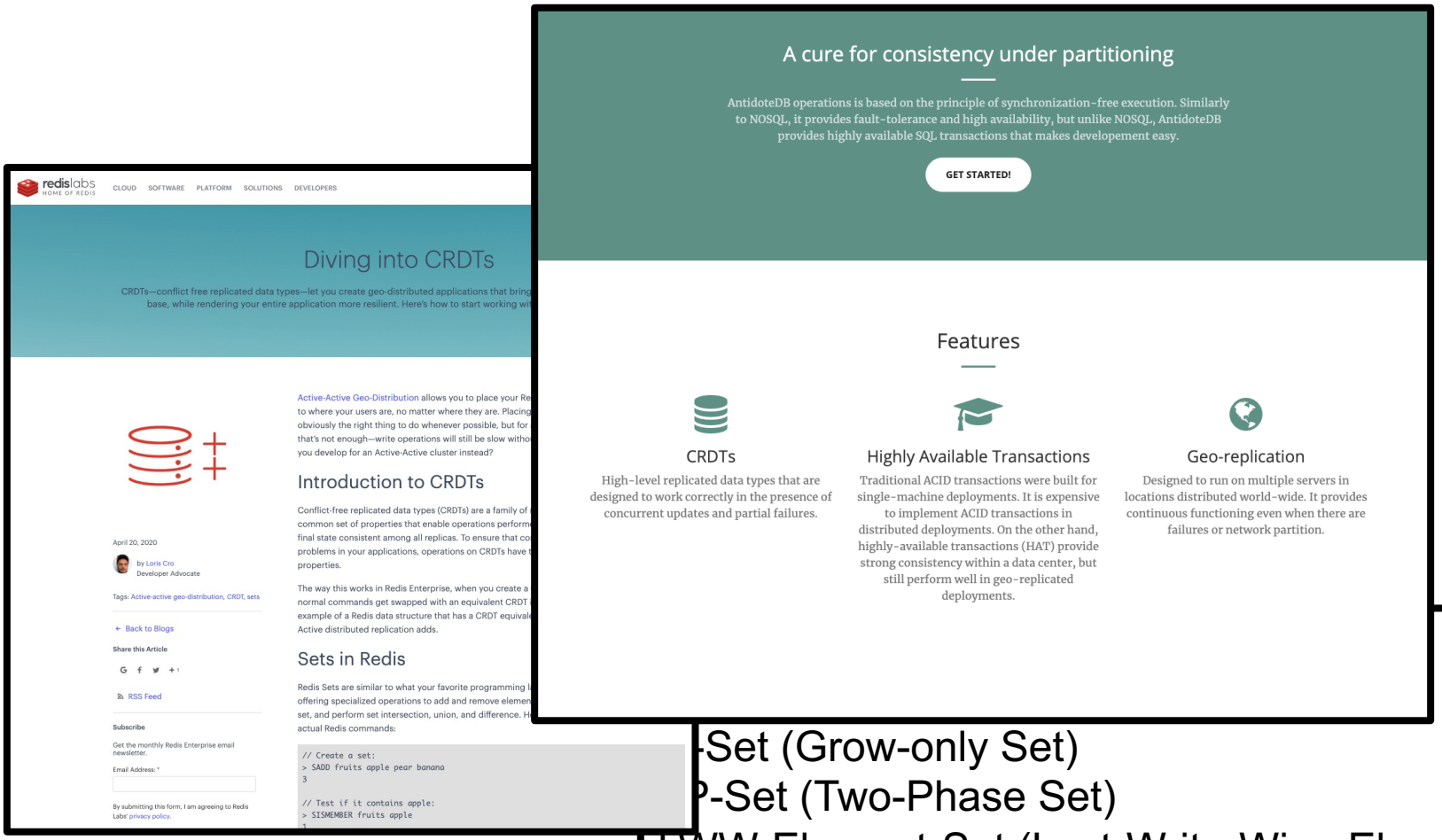


Eventual Consistency: CRDTs



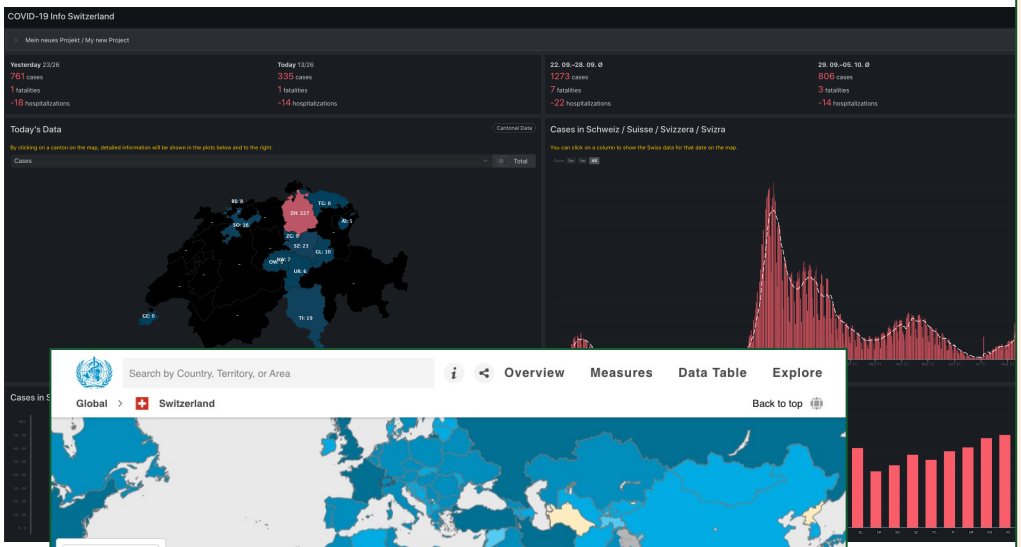
G-Counter (Grow-only Counter)
PN-Counter (Positive-Negative Counter)
G-Set (Grow-only Set)
2P-Set (Two-Phase Set)
LWW-Element-Set (Last-Write-Wins-Element-Set)
…

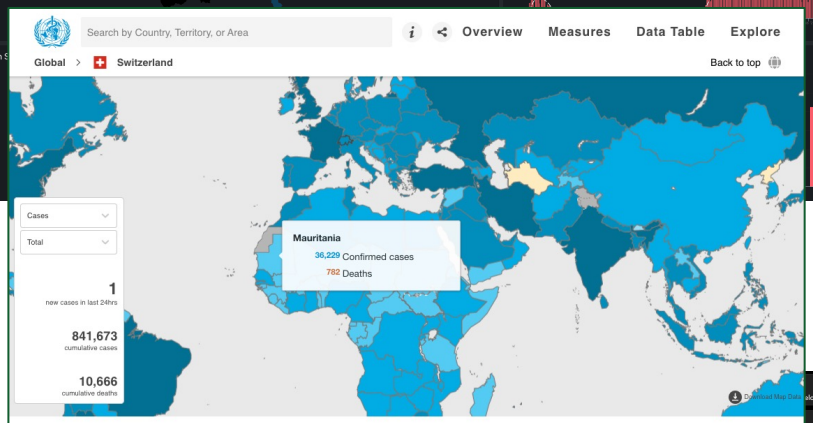
Eventual Consistency: CRDTs



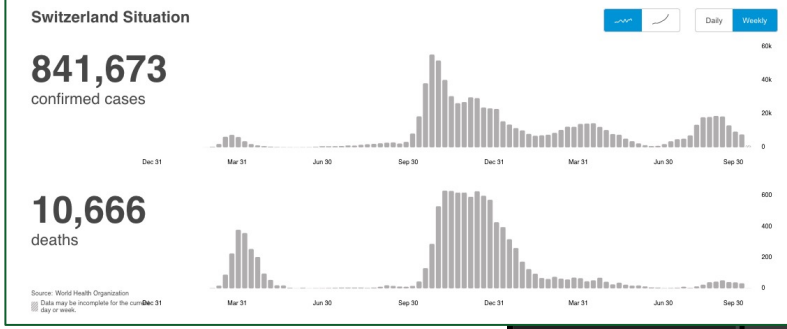
-Set (Grow-only Set)
P-Set (Two-Phase Set)
LWW-Element-Set (Last-Write-Wins-Element-Set)
…

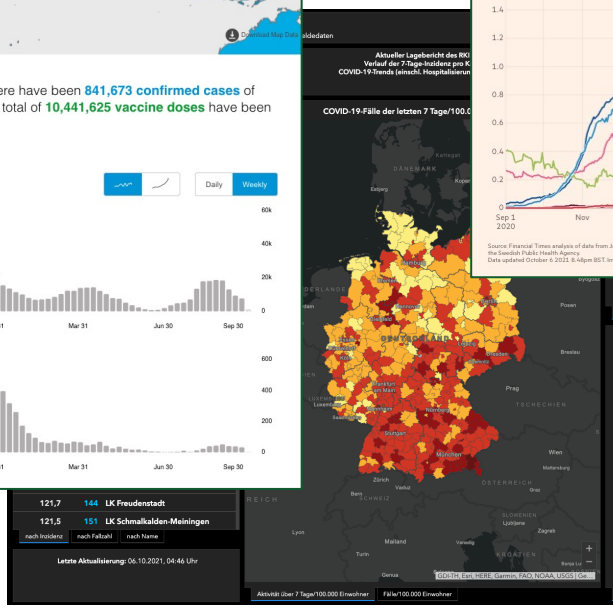
WHAT’S NEXT?

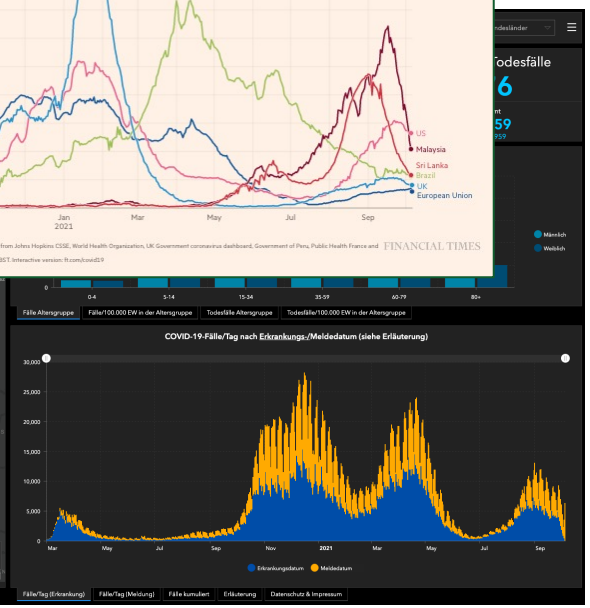


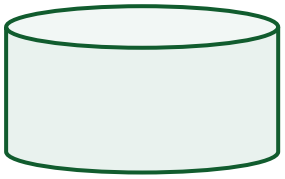


In **Switzerland**, from **3 January 2020** to **8:48pm CEST, 6 October 2021**, there have been **841,673 confirmed cases** of COVID-19 with **10,666 deaths**, reported to WHO. As of **30 September 2021**, a total of **10,441,625 vaccine doses** have been administered.





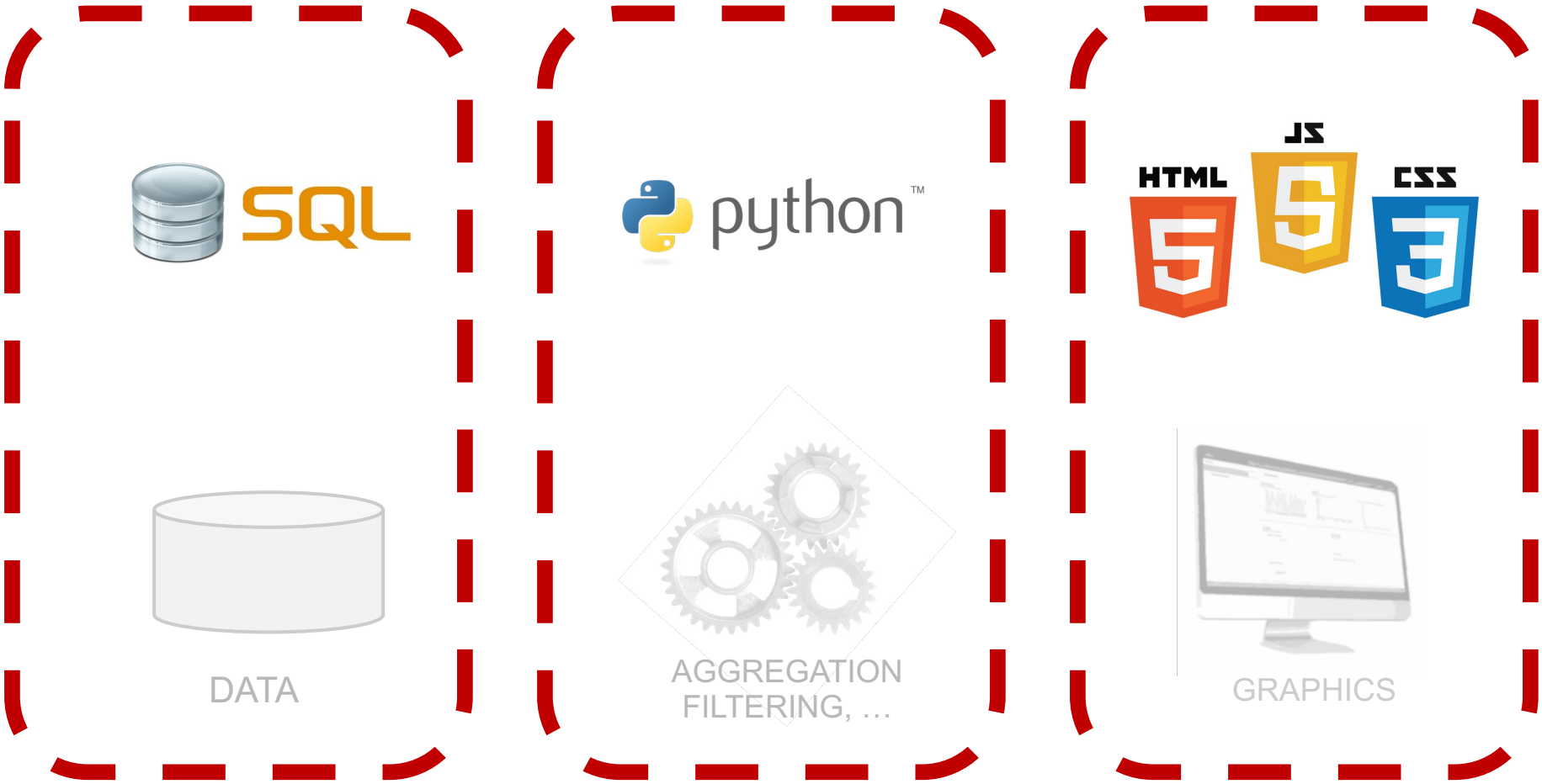




DATA

AGGREGATION
FILTERING, …

GRAPHICS


SQL
python
HTML
JS
CSS
DATA
AGGREGATION
FILTERING, …
GRAPHICS

GLOBAL PROGRAMMING

DATA

AGGREGATION
FILTERING, …

GRAPHICS

Secure Enclave Programming

Enclaves: hardware protection for data and code from system software

- ~250MB memory
- Native speed

Here to stay

- Google Cloud announces to offer enclaves to customers (Summer ‘20)
- Intel announces 3rd Gen Xeon Processors: enclaves up to 1TB (Oct ‘20)

enclave.edl

```
enclave {
    trusted {
      public void checkPassword([in, size=len] char* guess,
            [out] int* result, size_t len);
    };
};
```

enclave.cpp

```
const char* password = "secret";
void checkPassword(char* guess, int* result, size_t len) {
    strcmp(guess, password) == 0) ?
       *result = 1 : *result = 0;
}
```

main.cpp

```
#include "sgx_urts.h"
#include "enclave_u.h"
#define BUF_LEN 100
int main() {
    sgx_enclave_id_t eid;
    sgx_status_t ret = SGX_SUCCESS;
    sgx_launch_token_t token = {0};
    int updated = 0;
    ret = sgx_create_enclave(ENCLAVE_FILE, SGX_DEBUG_FLAG,
          &token, &updated, &eid, NULL);
    if (ret != SGX_SUCCESS) { ... /* exception */ }
    char* guess = ... // read guess from stdin
    int result = 0;
    checkPassword(eid, guess, &result, BUF_LEN);
    if (SGX_SUCCESS != sgx_destroy_enclave(eid)) {...}
    return 0;
}
```

C++ - Intel Visual Studio

PasswordChecker.java

```
@Enclave
class PasswordChecker {
    @Secret static String password = "secret";

    @Gateway
    public static boolean checkPassword(String guess) {
       String guessE = endorse(guess);
       return declassify(guessE.equals(password));
} }
```

Main.java

```
class Main {
    public static void main(String[] args) {
       String guess = ... // read guess from stdin
       PasswordChecker.checkPassword(guess);
} }
```

JE*

[A. Oak, A. M. Ahmadian, M. Balliu, G. Salvaneschi, **Language Support for Secure Software Development with Enclaves, CSF'21]**

(Modeling) Infrastructure as Code

Example:

- Create a Web server in Google with TypeScript on Node.js.
- Create the instance, export the IP and Hostname.

Reactivity to external events

Deployment safety?

Deployment patterns?

```
const gcp = require("@pulumi/gcp");

const computeNetwork = new gcp.compute.Network("network", {
    autoCreateSubnetworks: true,
});

const computeFirewall = new gcp.compute.Firewall("firewall", {
    network: computeNetwork.selfLink,
    allows: [{
        protocol: "tcp",
        ports: [ "22", "80" ],
    }],
});

// Create a simple web server.
const startupScript =
    "#!/bin/bash \n" +
    "echo 'Hello, World!' > index.html \n" +
    "nohup python -m SimpleHTTPServer 80 &";

const computeInstance = new gcp.compute.Instance("instance", {
    machineType: "f1-micro",
    metadataStartupScript: startupScript,
    bootDisk: {
        initializeParams: {
            image: "debian-cloud/debian-8",
        },
    },
    networkInterfaces: [{
        network: computeNetwork.id,
        accessConfigs: [{}], // must be empty
    }],
    serviceAccount: {
      scopes: ["https://www.googleapis.com/auth/cloud-platform"],
    },
}, { dependsOn: [computeFirewall] });

exports.instanceName = computeInstance.name;
exports.instanceIP = computeInstance.networkInterfaces.apply(ni => ni[0
```

[**Automating Serverless Deployments for DevOps Organizations**, D. Sokolowski, P. Weisenburger, G. Salvaneschi, FSE'21]

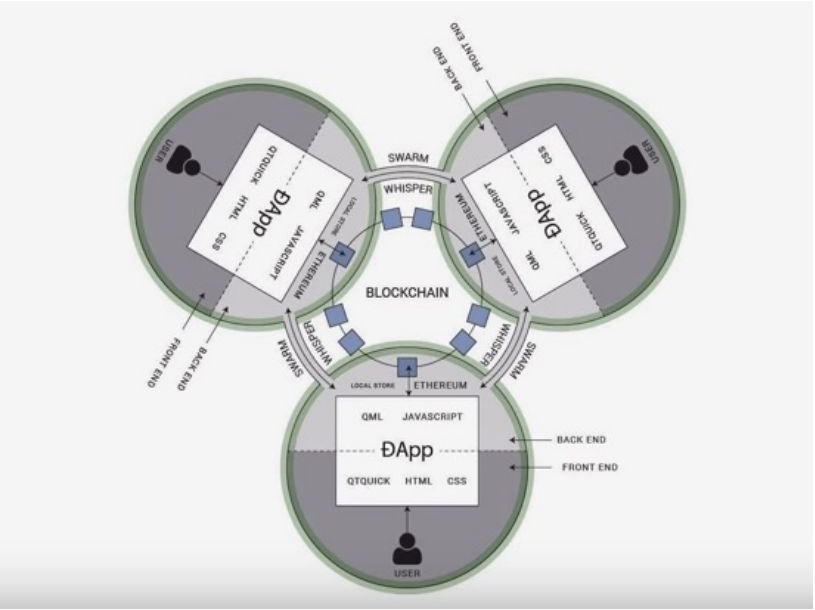
Smart Contracts

Single language for clients and contract

- Client compiles to JavaScript
- Contract compiles to Solidity

Multitier programming?

```
val gui on Client
val score on Contract
```










Thank you

THANKS!

@guidosalva
www.guidosalvaneschi.com

Barbara Liskov
Carlo Ghezzi
Mira Mezini
Andrew Myers
Patrick Eugster
Alessandro Margara
Nada Amin

Philipp Haller
Ragnar Mogk
Sebastian Erdweg
Joscha Drechsler
Sven Amann
Ralf Mitschke
Sebastian Proksch
Sarah Nadi