

Design and Implementation of Modern Programming Languages (Seminar)

Contact

By mail:

- Ragnar Mogk
 - mogk@cs.tu-darmstadt.de
- Pascal Weisenburger
 - weisenburger@st.informatik.tu-darmstadt.de

Timeline

- topic selection: 8.11 (Sunday after kick-off)
- paper submission: 18.12
- peer review submission: 15.01
- feedback meetings: 18.01-22.01
- camera ready: 29.01
- presentation day: **12.02**

Start Early!

- **Please let us know if you have a conflict with the presentation day**

The Seminar

- Start with several papers of a specific topic
- Understand the general area
- Find more references to complete your understanding
- Write your own paper about in that area
 - Tell a coherent story about one aspect of the topic.
 - Please present technologies, with your own words and your own examples.
E.g. : the same running example through the paper

What you learn

- Introduce students to the core techniques of scientific work
 - Process of writing and publishing research papers
 - Review a paper
 - Give a scientific talk
- Learn an advanced topic on PL
 - Compare and evaluate existing solutions

Reviews

- A summary of the paper
 - To demonstrate that you read and understood
- Suggestions for the author, regarding both content and presentation
 - Be realistic and concrete
- A list of the major positive and negative issues
 - Honest! mainly used to grade the reviewer, not the paper

Grades

- Find the details and more info of how to write the paper on the website
- Paper 40 %
- Talk 40 %
- Reviews 20 %
- Once more. Tell a nice story. Compare things. Find advantages and limitations. Provide examples in different cases

Implementation of Programming Languages (Project)

Intro

- Goal: Implement a software artifact supporting current research.
- General scope: programming languages:
 - Extensions, code generators
 - Tools, IDEs
 - Analysis techniques
 - Performance assessments
 - Prototypes with innovative abstractions

Process

- Project details are specified by discussing with your supervisor.
 - Periodic discussions/meetings.
- Allocate one day per week to work on the project
 - Send us an email about the weekly progress.
 - Not an issue If you have no time for 1 or 2 weeks
 - Don't abuse this.
- Presentation and delivery of the artifact and the documentation: end of the semester.

Guidelines and Suggestions

- Deliver RUNNING code
- Deliver some documentation
- Optional: final presentation / demo to discuss your work
- Team work is good!
 - Organize a team even with people you don't know (yet)

Grades

- How much of the specification was implemented?
- The code has to run
- What is the overall quality of the code?
 - E.g. is it painful to modify/extend it?
 - Is it self-documenting?
- What is the quality of the “documentation”?

SEMINAR TOPICS

Programming Languages for Smart Contracts

"an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way."

- How to write smart contracts?
- Need a (specification?) language...

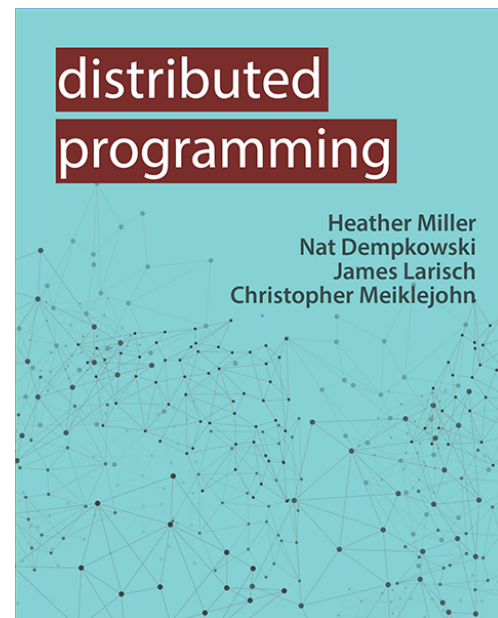


Some examples

- "Functional Smart Contracts"
 - Chakravarty et al. 2019. Functional Blockchain Contracts.
 - Seijas and Thompson. 2018. Marlowe: Financial Contracts on Blockchain. ISoLA.
- Object-Oriented Smart Contracts
 - Coblenz. Obsidian: a safer blockchain programming language. ICSE 2017.
 - Coblenz, et al. 2019. User-Centered Programming Language Design in the Obsidian Smart Contract Language.
 - Coblenz, et al. 2019. Obsidian: Typestate and Assets for Safer Blockchain Programming
 - Schrans et al. Writing safe smart contracts in Flint. Programming 2018

Programming models for Distributed Computing

- Very popular in the '80, early '90
- A lot of ideas still influential
 - RMI, CORBA, Argus, Emerald, Akka, Linda
- Survey of the most important approaches



Information Flow in Languages for Distributed Systems

- Distributed systems can leak private data
- Information flow type systems can prevent it
 - Attach security labels to values
 - Reject programs in case of violation
- Current Research topics
 - Interaction with distribution
 - Boundaries with databases
 - Big data
 - ...

```
var l, h  
if h = true  
then  
    l := 3  
else  
    l := 42
```

Macros and Metaprogramming

- tools and ideas on how to program interpreters and compilers
- survey on history of macro systems
- what features should modern languages support?

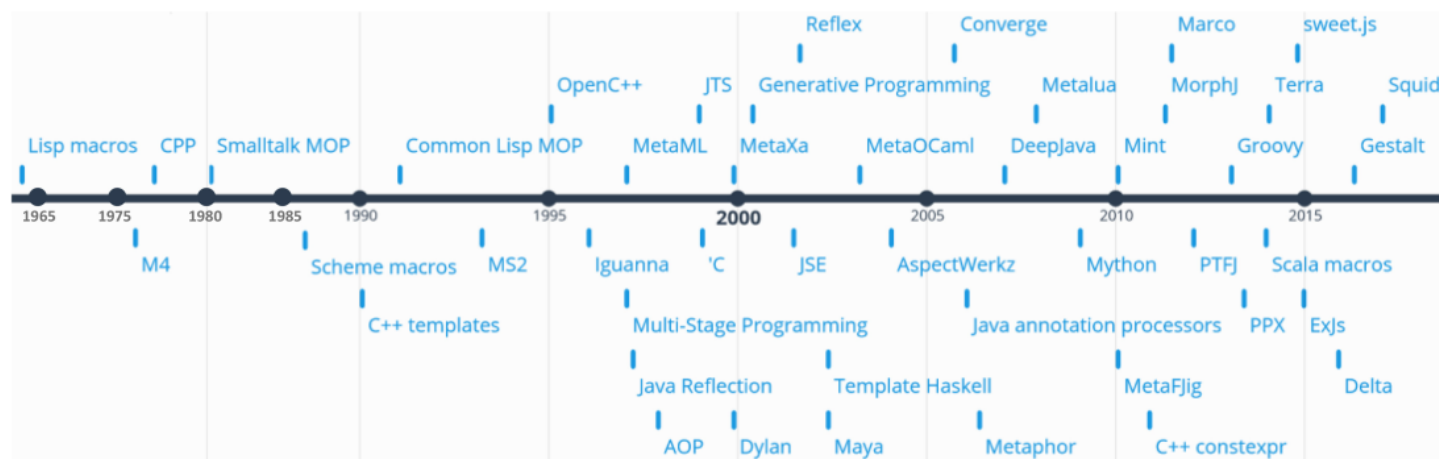
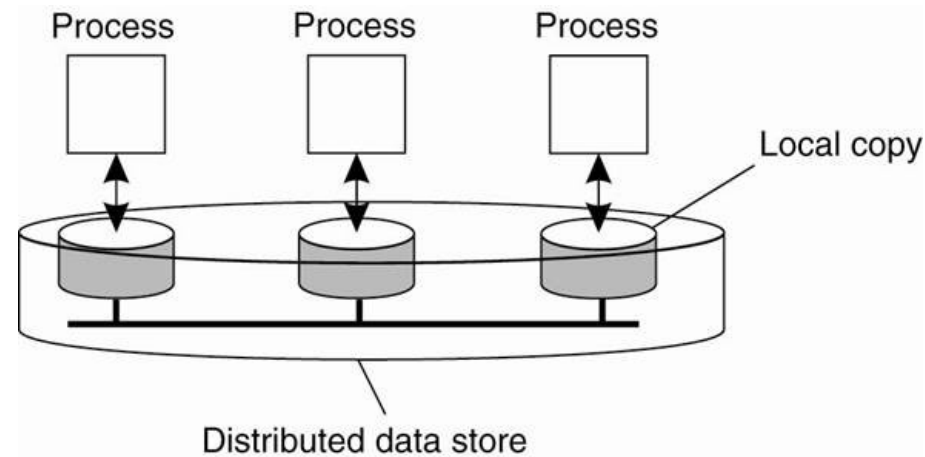


Fig. 1. Timeline of metaprogramming languages and systems; the list is not exhaustive.

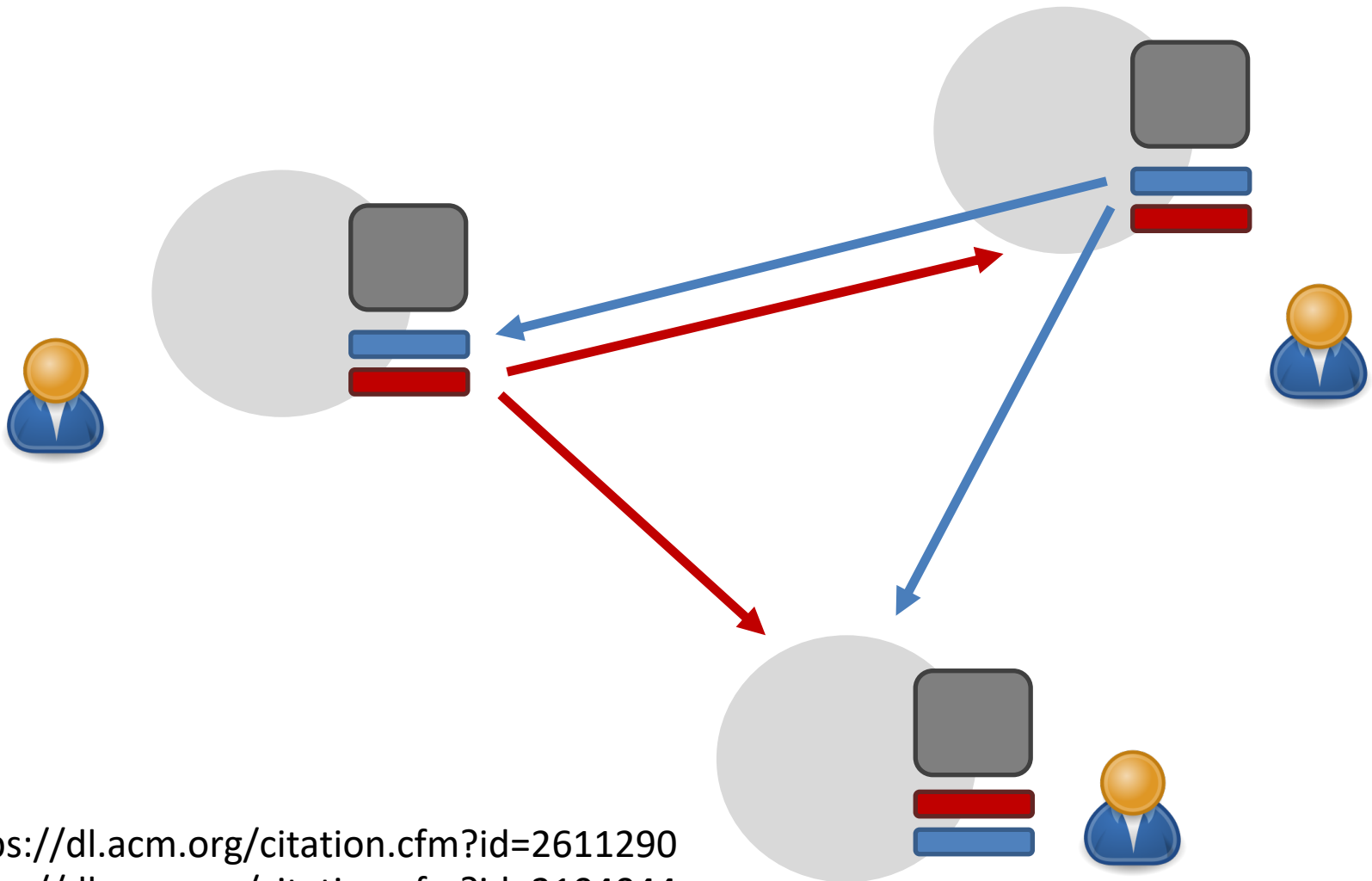
TOPICS

Consistency in PL

- Distributed systems adopt a consistency level
 - Eventual consistency
 - Causal consistency
 - ...
- So far, considered only in the middleware



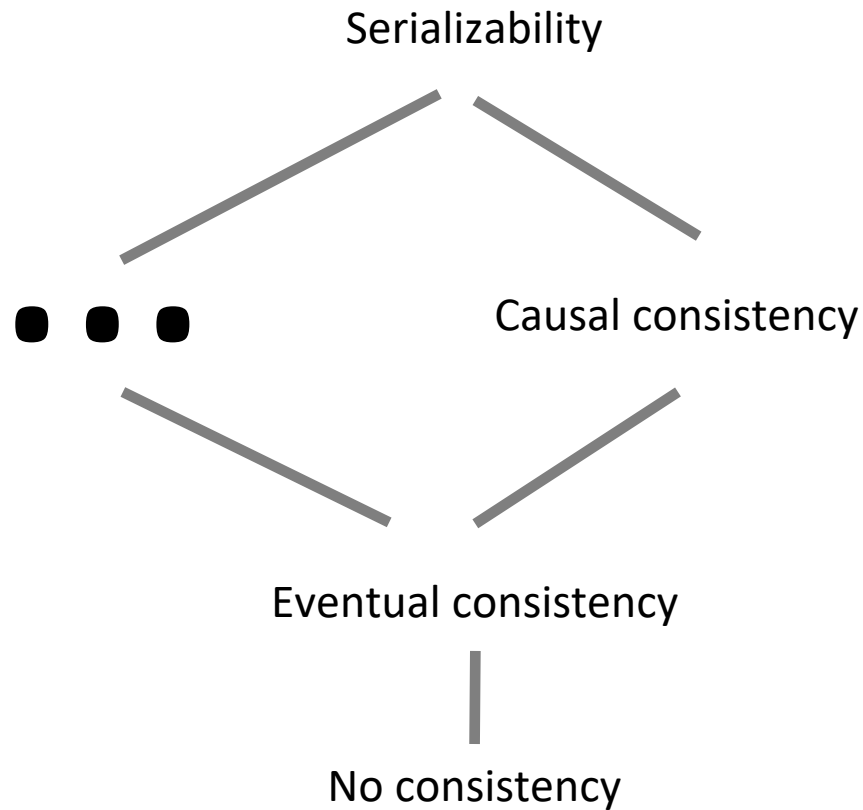
Consistency in Distributed Systems



<https://dl.acm.org/citation.cfm?id=2611290>

<https://dl.acm.org/citation.cfm?id=3104044>

Consistency Levels form a Lattice



Eventual consistency
Causal consistency
Fifo consistency

....

DATASTAX

GLOSSARYSUPPORTDEVELOPER BLOGS

Apache Cassandra™ 2.1 (Supported)

Cassandra 2.1 (used by DSE 4.7, 4.8)

About Cassandra

What's new

CQL

Understanding the architecture

Planning a deployment

Installing

Initializing a cluster

Security

Database internals

Configuration

Operations

Backing up and restoring data

Cassandra tools

References

Moving data to/from other databases

Troubleshooting

Release notes

Search

Advanced search

Home / Database internals / Data consistency / Configuring data consistency

Configuring data consistency

Consistency levels in Cassandra can be configured to manage availability versus data accuracy. You can configure consistency on a cluster, datacenter, or individual I/O operation basis. Consistency among participating nodes can be set globally and also controlled on a per-operation basis (for example insert or update) using Cassandra's drivers and client libraries.

Write consistency levels

This table describes the write consistency levels in strongest-to-weakest order.

Write Consistency Levels

Level	Description	Usage
ALL	A write must be written to the commit log and memtable on all replica nodes in the cluster for that partition.	Provides the highest consistency and the lowest availability of any other level.
EACH_QUORUM	Strong consistency. A write must be written to the commit log and memtable on a quorum of replica nodes in <i>all</i> datacenter.	Used in multiple datacenter clusters to strictly maintain consistency at the same level in each datacenter. For example, choose this level if you want a read to fail when a datacenter is

Menuamazon web services

EnglishSign In to the Console

Amazon DynamoDBDeveloper Guide (API Version 2012-08-10)

Documentation - This Guide

Search

Amazon DynamoDB

AWS Documentation » Amazon DynamoDB » Developer Guide » What Is Amazon DynamoDB? » Amazon DynamoDB: How It Works » Read Consistency

Read Consistency

Amazon DynamoDB is available in multiple AWS regions around the world. Each region is completely independent and isolated from other AWS regions. For example, if you have a table called `People` in the `us-east-1` region and another table named `People` in the `us-west-2` region, these are considered two entirely separate tables. For a list of all the AWS regions in which DynamoDB is available, see [AWS Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Every AWS region consists of multiple distinct locations called Availability Zones. Each Availability Zone is isolated from failures in other Availability Zones, and to provide inexpensive, low-latency network connectivity to other Availability Zones in the same region. This allows rapid replication of your data among multiple Availability Zones in a region.

When your application writes data to a DynamoDB table and receives an HTTP 200 response (OK), all copies of the data are updated. The data will eventually be consistent across all storage locations, usually within one second or less.

DynamoDB supports *eventually consistent* and *strongly consistent* reads.

Eventually Consistent Reads

When you read data from a DynamoDB table, the response might not reflect the results of a recently completed write operation. The response might include some stale data. If you repeat your read request after a short time, the response should return the latest data.

Strongly Consistent Reads

When you request a strongly consistent read, DynamoDB returns a response with the most up-to-date data, reflecting the updates from all prior write operations that were successful. A strongly consistent read might not be available in the case of a network delay or outage.

Note

DynamoDB uses eventually consistent reads, unless you specify otherwise. Read

mongoDB | FOR GIANT IDEAS

How does MongoDB ensure consistency?

Back to Table of Contents

MongoDB is consistent by default: reads and writes are issued to the primary member of a replica set. Applications can optionally read from secondary replicas, where data is eventually consistent by default. Reads from secondaries can be useful in scenarios where it is acceptable for data to be slightly out of date, such as some reporting applications. Applications can also read from the closest copy of the data (as measured by ping distance) when latency is more important than consistency.

Learn more in the MongoDB [Architecture Guide](#).

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



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 mixed 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-oriented programming

Consistency is fully integrated with object-oriented abstractions.

Overview

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

Distribution

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

Extending ConSysT

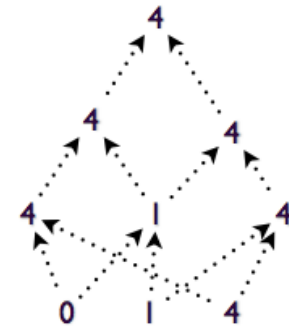
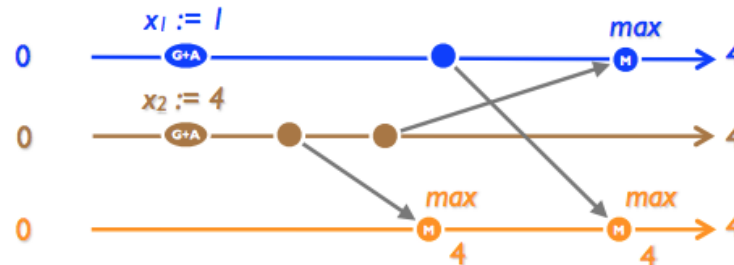
- Consistency can be associated to types e.g., High[Int]
 - Design consistency for operations, e.g., shopCheckout()
- Language support for multiple consistency levels on top of existing middleware, e.g., Cassandra

CRDTs

A data structure which can be replicated across multiple computers where replicas can be updated independently and concurrently without coordination, and where it is always mathematically possible to resolve inconsistencies

Example: collaborative editing

CRDT: Grow-only counter



Applications

- Redis: a distributed, highly available and scalable in-memory database
- Riak: a distributed NoSQL key-value data store based on CRDTs
- Facebook implements CRDTs in their Apollo low-latency "consistency at scale" database

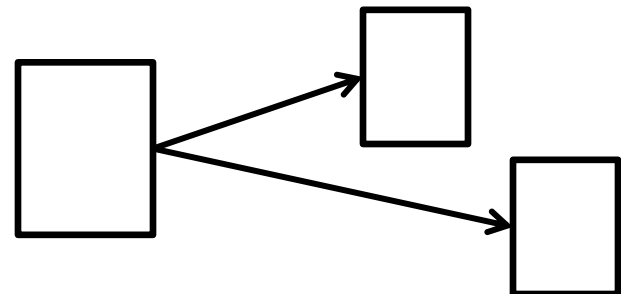
Delta CRDTs

- Implementation of CRDTs in Scala
- Delta replication for efficiency
- Composability of CRDTs.
How to derive new CRDTs by composition?

Tierless/Distributed Languages

Tierless Languages

- Traditional development of Web applications
 - Server side (e.g. servlet, php script, ..)
 - Client side (Javascript, ...)
- Tierless languages unify the development of server-side and client-side components
 - Network communication is hidden
 - The compiler automatically generates the code for the server and for the client



Placement Types

```
trait Registry extends Peer  
trait Node extends Peer
```

Peers

```
val message: Event[String] Registry  
= placed { getMessageStream() }
```

Placement Types



ScalaLoci

Research and development of
language abstractions for
distributed applications in Scala

Coherent

Implement a cohesive
distributed application in a
single multi-tier language

Comprehensive

Freely express any
distributed architecture

Safe

Enjoy static type-safety
across components

1



Specify Architecture

Define the architectural relation of
the components of the distributed
system

```
trait Server extends Peer {  
  type Tie = Multiple[Client]  
}  
  
trait Client extends Peer {  
  type Tie = Single[Server]  
}
```

2



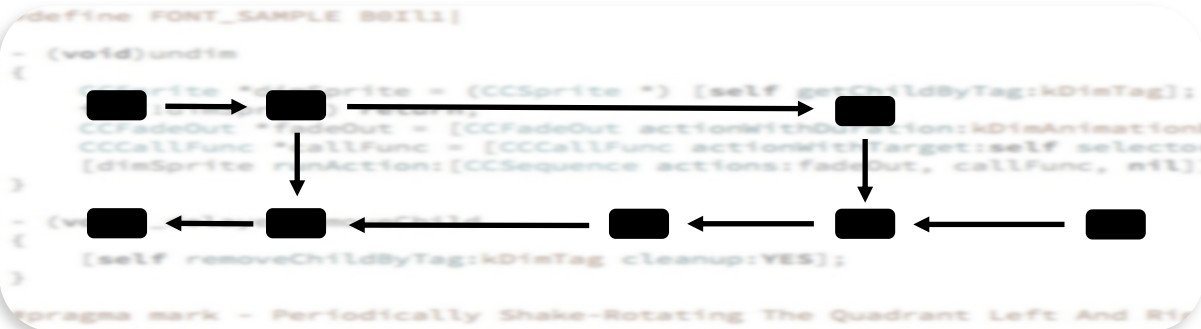
Specify Placement

Control where data is located and
computations are executed

```
val items = placed[Server] {  
  getCurrentItems()  
}  
  
val ui = placed[Client] {  
  new UI  
}
```

www.scala-loci.github.io

Dynamic placement



Data streams
cross Tier
boundaries

Compiler
splits the
code

Application
deployment

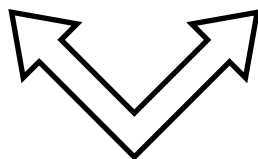
```
define FONT_SAMPLE B0111]
- (void)undin
{
    CCSprite *fadeSprite = (CCSprite *) [self getChildByTag:kDinTag];
    if (!fadeSprite) return;
    CCFadeOut *fadeOut = [CCFadeOut actionWithDuration:kDinAnimation];
    CCCallFunc *callFunc = [CCCallFunc actionWithTarget:self selector:
    [fadeSprite runAction:[CCSequence actions:fadeOut, callFunc, nil];
}
- (void)delayedRemoveChild
{
    [self removeChildByTag:kDinTag cleanup:YES];
}
#pragma mark - Periodically Shake-Rotating The Quadrant Left And Right
```



```
define FONT_SAMPLE B0111]
- (void)undin
{
    CCSprite *fadeSprite = (CCSprite *) [self getChildByTag:kDinTag];
    if (!fadeSprite) return;
    CCFadeOut *fadeOut = [CCFadeOut actionWithDuration:kDinAnimation];
    CCCallFunc *callFunc = [CCCallFunc actionWithTarget:self selector:
    [fadeSprite runAction:[CCSequence actions:fadeOut, callFunc, nil];
}
- (void)delayedRemoveChild
{
    [self removeChildByTag:kDinTag cleanup:YES];
}
#pragma mark - Periodically Shake-Rotating The Quadrant Left And Right
```

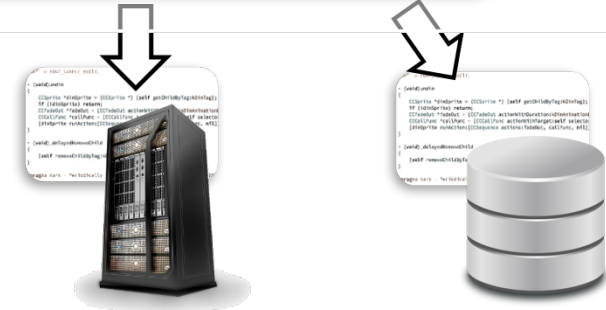
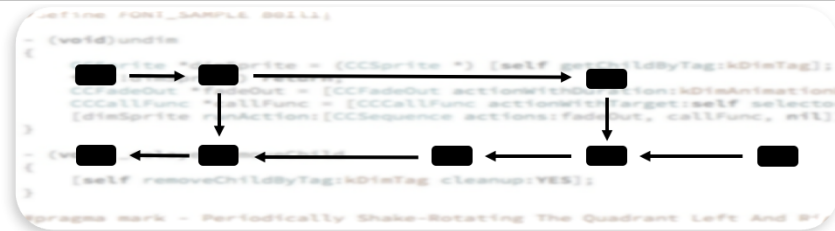


```
define FONT_SAMPLE B0111]
- (void)undin
{
    CCSprite *fadeSprite = (CCSprite *) [self getChildByTag:kDinTag];
    if (!fadeSprite) return;
    CCFadeOut *fadeOut = [CCFadeOut actionWithDuration:kDinAnimation];
    CCCallFunc *callFunc = [CCCallFunc actionWithTarget:self selector:
    [fadeSprite runAction:[CCSequence actions:fadeOut, callFunc, nil];
}
- (void)delayedRemoveChild
{
    [self removeChildByTag:kDinTag cleanup:YES];
}
#pragma mark - Periodically Shake-Rotating The Quadrant Left And Right
```



Dynamic
Placement

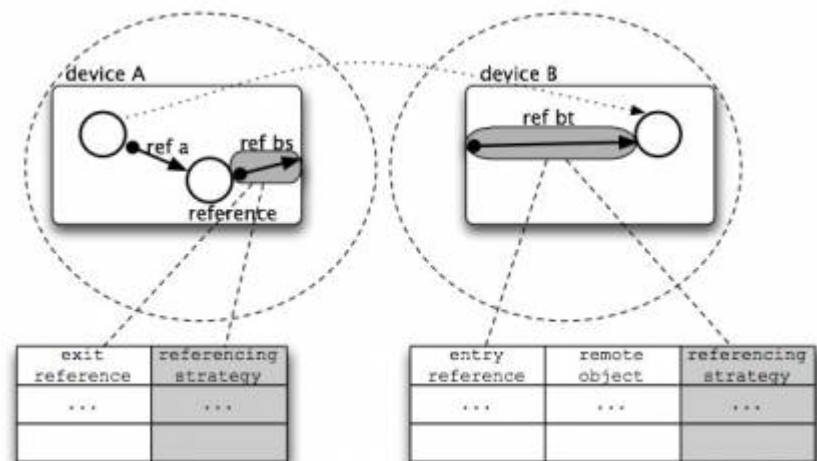
IoT & Edge: Drones



Dynamic Placement

Distributed Garbage Collection

- Expensive in the general case!
- Use a type system to track ownership
 - The type system guarantees that there are no other refs to an object
 - The object can be deleted

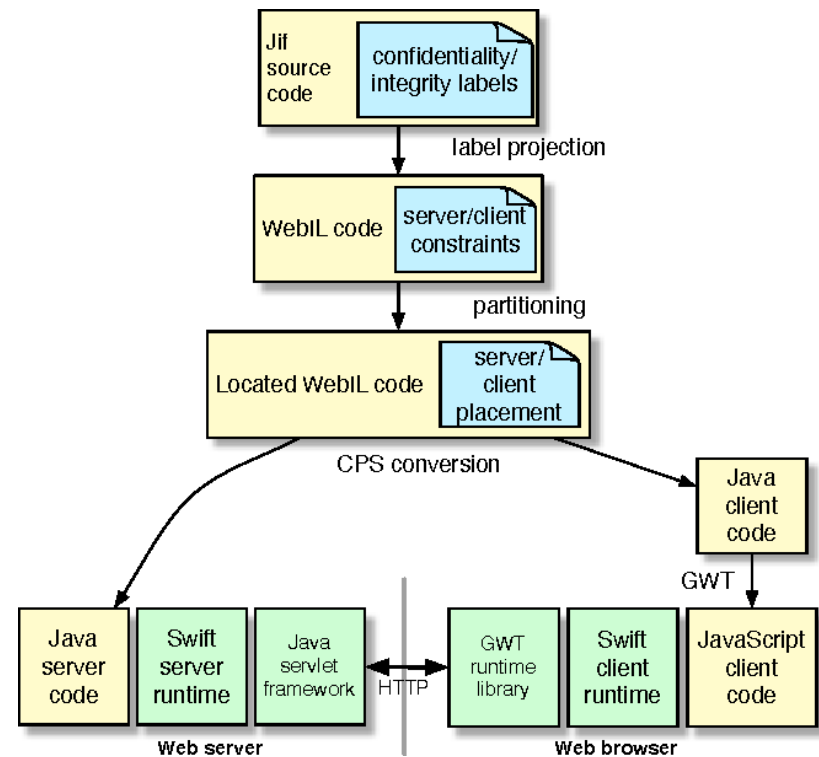


Secure Tierless Languages

- Place functionalities based on **privacy** requirements
 - Function $f(x)$ may run on the server or on the client
 - Decide based on the privacy of x

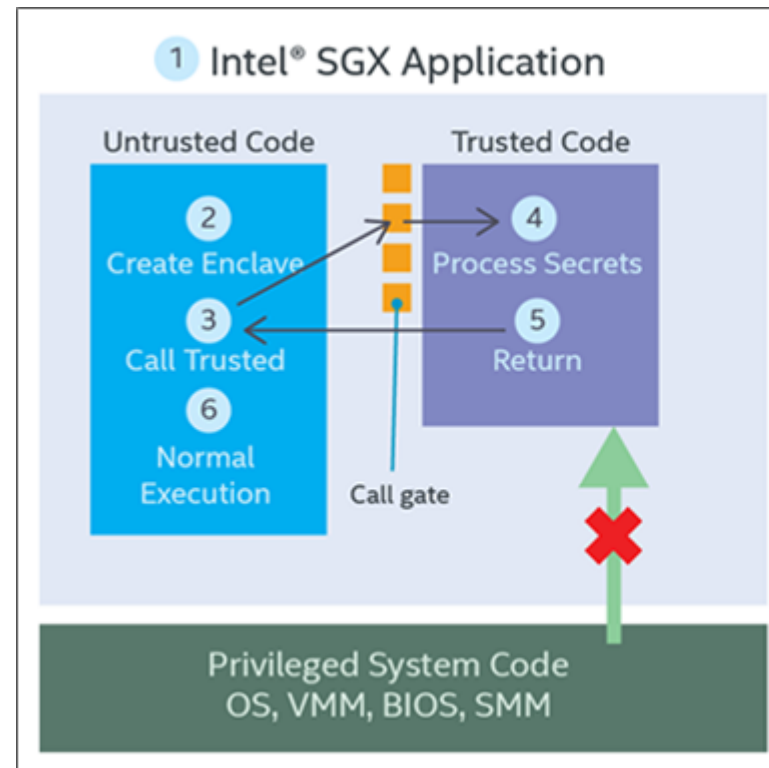
Challenges:

- Information flow analysis
- Decision at runtime?



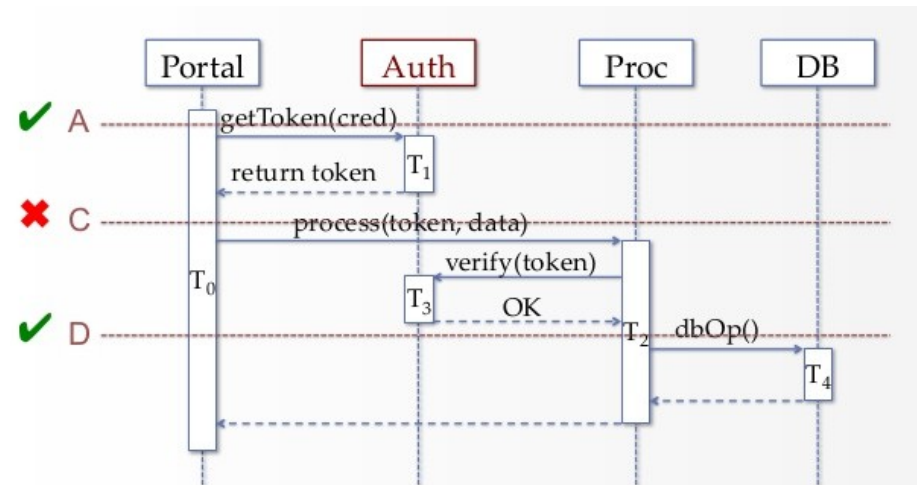
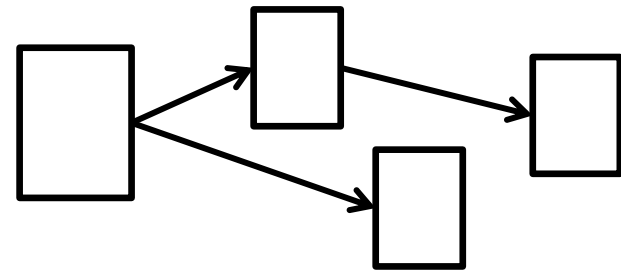
Secure Tiers

- Intel Software Guard Extensions (SGX)
- Code running inside secure enclave can be seen as a separate tier
- Automate secure deployment



Dynamic Software Updates

- Complex component-based systems require updates without downtime
- When is an update safe?
- Develop a system for safe dynamic updates



Distributed Reactive Programming

Reactive Programming

REScala

REScala is a Scala library for functional reactive programming on the JVM and the Web. It provides a rich API for event stream transformations and signal composition with managed consistent up-to-date state and minimal syntactic overhead. It supports concurrent and distributed programs.

Flexible

Abstractions for Events and Signals. Integrating with imperative, object-oriented, functional and any other paradigm on the JVM.

Consistent

No temporary inconsistencies, no data races, no surprises. Write code which behaves as expected.

Thread-safe

Multi-threaded applications are fully supported. Reactive abstractions can be safely accessed from any thread and they are updated in parallel.

Visit the [manual](#) to get started.

Project Description

Software applications react to external changes such as the input from the user and network messages. Traditionally, object-oriented software adopts the Observer pattern to implement reactivity and decouple the observers from the observables. Whereas researchers have highlighted the drawbacks of this style for a long time, alternatives struggle to be widely accepted. In particular, functional reactive programming and dataflow programming – which aim to represent time-changing values as first class abstractions – are promising, but hardly escape the functional setting. On the other hand, event-based languages directly support events but do not achieve the declarative style of more functional approaches.

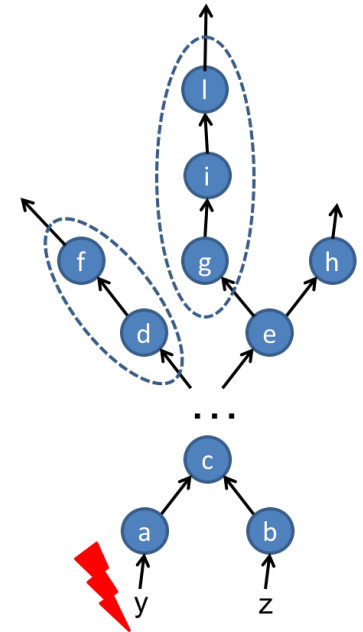
REScala is a reactive language which integrates concepts from event-based and functional-reactive programming into the object-oriented world. Rescala supports the development of reactive applications by fostering a functional and declarative style which complements the advantages of object-oriented design.

Contributors

Project lead:

- Mira Mezini
- Guido Salvaneschi

```
List<String> myList = Arrays.asList("a1", "a2", "b1", "c2", "c1");  
myList.stream()  
    .filter(s -> s.startsWith("c"))  
    .map(String::toUpperCase)  
    .sorted()  
    .forEach(System.out::println);
```



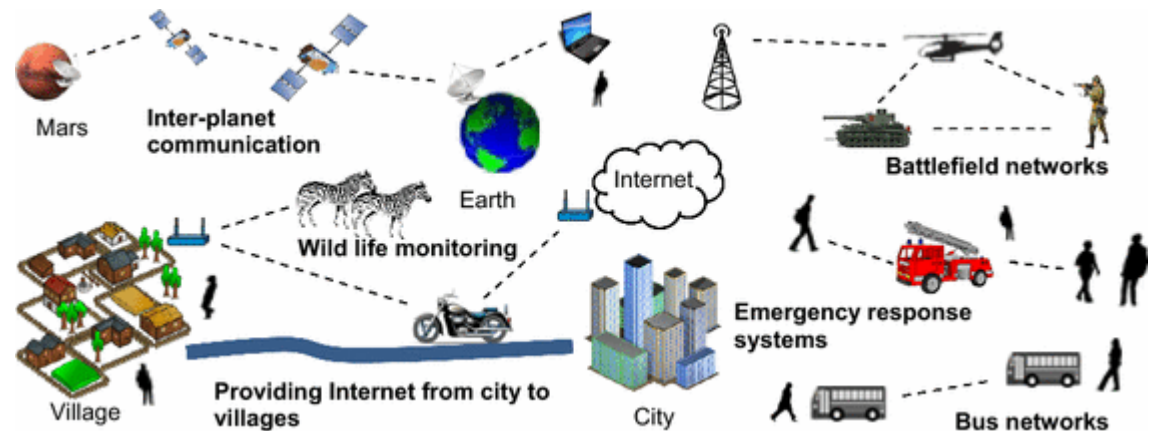
www.rescala-lang.com

IoT & Edge: Distributed Dataflow

- Manage dataflow graph of many devices
- Use device specific effects for inputs & outputs
- Efficient compilation
- Failure modes? Dynamic placement? CRDTs?

Delay Tolerant Routing

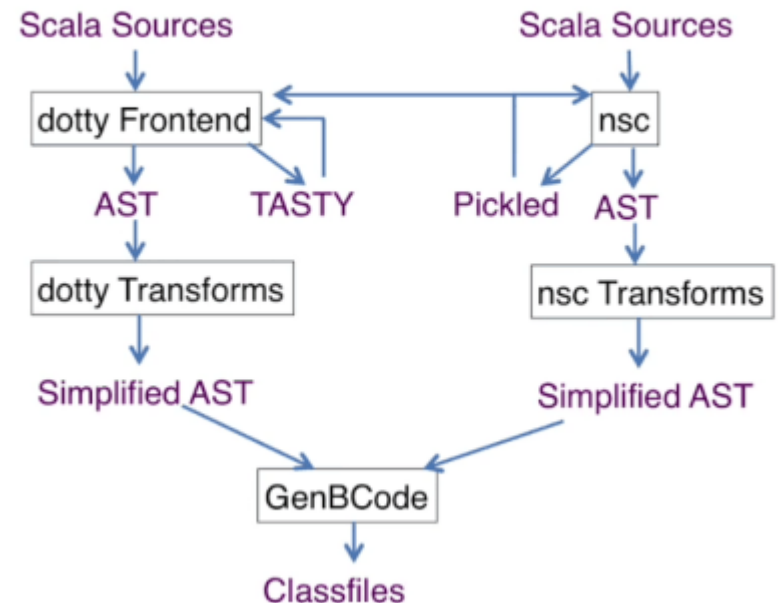
- Some networks have long delays
 - Inter Planet!
 - Hand USB sticks to your friends
 - IP Datagrams on Avian Carriers (RFC 1149)
- Special routing protocols exists
- Integrate with dataflow



Staged Dataflow Graph

- dotty (new scala version) has a staged macro system
- can this be used to implement dataflow graphs during compile time?

dotty Architecture



QUESTIONS?