Implementation of Programming Languages (Project)
Outline

- Administrivia
- Intro
- Schedule
- Topics
GENERAL INFORMATION
Intro

• Implement a software artifact according to a given specification.

• General scope: programming languages:
  – Extensions, code generators
  – Tools, IDEs
  – Analysis techniques
  – Performance assessments
  – Prototypes with innovative abstractions
Timeline

• Periodic discussions/meetings.

• Presentation and delivery of the artifact and the documentation: end of the semester.
Weekly progress

• Allocate one day per week to work on the project
• Send me an email about the weekly progress.
• Not an issue If you have no time for 1 or 2 weeks
  – E.g., exams.
  – Send me an email anyway.
  – Don’t abuse this.
• If I don’t hear from you for a while* the project is canceled

* Potentially any measure greater than a week, see above
Guidelines and Suggestions

• Depending on the goal of the project

• In general:
  – Deliver RUNNING code
  – Deliver some documentation
  – Optional: final presentation / demo to discuss your work

• Team working is good!
  – Organize a team even with people you don’t know (yet)
Grades

• Depending on the project
• Some guidelines:
  – How much of the specification was implemented?
  – Did I say that 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?
Where to start

• Send me an email with your name and a list of 3 topics in order of preference.
• Or arrange an appointment:
  – Ask clarifications if needed
  – Discuss possible customizations
  – Discuss the projects in more detail
  – Consider your programming skills
• Topics will be assigned
• Discuss the project in detail with the instructor
TOPICS
Distributed Event System

• Scala domain-specific language for event-based programming
• Distributed system
  – Akka actors
  – Esper correlation engine inside each actor
• Quality of service requirements

Find all stock options whose value increased of at least 30% in the last month

Statement stmt = conn.createStatement();
ResultSet rs = SELECT Lname FROM Customers
WHERE Snum = 2001

https://github.com/pweisenburger/AdaptiveCEP
https://dl.acm.org/citation.cfm?id=3105530
Consistency in PL

• Distributed systems adopt a consistency level
  – Eventual consistency
  – Causal consistency
  – ...

• So far, considered only in the middleware

• Reason on consistency in the type system
  – EC[Int] -> An Int with eventual consistency
  – **Data-flow type system**: different levels do not mix

https://dl.acm.org/citation.cfm?id=2611290
https://dl.acm.org/citation.cfm?id=3104044
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

https://github.com/scala-loci
IoT & Edge: Tierless Languages

Challenges:
• Abstractions for a dist. System within a single unit
• Transparent distribution vs. performance impact
• Type system

Data streams cross Tier boundaries
Compiler splits the code
Application deployment

https://github.com/scala-loci
Deterministic concurrent programming

• Concurrent programming is hard
  – Race conditions -> Bugs
  – Different interleavings -> Hard to reason about

• Deterministic abstractions avoid races by construction and results do not depend on order
  – Example: threads add the same element to a shared map

• Development of a static analysis

https://github.com/phaller/reactive-async
Preventing data leaks in Big Data systems

• Computing systems can leak (sensitive) data.
• In distributed systems such problem is even more significant
• Big data analytics
  – Different teams access the cluster
  – Data can have different levels of confidentiality

• Implement data leaks prevention in Spark
Testing of Big Data applications

- Big data systems: Spark, MapReduce, Flink, ...
- Testing desktop apps? Junit, random testing, ...

- What to do for Big data systems?

- Evaluate effectiveness/improvements of existing testing techniques
Automated Refactoring to RxJava

• RxJava supports combination of (asynchronous) event streams – similar to Java 8 streams
• Need to port a lot of existing software
• Implement a proof of concept tool for automatic refactoring

```java
dataGetFromNetwork()
  .skip(10)
  .take(5)
  .map({ s -> return s + " transformed" })
  .subscribe({ println "onNext => " + it })
```

http://reactivex.io
QUESTIONS?
Reactive Data Structures

• Data structures are fundamental in real programs

• They can be made reactive:
  
  ```python
  list1 = list2.filter(x > 10)
  list2.add(5)
  list2.add(11)
  list1 ??
  ```

• Project: implement a small library of reactive data structures that react to changes in the base structure
  
  – Use out-of-the-box reactive abstractions
Advanced Types for Events Embedding

• Full-fledged type checking requires dependent types
  – Example: \[ E1(\text{String},\text{Int},\text{Int}) \text{ Join } E2(\text{Int},\text{Int}) \text{ on } (E1._3 === E2._1) \]

• The Shapeless Scala library approximates dependent types thanks to the Scala inference system
• Apply the features of Shapeless to an embedded DSL for event-based programming.

• https://github.com/milessabin/shapeless