Programming with weak synchronization models

Guest lecture ID2203

Peter Van Roy
Christopher Meiklejohn
Annette Bieniusa

LIGHTKONE

Overview of the lesson

- Motivation and principles
  - "As easy as strong consistency, as efficient as weak consistency"
  - A sweet spot: Strong Eventual Consistency
- Convergent data structures
  - Conflict-free replicated data types (CRDTs)
- Lasp
  - Programming language and platform based on composing CRDTs
- Antidote
  - Causal transactional database based on CRDTs

LIGHTKONE

Guest lecturers

- This lesson is brought to you by:
  - Peter Van Roy, Université catholique de Louvain
  - Christopher Meiklejohn, Université catholique de Louvain
  - Annette Bieniusa, Technische Universität Kaiserslautern



- This research is being done in two European projects:

SyncFree — **EU FP7 2013-2016**

LightKone — **EU H2020 2017-2019**

LightKone

Both easy and efficient

- One of the holy grails of distributed systems is to make them both easy to program and efficient to execute
- Strong consistency (linearizability) is easy to program but inefficient
- Eventual consistency (operations eventually complete) is efficient to execute but hard to program
- Can we get the best of both worlds?
  - Synchronization-free programming aims to combine the ease of strong consistency with the efficiency of eventual consistency
  - How can this work?

LIGHTKONE

Back to basics

- Distributed system = a collection of networked computing nodes that behaves like one system (= consistency model)
- To make this work, the nodes will coordinate with each other according to well-defined rules (= synchronization algorithm)
- For example, a reliable broadcast algorithm guarantees the all-or-none property: all correct nodes deliver, or none do

LIGHTKONE

How far can we go?

- We would like the consistency model to be as strong as possible (easy to program) and the synchronization algorithm to be as weak as possible (efficient to execute)
- Let's try the extreme case: the weakest possible synchronization is no synchronization (no rules), which enforces no consistency at all!
  - So it's clear we need *some* synchronization
  - How little can we get away with?

LIGHTKONE

A sweet spot: SEC

- Strong Eventual Consistency (SEC)
  - The data structure is defined so that n replicas that receive the same updates (in any order) have equivalent state
  - Synchronization is eventual replica-to-replica communication

- This consistency model is surprisingly powerful
  - It supports a programming model that resembles a concurrent form of functional programming
  - It handles both nondeterminism and nonmonotonicity
  - It has an efficient, resilient implementation

LIGHTKONE

Let’s exploit SEC!

- In the rest of the lesson we will see how far we can go with Strong Eventual Consistency
  - Convergent data structures (CRDTs)
  - Programming by composing CRDTs (Lasp)
  - Causally consistent transactions on CRDTs (Antidote)
  - Applications

LIGHTKONE