PARA DIME

Speculative Concurrent Processing with Transactional Memory in the Actor Model

OPODIS 2013
December 17, 2013

Yaroslav Hayduk, Anita Sobe, Derin Harmanci, Patrick Marlier and Pascal Felber

unine UNIVERSITÉ DE NEUCHÂTEL

University of Neuchatel, Switzerland

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 318693

A bit of background: The Actor Model

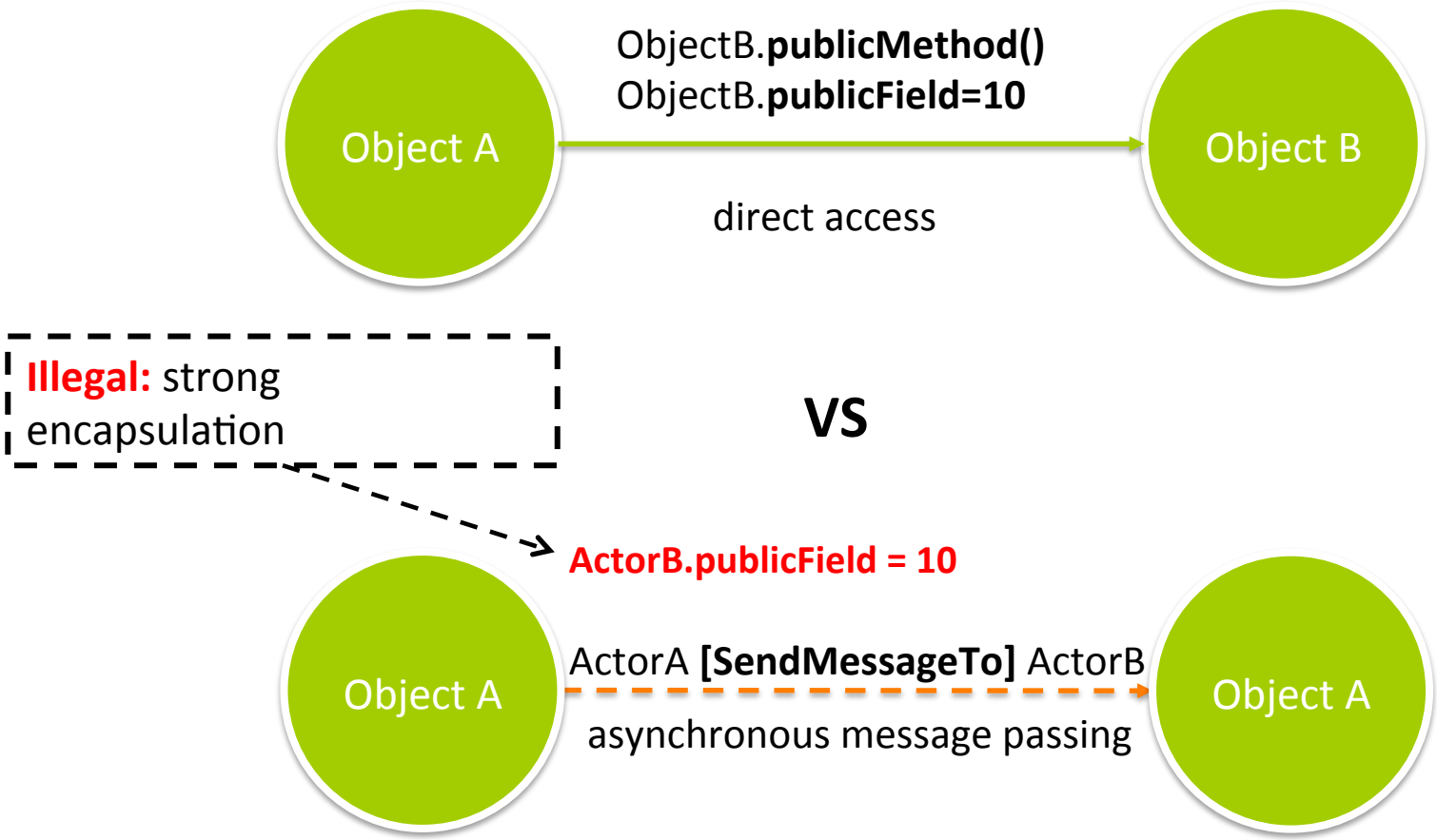
- **Hewitt & Baker (IFIP Congress'77) – „Laws for Communicating Parallel Processes“**

Motivated by the prospect of highly parallel computing machines with many microprocessors + own local memory

OOP and actors: Communication

- Everything is an actor (VS an object)
- **Asynchronous** message passing
- Has access to its local state only
- Strong encapsulation
- Inherently concurrent

OOP and actors: Communication



Problem statement

- Sequential processing of messages limits performance & throughput
- Multiple actors participating in the same coordinated transaction block, causing message processing delays

Main contributions

- A method for processing many messages concurrently using STM, and
- A method for removing excessive blocking, associated with the processing of coordinated transactions

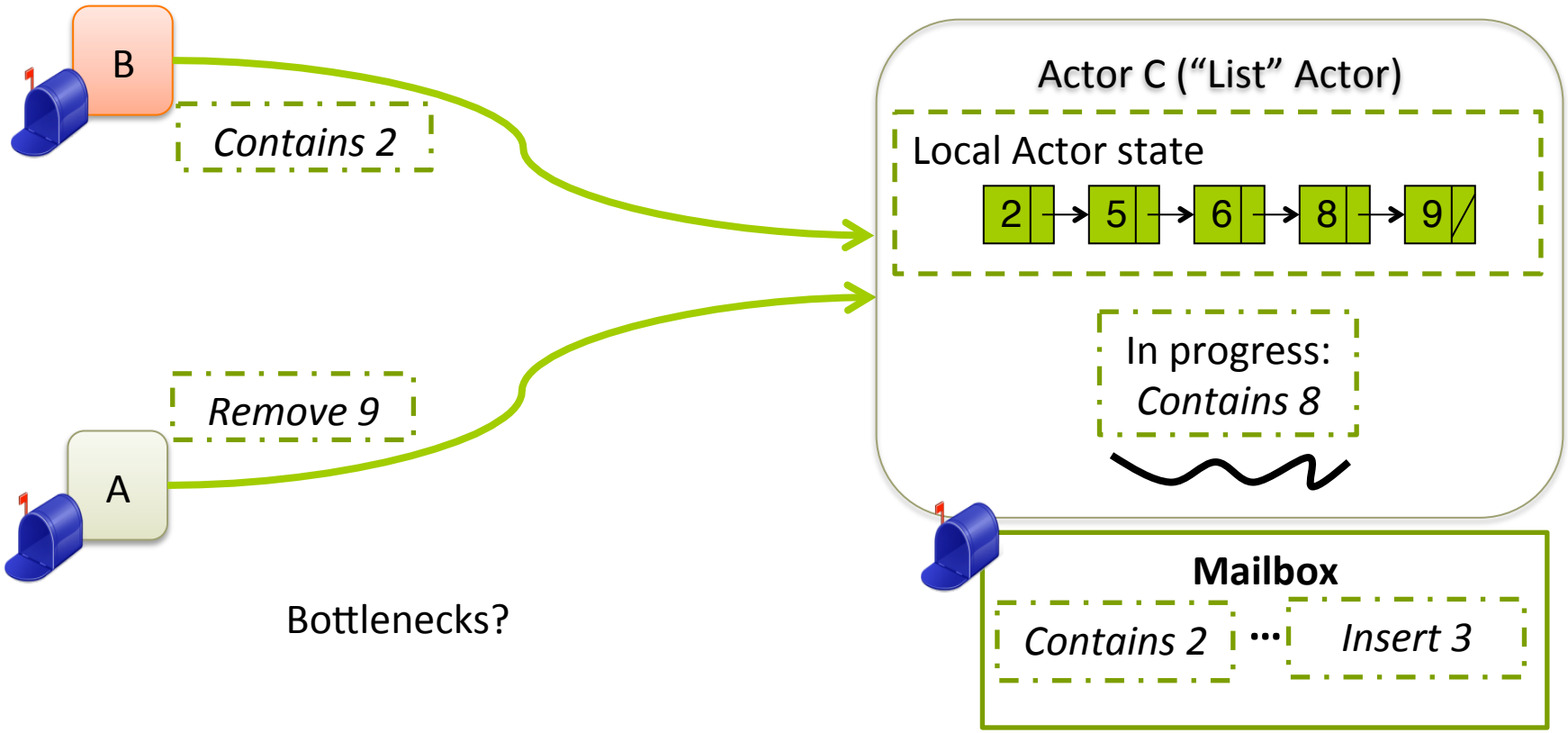
  ⇒both methods preserve the semantics of the Actor Model

A current solution

- Habanero Scala - Shams et al. (Scala Days'12) – Habanero Scala

  ⇒async-finish programming model
- Main strength: processes parts of one message concurrently

Case 1: Concurrent message processing



Concurrent message processing

- Possible issues?

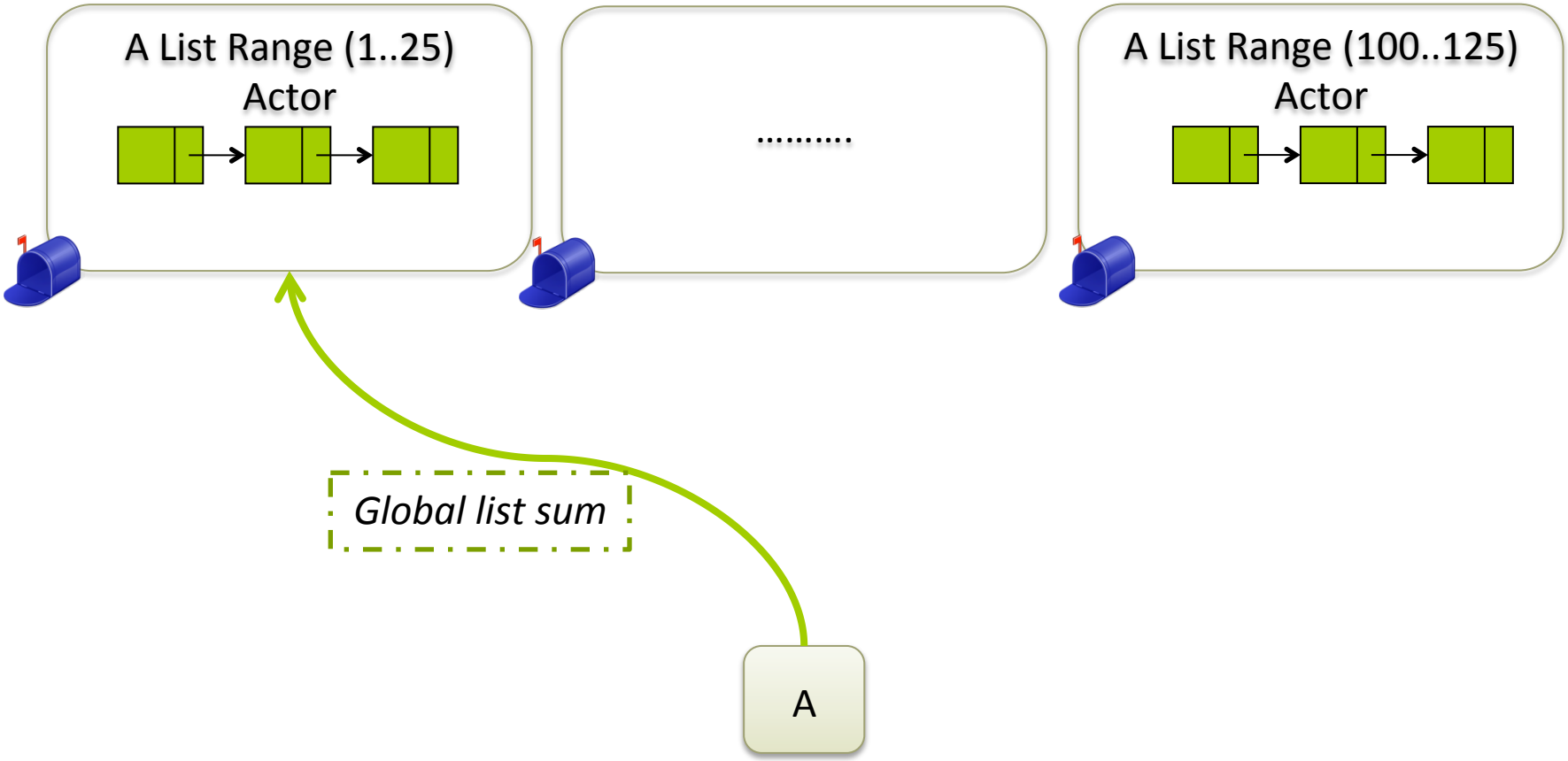
- *List corruption*

  ⇒wrap message processing in an STM transaction.

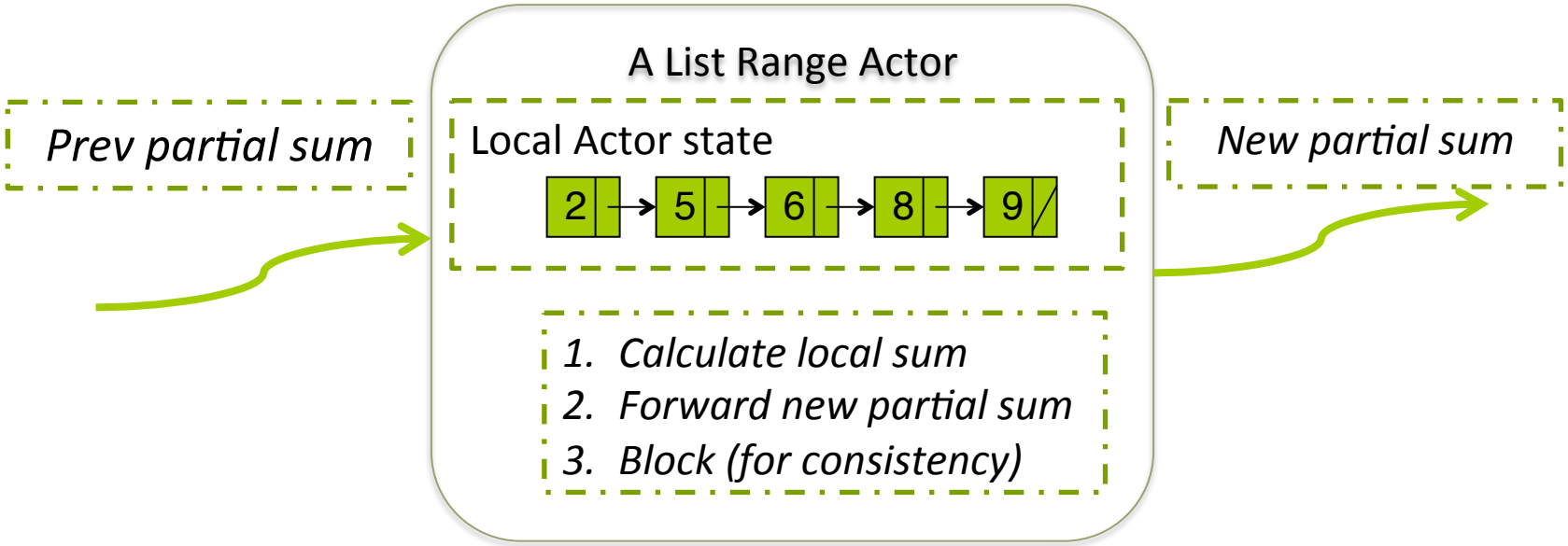
- *Altered message ordering*

  ⇒messages are *not guaranteed* to arrive in order because they are sent asynchronously

Case 2: Coordinated message processing



Global list sum – actor operations magnified

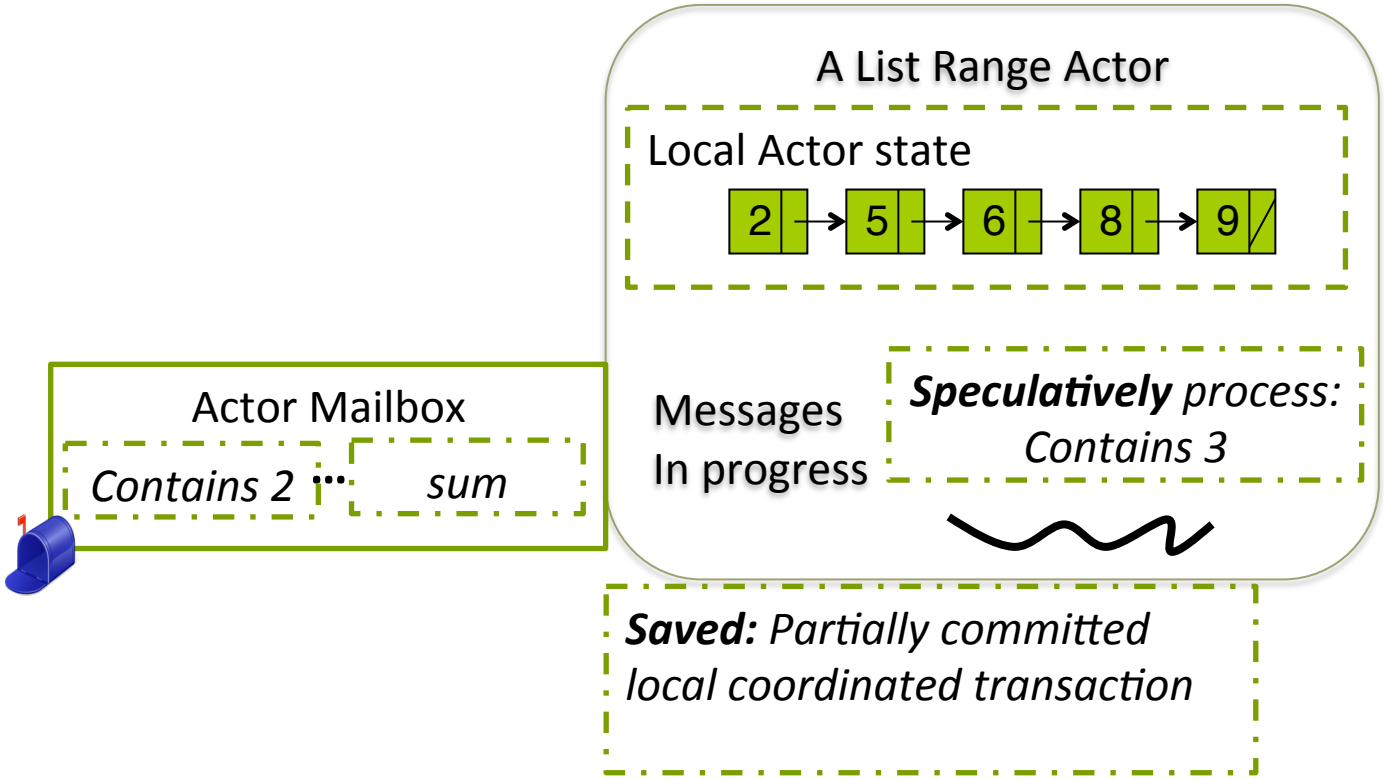


Bottlenecks?

Case 2: Our solution

- Remove blocking; process other messages speculatively
  - ⇒pre-commit the local coordinated transaction
  - ⇒process other messages in a transaction speculatively

Our solution illustrated

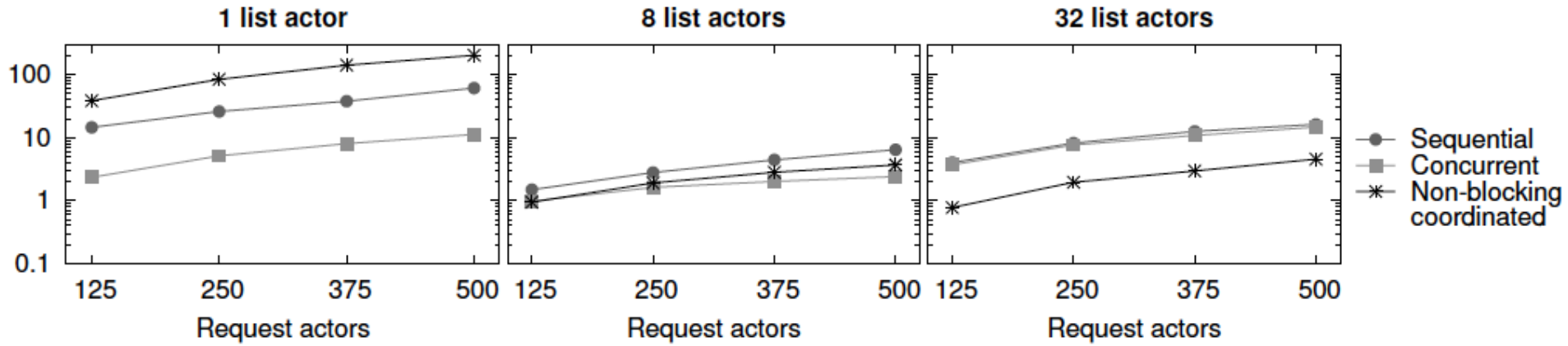


Experimental settings

- Software: a modified Akka 2.10 distribution & Scala 2.10
- Hardware: 48-core AMD Opteron 6172 CPUs running at 2.1GHz
- Application: Stateful distributed sorted integer linked-list

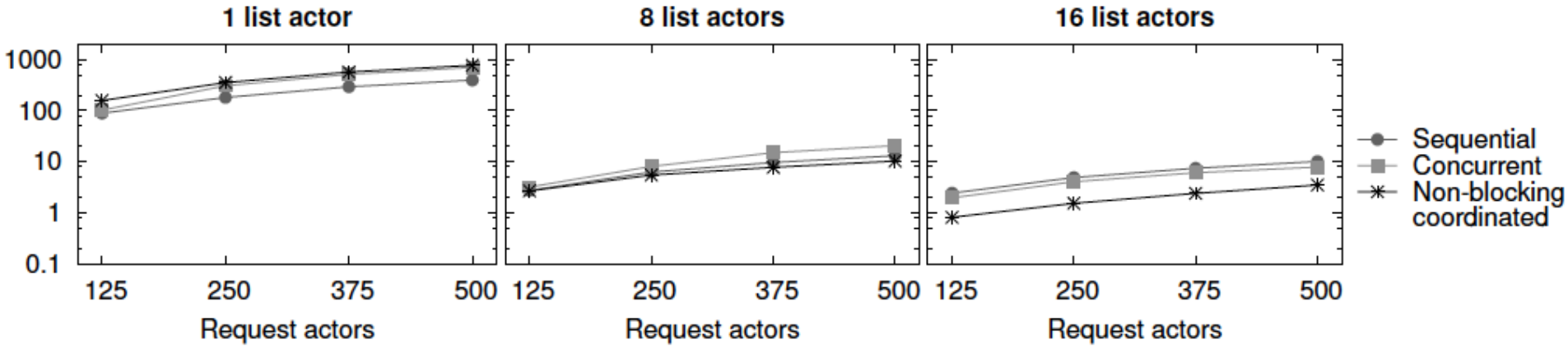
Results

Execution time for sequential, concurrent, and non-blocking: **write-dominated** workload



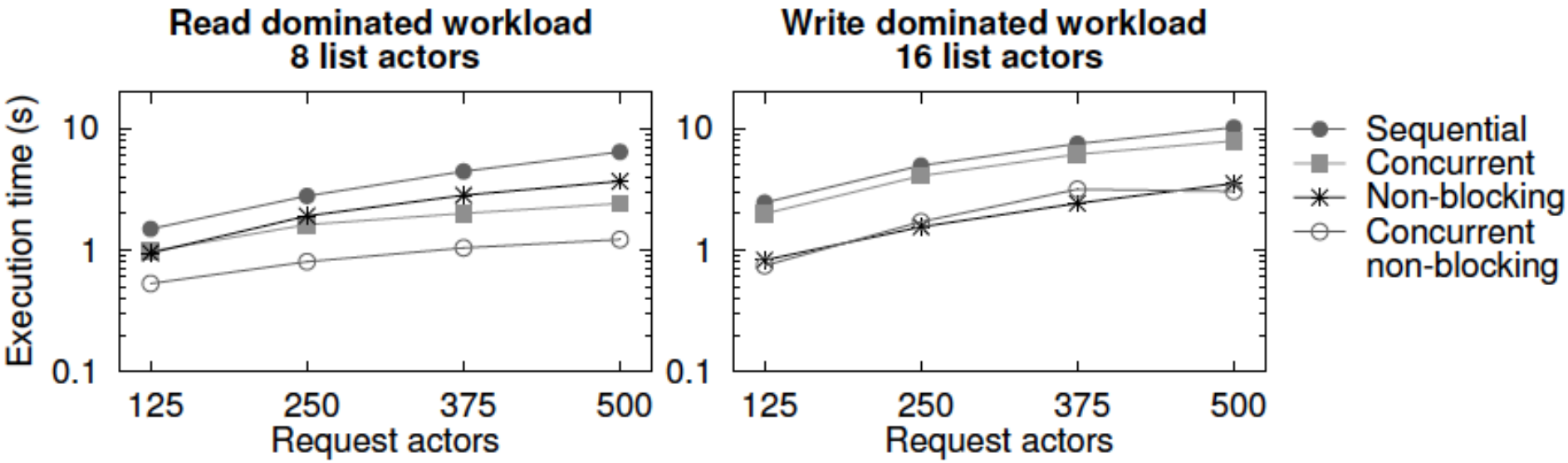
Results

Execution time for sequential, concurrent, and non-blocking: **read-dominated** workload



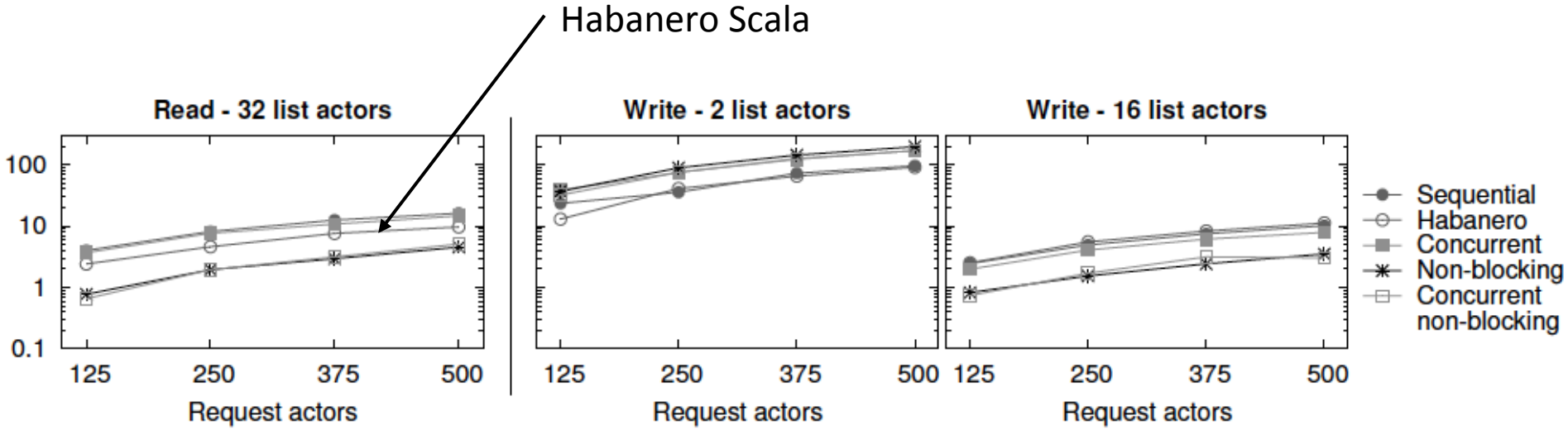
Results

Combined concurrent & non-blocking execution



Results

Comparing with Habanero Scala



Summary

- By using speculation, we can achieve a higher message throughput in the Actor Model
- By using STM we guarantee that the Actor's state is never corrupted

UNIVERSITÉ DE NEUCHÂTEL

Questions?