Concurrent Message Processing using Transactional Memory in the Actor Model

WTM 2013

Pascal Felber, Derin Harmanci,
Yaroslav Hayduk and Anita Sobe

first.last@unine.ch
The Actor Model

- Hewitt & Baker - „Laws for Communicating Parallel Processes“

- Motivated by the prospect of highly parallel computing machines with many microprocessors + own local memory
The Actor Model – basic principles

• 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

Object A

ObjectB.publicMethod()
ObjectB.publicField=10
direct access

VS

ActorA [SendMessageTo] ActorB
asynchronous message passing

Object B
Essential Actor properties

• Processes **one** message at a time
  – *Yey! No race conditions or other hazards*

• No (access to) shared state

• Can send messages to other actors only

• Switch its behavior
Behind the (Actor) scenes

Mailbox

Contains 2

Insert 3

Remove 9

A “List” Actor

Local Actor state

2 → 5 → 6 → 8 → 9

In progress:

Contains 8

Yaroslav Hayduk
Main issues

1. The processing of one message at a time
Notable Related Work

• Scholliers et al. – Parallel Actor Monitors
  – does not use STM for processing messages concurrently

• Shams et al. – Habanero Scala
  – async-finish programming model
  – processes parts of one message (and not many messages) concurrently
Idea:

Process multiple messages concurrently

• Problem: the list can be easily corrupted
• Solution: Wrap each message processing in an STM transaction. When conflicts happen, (roll back and) repeat the message processing
Concurrent message processing

Each message is processed in a transaction
Coordinated Transactions

A List Range (1..25)
Actor

A List Range (100..125)
Actor

Global list sum

A
Local Actor Sum: Zoom-in

A List Range Actor

Local Actor state

Messages In progress

1. Calculate local sum
2. Forward new partial sum
3. Block (for consistency)

Prev partial sum

New partial sum
Main issues

2. Extensive blocking of actors involved in a coordinated transaction
Idea:

Remove blocking; process other messages speculatively

• Pre-commit the local coordinated transaction
• Process other messages in a transaction speculatively
Non-blocking coordinated transactions

A List Range Actor

Local Actor state

[2, 5, 6, 8, 9]

Messages In progress

Speculatively process:
Insert 3

Saved: Partially committed local coordinated transaction

Actor Mailbox
Contains 2

List sum

...
Evaluation

List Actors = 8, Writes = 98%, Sum = 1%
Evaluation

List Actors = 16, Writes = 98%, Sum = 1%
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
Thank you
Questions?