Università della Svizzera italiana

Faculty of Informatics

Efficient Profiling of Actor-based Applications in Parallel and Distributed Systems

Andrea Rosà*, Lydia Y. Chenˆ, and Walter Binder*

*Università della Svizzera italiana (USI), Faculty of Informatics, Lugano, Switzerland
ˆIBM Research Lab Zurich, Rüschlikon, Switzerland

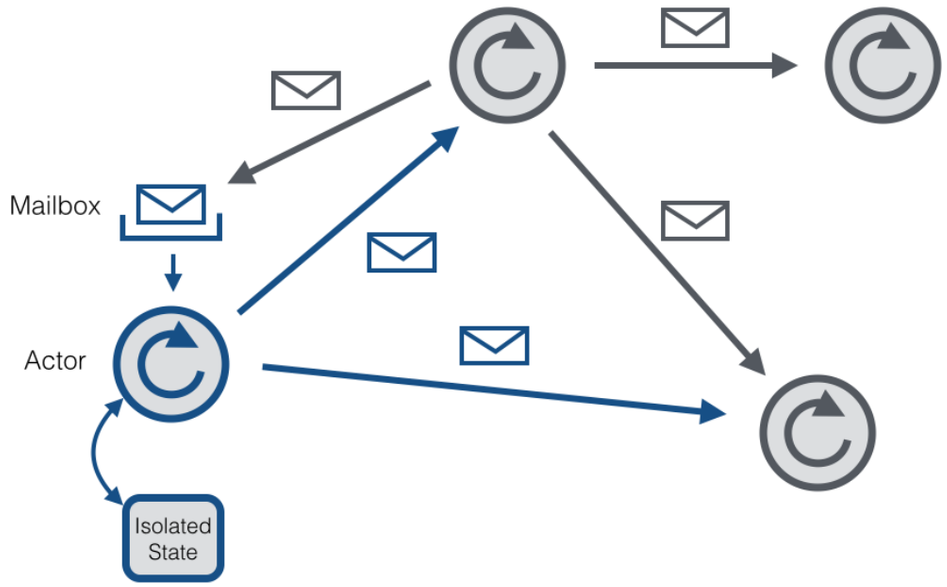
ICOOOLPS 2016
July 18th, 2016
Rome, Italy

Overview

- Position paper on the need for **profiling actors**
- In particular:
  - **Actor utilization**
  - **Communication between actors**
- Preliminary results to support our position

Actors

- Atomic entities communicating via messages
- Continuously listen for incoming messages
- Execute work in response to a message:
  - Send messages
  - Create new actors
  - Change behavior
  - …



Actors

- Properties:
  - Cannot share state
  - Communicate only via asynchronous messages
  - Opaque addressing
- Benefits:
  - Avoid data races
  - Absence of locks in the programming model helps avoid deadlocks
  - Keep the design simple
  - Easy to distribute across cores or machines

Actors in practice

- Programming languages:
  - Erlang
  - Elixir
  - …
- Libraries:
  - Many implementations for Java, C++, Python, .NET, Haskell, …
  - On the JVM: Akka
    - Replaced Scala actors since 2013

Actors in practice

- Applications:
  - Computing workers (e.g., Signal/Collect)
  - Communication endpoints (e.g., Apache Spark, Apache Flink)
  - Used in several commercial products (e.g., Amazon's SimpleDB, Facebook Chat System, WhatsApp)
  - Corpus of Akka applications [1]
- Actors are typically mixed with other concurrent abstractions (e.g., threads, futures)

[1] http://actor-applications.cs.illinois.edu/akka.html

Actor profiling

- Computing workers

Actor utilization

- Communication endpoints

Communication between actors

Actor profiling

Actor utilization

$$\frac{\text{Computations executed by actors}}{\text{Creation cost}}$$

- Low values:
  - Bad division of computations to actors
  - Too many actors wrt. work to be done
  - Small computations per message

Actor profiling

Actor utilization

$$\frac{\text{Computations executed by actors}}{\text{Creation cost}}$$

- Low values:
  - Pinpoint that rethinking the application might be useful:
    - Redesign division of computations to actors
    - Remove some actors

Actor profiling

Actor utilization

$$\frac{\text{Computations executed by actors}}{\text{Creation cost}}$$

- High values:
  - Parallelization opportunities might be missed
  - Depending on idle system resources, it might be beneficial to:
    - Decrease the amount of computations executed per message
    - Add more actors

Actor profiling

Actor utilization

$$\frac{\text{Computations executed by actors}}{\text{Creation cost}}$$

- Expressed as bytecode count
  - Platform-independent
  - Ensures comparable profiles
  - Ensures reproducible profiles for fully deterministic applications and environments
  - Not affected from instrumentation perturbations
  - Requires full bytecode coverage to be accurate

Actor profiling

Communication between actors

- # messages sent/received
- Message types sent/received
- Computations executed per message (type)
- …
- Possible analyses:
  - Analysis of message flow
  - Identify messages that trigger execution of few computations
  - Identify unhandled messages
  - …

Related work

- In general, there is a shortage of profilers for actors
- Exception: Akka [2, 3, 4, 5, 6]
  - Little focus on utilization/communication
  - Typical focus: mailbox size, time in mailbox, errors, dispatchers, …
- Not actor-centric profilers are little applicable to actors
  - Several profilers focus mainly on threads
  - Communication profilers focus mainly on the network stack

---

[2] Lightbend Monitoring. https://www.lightbend.com/products/monitoring.
[3] Takipi. https://www.takipi.com.
[4] Akka Tracing. https://github.com/levkhomich/akka-tracing.
[5] AppDynamics. https://www.appdynamics.com/java/akka/.
[6] NewRelic. https://newrelic.com.

Recap

- Profiling actors can benefit several applications and users
- Tracking actor utilization and communication can lead to useful analyses
- Existing profilers are little adequate

Preliminary evaluation

- We show preliminary evaluation results on the Savina actor-based suite [7]
  - Utilizes actors as computing workers
- We rely on the DiSL dynamic program analysis framework [8]
  - Guarantees full bytecode coverage

---

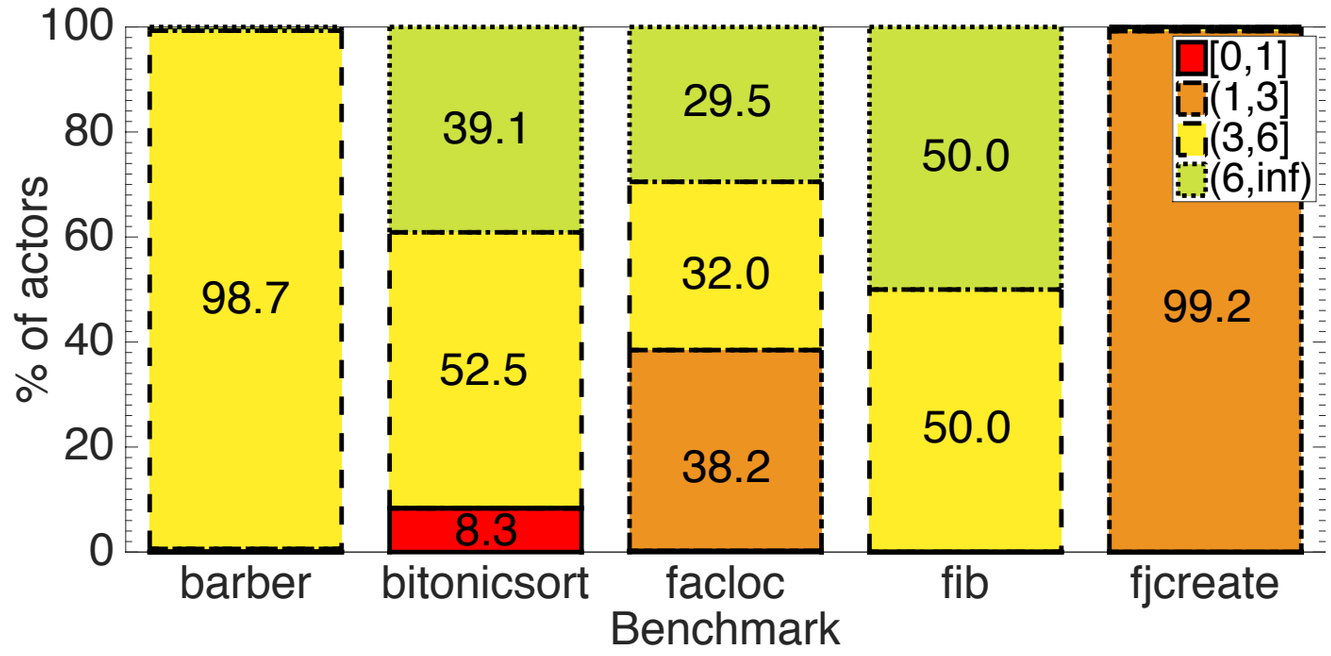
[7] S. M. Imam and V. Sarkar. Savina - An Actor Benchmark Suite: Enabling Empirical Evaluation of Actor Libraries. In AGERE!, pages 67–80, 2014.

[8] L. Marek, A. Villazon, Y. Zheng, D. Ansaloni, W. Binder, and Z. Qi. DiSL: A Domain-specific Language for Bytecode Instrumentation. In AOSD, pages 239–250, 2012.

Preliminary evaluation

Results related to the 5 Akka benchmarks with the highest number of actors

Benchmark	Actors		Messages	
	#	# types	#	# types
barber	5007	7	41474	10
bitonicsort	190525	16	2674730	8
facloc	1370	5	743792	9
fib	150052	4	450197	6
fjcreate	40004	4	80003	5

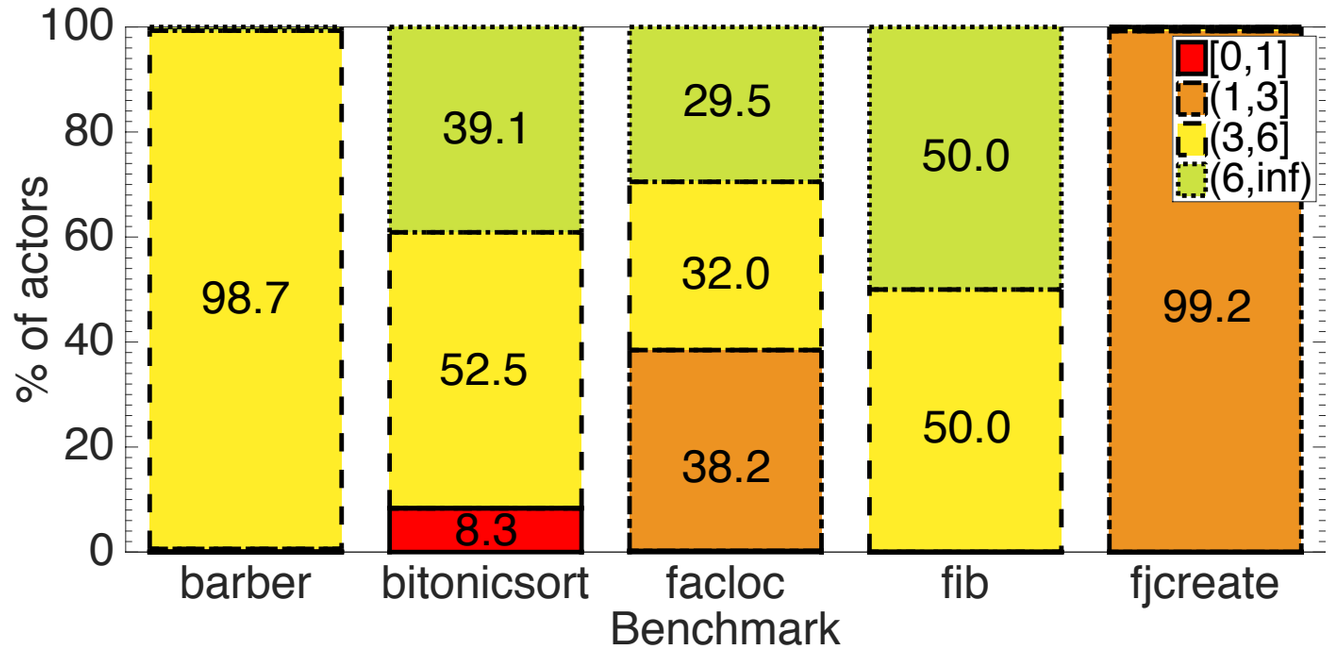


- Many actors are little utilized
- In some actors, message reception might trigger the execution of too little computations
- Potential optimization: increase the amount of computations processed per message

Preliminary evaluation

Results related to the 5 Akka benchmarks with the highest number of actors

Benchmark	Actors		Messages	
	#	# types	#	# types
barber	5007	7	41474	10
bitonicsort	190525	16	2674730	8
facloc	1370	5	743792	9
fib	150052	4	450197	6
fjcreate	40004	4	80003	5



- The system spends resources in creating actors that execute little computations
- Potential optimizations:
  - Redesign assignment of computations to actors
  - Reduce # actors of the same type, preserving application semantics

Conclusions

- Profiling actor utilization and communication can enable optimizations in applications using actors
- Preliminary results encourage further investigation on this topic
- Ongoing work:
  - Design general profiling technique for actors
  - Derive profiler for actor libraries
  - Investigate performance of computing frameworks (e.g., Signal/Collect, Apache Spark, Apache Flink)

Conclusions

Thank you for the attention

- `http://inf.usi.ch/phd/rosaa/icooolps16.pdf`
- Contact detail:

Andrea Rosà
andrea.rosa@usi.ch
http://www.inf.usi.ch/phd/rosaa