Towards a Scalable Programming Platform for Distributed Actors with Debugging Support

Dominik Charousset
dominik.charousset@haw-hamburg.de

iNET RG, Department Informatik
HAW Hamburg

August 2014

Parallel Execution No Longer Optional

- Increasing number of cores, even on mobiles

Parallel Execution No Longer Optional

- Increasing number of cores, even on mobiles
- Specialized HW components are already widely deployed

Parallel Execution No Longer Optional

- Increasing number of cores, even on mobiles
- Specialized HW components are already widely deployed
- Infrastructure software runs in elastic, virtualized environments

Parallel Execution No Longer Optional

- Increasing number of cores, even on mobiles
- Specialized HW components are already widely deployed
- Infrastructure software runs in elastic, virtualized environments

$\Rightarrow$ Established programming paradigms often too low level

The Actor Model of Computation

- Actors are isolated, concurrent software entities

The Actor Model of Computation

- Actors are isolated, concurrent software entities
- Message passing based on *logical* addresses

The Actor Model of Computation

- Actors are isolated, concurrent software entities
- Message passing based on *logical* addresses
- Actors can dynamically create—"spawn"—new actors

The Actor Model of Computation

- Actors are isolated, concurrent software entities
- Message passing based on *logical* addresses
- Actors can dynamically create—“spawn”—new actors
- Error propagation & hierarchical fault management

Previous Work

- Extend the actor model with publish/subscribe semantics
  - Original actor model only foresees 1:1 communication
  - Internet scale requires loose coupling

Previous Work

- Extend the actor model with publish/subscribe semantics
  - Original actor model only foresees 1:1 communication
  - Internet scale requires loose coupling
- `libcppa` – A scalable, native actor library in C++
  - High-performance and embedded environments require efficiency
  - Lightweight actors allow millions of active actors

Previous Work

- Extend the actor model with publish/subscribe semantics
  - Original actor model only foresees 1:1 communication
  - Internet scale requires loose coupling
- `libcppa` – A scalable, native actor library in C++
  - High-performance and embedded environments require efficiency
  - Lightweight actors allow millions of active actors
- Integrated heterogeneous hardware components into `libcppa`
  - GPUs can outperform CPUs by orders of magnitude
  - Transparent integration of OpenCL allows flexible deployment

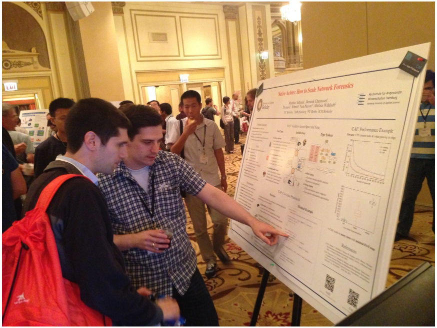
Agenda

Recent Activities – Rebranding

All activities are now bundled as *"CAF: C++ Actor Framework"*

- More than just a library
- `libcppa` was split into `libcaf_core` and `libcaf_io`
- New components were added as optional submodules
- Launched new project homepage `actor-framework.org`
- Moved repository to `github.com/actor-framework`
- Adoption in academia and industry

Recent Activities – Demo at SIGCOMM



- Cooperation with UC Berkeley
- CAF as platform for scalable network forensics (VAST)

Recent Activities – Actors in the IoT

Programming the IoT is challenging

- Constrained HW devices require efficient, resource-aware SW
- Unreliable networking capabilities
- Inherently distributed work flows

$\Rightarrow$ Profound domain knowledge required

Recent Activities – Actors in the IoT

Actor programming as foundation for IoT applications

- The IoT is inherently based on message passing
- Native implementation can scale down to embedded devices
- High level of abstraction improves reusability and testability
  - Program logic independent from deployment
  - Actors can be developed & tested locally
  - Extensible network layer allows to adapt CAF to the IoT

Recent Activities – Actors in the IoT

Actor programming as foundation for IoT applications

- The IoT is inherently based on message passing
- Native implementation can scale down to embedded devices
- High level of abstraction improves reusability and testability
  - Program logic independent from deployment
  - Actors can be developed & tested locally
  - Extensible network layer allows to adapt CAF to the IoT

Specific challenges in CAF

- Error detection & propagation in connectionless networks
- Adapt to limited frame sizes (6LoWPAN)
- Transactional message passing using CoAP

Agenda

Type-safe Message Passing

The original model[1] defines actors in terms of

- Message passing primitives
- Patterns specifed to dispatch on the content of incoming data

⇒ Dynamic type checking

  - Coding errors occur at runtime
  - Non-local dependencies are hard to track manually
  - Extensive integration testing required

[1] Carl Hewitt, Peter Bishop, and Richard Steiger. A Universal Modular ACTOR Formalism for Artificial Intelligence.
In *Proceedings of the 3rd IJCAI*, pages 235–245, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc.

Type-safe Message Passing

Lift type system of C++ and make it applicable to actor interfaces

- Compiler statically checks protocols between actors
- Protocol violation cannot occur at runtime
- Compiler verifies both incoming and outgoing messages:

```
using math =
  typed_actor<
    replies_to<int, int>::with<int>,
    replies_to<float>::with<float, float>>;
// ...
auto ms = typed_spawn(...);
sync_send(ms, 10, 20).then(
  [](float result) {
    // compiler error: result is int, not float
  }
);
```

Agenda

Scheduling Infrastructure

CAF aims at scaling to millions of actors on hundreds of processors

- Actors cannot be implemented as threads
- Running in userspace prohibits preemption

[2] M.L. Dertouzos and AK. Mok. Multiprocessor Online Scheduling of Hard-Real-Time Tasks. *Software Engineering, IEEE Transactions on*, 15(12):1497–1506, Dec 1989

Scheduling Infrastructure

CAF aims at scaling to millions of actors on hundreds of processors

- Actors cannot be implemented as threads
- Running in userspace prohibits preemption
- Previous design deployed a centralized cooperative scheduler
  - Short-lived tasks cause significant runtime overhead
  - Central job queue is a bottleneck
  - *Could* schedule actors for real-time with a priori knowledge [2]

[2] M.L. Dertouzos and AK. Mok. Multiprocessor Online Scheduling of Hard-Real-Time Tasks. *Software Engineering, IEEE Transactions on*, 15(12):1497–1506, Dec 1989

Scheduling Infrastructure

CAF aims at scaling to millions of actors on hundreds of processors

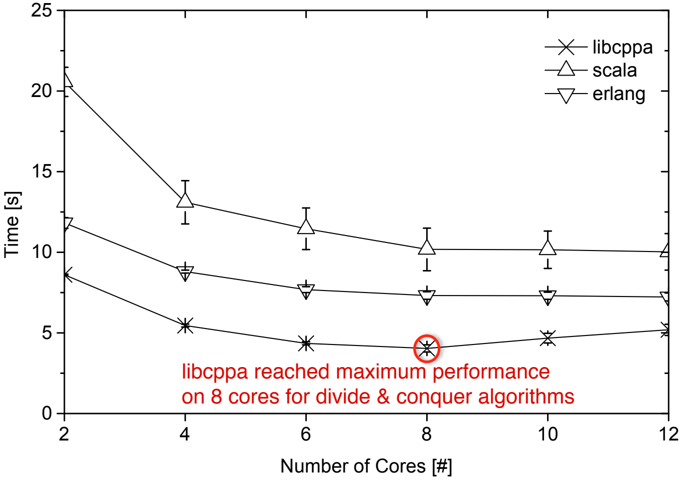
- Actors cannot be implemented as threads
- Running in userspace prohibits preemption
- Previous design deployed a centralized cooperative scheduler
  - Short-lived tasks cause significant runtime overhead
  - Central job queue is a bottleneck
  - *Could* schedule actors for real-time with a priori knowledge [2]

⇒ Decentralized approach required to scale to manycore systems

[2] M.L. Dertouzos and AK. Mok. Multiprocessor Online Scheduling of Hard-Real-Time Tasks. *Software Engineering, IEEE Transactions on*, 15(12):1497–1506, Dec 1989

Scheduling Infrastructure

Divide & conquer with `libcppa` (central scheduling)



Scheduling Infrastructure

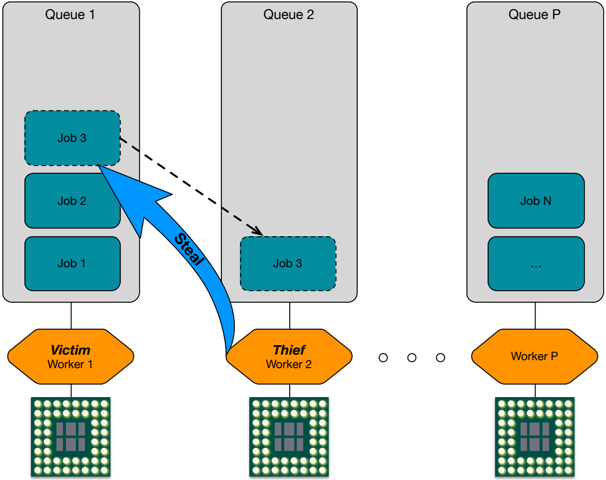
Decentralized scheduling using Work Stealing[3]

- One job queue and worker per core
- Worker tries *stealing* work items from others when idle
- Stealing is a rare event for most work loads[4]
- Widely known variant of work stealing: *fork-join*
- *But:* A priori knowledge cannot be exploited (no global view)

[3] Robert D. Blumofe and Charles E. Leiserson. Scheduling Multithreaded Computations by Work Stealing. *J. ACM*, 46(5):720–748, September 1999.

[4] Vivek Kumar, Daniel Frampton, Stephen M. Blackburn, David Grove, and Olivier Tardieu. Work-stealing Without the Baggage.
In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pages 297–314, New York, NY, USA, 2012. ACM.

Scheduling Infrastructure



Scheduling Infrastructure

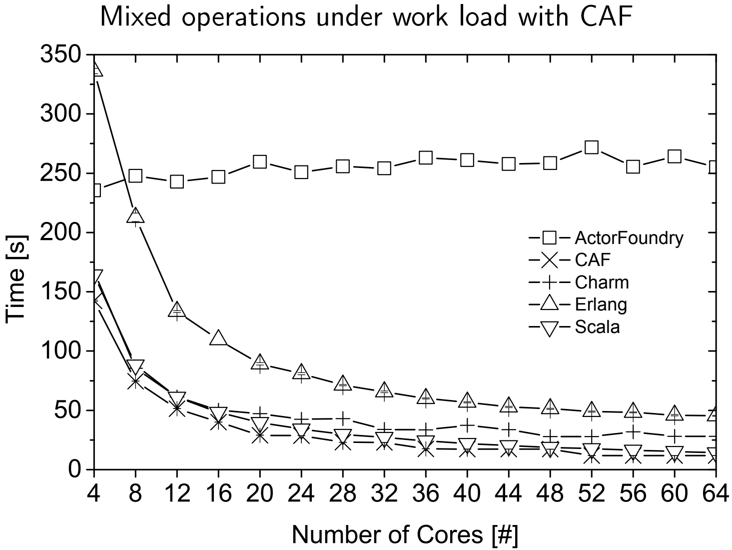
Framework has no a priori knowledge → Work Stealing as default

- Developers can deploy custom scheduler using

```
template <class Policy = work_stealing>
void set_scheduler(size_t num_workers = ...,
                   size_t max_msgs = indefinite);
```

- `max_msgs` restricts nr. of messages actors can consume at once
  - Low value increases fairness and avoids bursts
  - High value minimizes queue access, usually maximizing throughput
- `Policy` can be implemented to exploit a priori knowledge, if possible
- Using Work Stealing, CAF scales up to at least 64 cores

Scheduling Infrastructure



Agenda

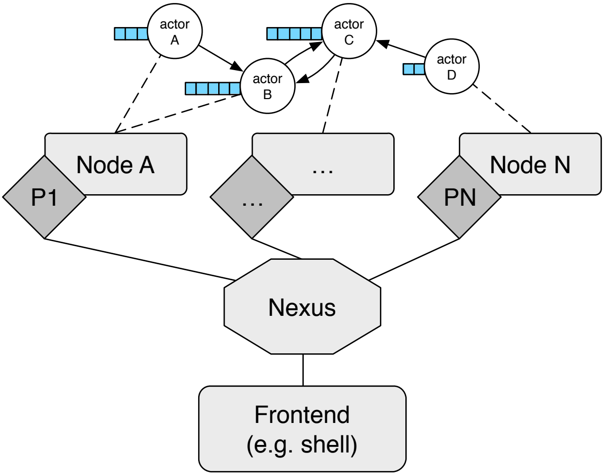
Runtime Inspection & Debugging

- Debugging of distributed systems is inherently complex
- Non-trivial program flow, no global clock, diverging states, etc.
- Recording messages is crucial for on-line or post-mortem debugging
- Erroneous behavior can be reproduced using message replaying [5]
- Visualization tools can help understanding complex errors [6]
- Neither approach has been used to analyze distributed actors

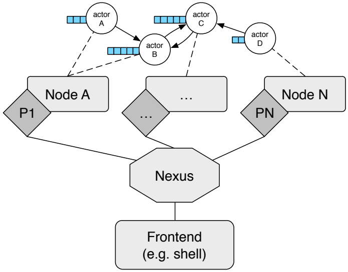
[5] Dennis Michael Geels, Gautam Altekar, Scott Shenker, and Ion Stoica. Replay debugging for distributed applications.
In *Proc. of USENIX'06 Ann. Tech. Conf.*, pages 289–300. USENIX Assoc., 2006.

[6] Terry Stanley, Tyler Close, and Mark S Miller. Causeway: A message-oriented distributed debugger.
Technical Report HPL-2009-78, HP Laboratories, 2009.

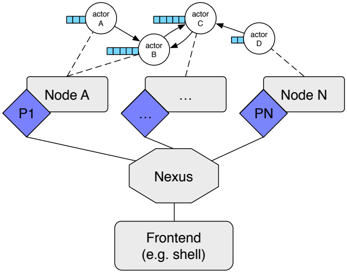
Runtime Inspection & Debugging



Runtime Inspection & Debugging



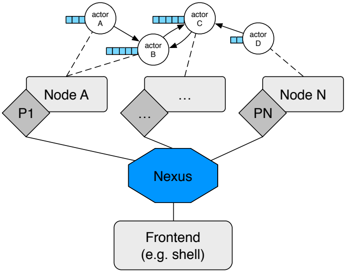
Runtime Inspection & Debugging



Probes

- Intercept & forward three kinds of messages to the Nexus:
  - **Activity events**: incoming & outgoing messages
  - **Error events**: network & system failures
  - **Runtime statistics**: periodic collection of CPU load, etc.

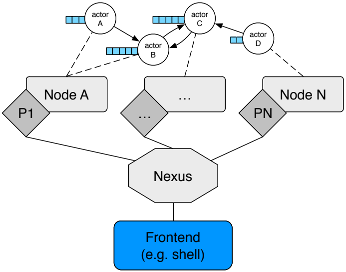
Runtime Inspection & Debugging



The Nexus

- Provides global view of the distributed system
- Receives & collects events from Probes
- Statefully configures verbosity of Probes

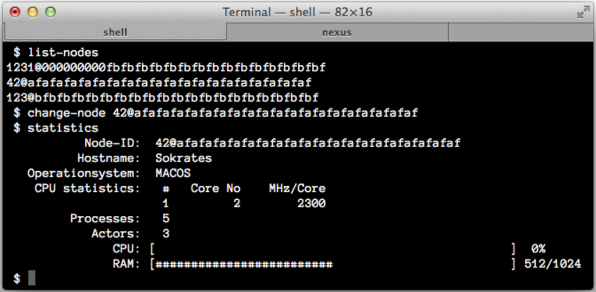
Runtime Inspection & Debugging



Frontend application categories

- **Observing agents**: monitoring & threshold-based alerts
- **Supervising agents**: active manipulation of running app.
- **Monitoring & visualization**: access to aggregate state
  - ⇒ For instance, an *interactive inspection shell*

Interactive Inspection Shell



- Allows users to inspect distributed system
- In global mode:
  - Show all participating nodes
  - Global view to the system, e.g., total number of actors
- In node mode:
  - Access to statistics such as RAM usage, CPU load, etc.
  - Direct interaction with actors on that node

Agenda

Conclusion

- CAF is a robust, scalable platform for actor programming
- Ongoing effort to scale
  - *Down* to IoT devices
  - *Up* to many cores and nodes
- Interactive shell: first step towards debugging distributed actors

Open Research Questions

- Scheduling & distributed load balancing
  - Can we lift realtime capabilities of underlying OS for actors?
  - What are efficient algorithms for actor migration strategies?

Open Research Questions

- Scheduling & distributed load balancing
  - Can we lift realtime capabilities of underlying OS for actors?
  - What are efficient algorithms for actor migration strategies?
- Embedded hardware & communication infrastructure in the IoT
  - How to support fault tolerance in self-healing networks?
  - What is the minimal overhead (RAM, CPU, energy consumption)?

Open Research Questions

- Scheduling & distributed load balancing
  - Can we lift realtime capabilities of underlying OS for actors?
  - What are efficient algorithms for actor migration strategies?
- Embedded hardware & communication infrastructure in the IoT
  - How to support fault tolerance in self-healing networks?
  - What is the minimal overhead (RAM, CPU, energy consumption)?

Open Research Questions

- Scheduling & distributed load balancing
  - Can we lift realtime capabilities of underlying OS for actors?
  - What are efficient algorithms for actor migration strategies?
- Embedded hardware & communication infrastructure in the IoT
  - How to support fault tolerance in self-healing networks?
  - What is the minimal overhead (RAM, CPU, energy consumption)?
- Security considerations
  - How to achieve identity-based cryptography for actors?
  - Opportunistic encryption feasible for CAF in the IoT?

Publications

- Dominik Charousset, Thomas C. Schmidt, Raphael Hiesgen, and Matthias Wählisch. Native Actors – A Scalable Software Platform for Distributed, Heterogeneous Environments.
  In *Proc. of the 4rd ACM SIGPLAN Conference on Systems, Programming, and Applications (SPLASH '13), Workshop AGERE!*, New York, NY, USA, Oct. 2013. ACM
- Matthias Vallentin, Dominik Charousset, Thomas C. Schmidt, Vern Paxson, and Matthias Wählisch. Native Actors: How to Scale Network Forensics.
  In *Proc. of ACM SIGCOMM, Demo Session*, New York, August 2014. ACM
- Raphael Hiesgen, Dominik Charousset, and Thomas C. Schmidt. Embedded Actors – Towards Distributed Programming in the IoT.
  In *Proc. of the 4th IEEE Int. Conf. on Consumer Electronics - Berlin*, ICCE-Berlin'14, Piscataway, NJ, USA, Sep. 2014. IEEE Press

Thank you for your attention!

Homepage: http://actor-framework.org

Sources: https://github.com/actor-framework

iNET Working Group: http://inet.cpt.haw-hamburg.de